```
;
; Edit Date: 08/02/83
;
;_____________________________________________________________
;
;    File: BootROM.TEXT
; Purpose: This is the main flow for the Lisa 1.75 Boot ROM
;
;
          .PROC     BootROM,0
          .ref      MakeDesk
;
ROM0            EQU   $00000000     ;Location of ROM
ROM1            EQU   $00020000     ;Location of second ROM
ROMSize         EQU   $00007FFF     ;Size of each ROM
VideoMemory     EQU   $00000000     ;Location of Video Memory
VidSize         EQU   $00020000     ;Size of Video Memory
LEDon           EQU   $0000300D     ;Turn on  LED on CPU BOARD, disable Serial Port A
LEDoff          EQU   $0000300C     ;Turn off LED on CPU BOARD, enable Serial Port A
;
          move.L    (SP)+,A0        ;save return address
          movem.l   d0-d7/a0-a6,-(sp)  ;Save everyone
;
; Set exception vector pointer to defaults in ROM
        LEA     RESET0,A0         ;Init new Exception Base ******************
;****         MOVE.L  (a0),VBR          ;*******************************************
;
; Set stack to video memory
;
          tst.b     LEDon           ;Turn on CPU board LED
; Set Lisa Video mode.
;
          move.l    #ROM0,a1        ;Address of ROM for Checksum test
          move.l    #ROMSize,d1     ;Length of ROM
          lea       ROM0Done,a0     ;Return address, can not use the stack
          bra       CheckSum        ;Go verify the checksum of first ROM
ROM0Done
;
          move.l    #ROM1,a1        ;Address of ROM for Checksum test
          move.l    #ROMSize,d1     ;Length of ROM
          lea       ROM1Done,a0     ;Return address, can not use the stack
          bra       CheckSum        ;Go verify the checksum of second ROM
ROM1Done
;
          move.l    #VideoMemory,a1 ;Address of Video memory
          move.l    #VidSize,d1     ;Length of Video memory
          lea       VidMDone,a0     ;Return address, can not use the stack
          bra       VidMem          ;Go do video memory test
VidMDone
;
          move.l    #VideoMemory,a1 ;Address of Video memory
          move.l    #VidSize,d1     ;Length of Video memory
          lea       VidPMDone,a0    ;Return address, can not use the stack
          bra       VidPMem         ;Go do video memory parity test
VidPMDone
;
; Write screen area to all ones
; Rest of screen memory to all zeros
;
          jsr       MakeDesk        ;Make a desktop
;
          move.b    #1,d0           ;Timer #1 test (level 6).
          bsr       Timers
;
          bsr       RS232           ;RS232 tests (Level 6).
;
          bsr       IExpansion      ;Ck for expansion and slot 4 inter (5 to 2).
;
          move.b    #1,d0           ;Timer #0 test.
          bsr       Timers
;
          move.b    #1,d0           ;Timer #2 test.
          bsr       Timers
;
          bsr       COPSTest        ;Verify basic COPS operation.
;
;    SetContrast - Set new contrast value.
;    SetVolume   - Set speaker volume.
;    Silence     - Turn off speaker.
;    Beep        - Tones for speaker.
;    Poll        - Polling mode on COPS.
;    Keyboard    - Get keyboard I.D.
;    KeybdEvent  - Get a keyboard event, must also handle COPS error codes.
;    KeybdPeek   - Examine keyboard queue.
;
          bsr       SizeMemory      ;Size memory, find all memory
;
          bsr       MMUBasics       ;MMU read/write & address test.
;
;
          bsr       MEMPatterns     ;Main memory pattern tests.
;
          bsr       MEMParity       ;Main memory parity circuit test.
;
          bsr       MMUFunctional   ;MMU functional test.
;
```

```
;    Floppy driver - Read a sector.
;    Floppy driver - Eject a disk.
;    Floppy driver - See if disk is in.
;    Floppy driver - Debug commands required by Field Service.
          bsr       IWMChip         ;IWM, floppy driver chip, test.
;
;    Built-in hard disk driver - Read status of selftest.
;    Built-in hard disk driver - See if disk is ready yet.
;    Built-in hard disk driver - Read a sector.
          bsr       BuiltIn         ;Builtin hard disk port test
;
          bsr       Test1Expansion  ;Execute expansion card status program
;
          bsr       Test2Expansion  ;Execute expansion card status program
;
          bsr       Test3Expansion  ;Execute expansion card status program
;
          bsr       Test4Expansion  ;Execute expansion card status program
;
          movem.l   (sp)+,d0-d7/a0-a6   ;Restore the world
          jmp       (a0)
;
;_____________________________________________________________________
;_____________________________________________________________________
;==============================
; DEFAULT EXCEPTION HANDLERS
;==============================
RESET0          .LONG   DPROGRAM    ;( 0) Never here.
RESET1          .LONG   DPROGRAM    ;( 1) Never here.
BUSERR          .LONG   DBUSERR     ;( 2)
ADDRERR         .LONG   DADDRERR    ;( 3)
ILLINSTR        .LONG   DILLINSTR   ;( 4)
ZERODIV         .LONG   DPROGRAM    ;( 5)
CHKINSTR        .LONG   DPROGRAM    ;( 6)
TRAPVINSTR      .LONG   DPROGRAM    ;( 7)
PRIVILEGE       .LONG   DPROGRAM    ;( 8)
TRACE           .LONG   DPROGRAM    ;( 9) Never here.
LINE1010        .LONG   DRESERVED   ;(10) Never here.
LINE1111        .LONG   DRESERVED   ;(11) Never here.
RESV1           .LONG   DRESERVED   ;(12) Never here.
RESV2           .LONG   DRESERVED   ;(13) Never here.
FORMATERR       .LONG   DPROGRAM    ;(14)
UNINITINTER     .LONG   DPROGRAM    ;(15)
RESV3           .LONG   DRESERVED   ;(16) Never here.
RESV4           .LONG   DRESERVED   ;(17) Never here.
RESV5           .LONG   DRESERVED   ;(18) Never here.
RESV6           .LONG   DRESERVED   ;(19) Never here.
RESV7           .LONG   DRESERVED   ;(20) Never here.
RESV8           .LONG   DRESERVED   ;(21) Never here.
RESV9           .LONG   DRESERVED   ;(22) Never here.
RESV10          .LONG   DRESERVED   ;(23) Never here.
SPURIOUS        .LONG   DSPURIOUS   ;(24)
LEV1            .LONG   LEV1INT     ;(25)
LEV2            .LONG   LEV2INT     ;(26)
LEV3            .LONG   LEV3INT     ;(27)
LEV4            .LONG   LEV4INT     ;(28)
LEV5            .LONG   LEV5INT     ;(29)
LEV6            .LONG   LEV6INT     ;(30)
LEV7            .LONG   LEV7INT     ;(31)
;
;_____________________________________________________________________
;
DBUSERR                             ; bus error exception
        move.w  $187002,d0          ;get Status register #1
        btst    #9,d0               ;MMU error?
        bne     DMMUERR
        MOVEQ   #100,D7             ; bus error exception (bus timeout)
        BRA     FATAL
;
DMMUERR
        MOVEQ   #101,D7             ; bus error exception (invalid MMU access)
        BRA     FATAL
;_____________________________________________________________________
;
DPROGRAM
        MOVEQ   #102,D7             ; Programming error
        BRA     FATAL
;_____________________________________________________________________
;
DADDRERR
        MOVEQ   #103,D7             ; address error
        BRA     FATAL
;_____________________________________________________________________
;
DILLINSTR
        MOVEQ   #104,D7             ; illegal instruction error
        BRA     FATAL
;_____________________________________________________________________
;
DRESERVED
        MOVEQ   #105,D7             ; Reserved exception
        BRA     FATAL
;_____________________________________________________________________
;
DSPURIOUS
        MOVEQ   #106,D7             ; Spurious exception
```

```
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV1INT move.w  $187000,d0      ;get Status register #0
        btst    #4,d0           ;COPS?
        bne     LEV1COPS
        btst    #0,d0           ;Timer 0?
        bne     LEV1T0
        btst    #2,d0           ;Vertical Interrupt?
        bne     LEV1VI
        btst    #3,d0           ;Hard Disk?
        bne     LEV1HD
        MOVEQ   #107,D7         ; unexpected level 1 unknown interrupt
        BRA     FATAL
;
LEV1COPS MOVEQ   #108,D7        ; unexpected level 1 COPS interrupt
        move.b  $005000,d4      ;Get COPS data for error log table
        move.b  d4,$005000      ;Clear interrupt
        BRA     FATAL
;
LEV1T0  MOVEQ   #109,D7         ; unexpected level 1 Timer 0 interrupt
        move.b  #$00,$004007    ;Write Timer mode register to disable Timer 0
        BRA     FATAL
;
LEV1VI  MOVEQ   #110,D7         ; unexpected level 1 Vertical interrupt
        tst.b   $0030C0         ;Clear interrupt by this address
        BRA     FATAL
;
LEV1HD  MOVEQ   #111,D7         ; unexpected level 1 Hard disk interrupt
        tst.b   $003006         ;Clear interrupt by a) Disable interrupt
        tst.b   $003007         ;                  b) Enable interrupt
        tst.b   $003006         ;Disable interrupt for exiting
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV2INT MOVEQ   #112,D7         ; unexpected level 2 interrupt (expansion slot 4)
        LEA     OLDSR,a6
        and.w   #$f8ff,(a6)     ;Zero interrupt level
        or.w    #$0100,(a6)     ;Change exit interrupt level to 2
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV3INT MOVEQ   #113,D7         ; unexpected level 3 interrupt (expansion slot 3)
        LEA     OLDSR,a6
        and.w   #$f8ff,(a6)     ;Zero interrupt level
        or.w    #$0200,(a6)     ;Change exit interrupt level to 3
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV4INT MOVEQ   #114,D7         ; unexpected level 4 interrupt (expansion slot 2)
        LEA     OLDSR,a6
        and.w   #$f8ff,(a6)     ;Zero interrupt level
        or.w    #$0300,(a6)     ;Change exit interrupt level to 4
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV5INT MOVEQ   #115,D7         ; unexpected level 5 interrupt (expansion slot 1)
        LEA     OLDSR,a6
        and.w   #$f8ff,(a6)     ;Zero interrupt level
        or.w    #$0400,(a6)     ;Change exit interrupt level to 5
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV6INT move.w  $187000;d0      ;get Status register #0
        btst    #0,d0           ;Timer 1?
        bne     LEV6T1
;
        move.b  #3,$006003      ;select register to read
        move.b  $006007,d0      ;...read SCC status
        and.b   #$07,d0         ;is it B?
        bne     LEV6B
        MOVEQ   #116,D7         ; unexpected level 6 RS-232 port A interrupt
        BRA     FATAL
;
LEV6B   MOVEQ   #117,D7         ; unexpected level 6 RS-232 port B interrupt
        BRA     FATAL
;
LEV6T1  MOVEQ   #118,D7         ; unexpected level 6 Timer #1 interrupt
        move.b  #$40,$004007    ;Write Timer mode register to disable Timer 1
        BRA     FATAL
;-------------------------------------------------------------------------
;
LEV7INT move.w  $187000,d0      ;get Status register #0
        btst    #5,d0           ;Parity Error?
        bne     LEV7PE
;
        MOVEQ   #119,D7         ; unexpected level 7 (NMI) interrupt
        BRA     FATAL
;
LEV7PE  MOVEQ   #120,D7         ; unexpected level 7 (parity error) interrupt
        BRA     FATAL
;-------------------------------------------------------------------------
;
;
FATAL
; ****    LEA     EXP,A6
```

```
        move.w  (SP)+,(A6)+     ; ( 1) save Status register
        move.l  (SP)+,(A6)+     ; ( 2) save Program Counter
        move.w  (SP)+,(A6)+     ; ( 4) save Format and vector offset
;****    LEA     FORMAT,A6
        btst    #15,(a6)        ; Long or short format?
        beq     Short
        move.w  (SP)+,(A6)+     ; ( 5) save Special Status word
        move.l  (SP)+,(A6)+     ; ( 6) save Fault address
        move.w  (SP)+,(A6)+     ; ( 8) save Reserved
        move.w  (SP)+,(A6)+     ; ( 9) save Data Output buffer
        move.w  (SP)+,(A6)+     ; (10) save Reserved
        move.w  (SP)+,(A6)+     ; (11) save Data input buffer
        move.w  (SP)+,(A6)+     ; (12) save Reserved
        move.w  (SP)+,(A6)+     ; (13) save Instruction input buffer
        move.w  (SP)+,(A6)+     ; (14) save internal word  1
        move.w  (SP)+,(A6)+     ; (15) save internal word  2
        move.w  (SP)+,(A6)+     ; (16) save internal word  3
        move.w  (SP)+,(A6)+     ; (17) save internal word  4
        move.w  (SP)+,(A6)+     ; (18) save internal word  5
        move.w  (SP)+,(A6)+     ; (19) save internal word  6
        move.w  (SP)+,(A6)+     ; (20) save internal word  7
        move.w  (SP)+,(A6)+     ; (21) save internal word  8
        move.w  (SP)+,(A6)+     ; (22) save internal word  9
        move.w  (SP)+,(A6)+     ; (23) save internal word 10
        move.w  (SP)+,(A6)+     ; (24) save internal word 11
        move.w  (SP)+,(A6)+     ; (25) save internal word 12
        move.w  (SP)+,(A6)+     ; (26) save internal word 13
        move.w  (SP)+,(A6)+     ; (27) save internal word 14
        move.w  (SP)+,(A6)+     ; (28) save internal word 15
        move.w  (SP)+,(A6)+     ; (29) save internal word 16
Short
        BRA     EXIT            ;
;-------------------------------------------------------------------------
;
; Function - Compute a checksum on the memory pointed to.
;
; On entry expects
;   a0 = return address after test is done
;   a1 = start address to check
;   d1 = number of bytes to test
; On exit
;   d0 = 0 for checksum OK, and non-zero for bad checksum
;   d1 is destroyed
;   d2 = Expected checksum
;   d3 = Actual checksum
;
CheckSum
;
        jmp     (a0)
;
;-------------------------------------------------------------------------
;
;Function - Perform memory tests on the video memory
;
; On entry expects
;   a0 = return address after test is done
;   a1 = start address to check
;   d1 = number of bytes to test
; On exit
;   d0 = 0 for memory OK, and non-zero for bad memory
;   Memory is left at all zeros
VidMem
;
        jmp     (a0)
;-------------------------------------------------------------------------
;
; Function - Perform parity tests on video memory
;
; On entry expects
;   a0 = return address after test is done
;   a1 = start address to check
;   d1 = number of bytes to test
; On exit
;   d0 = 0 for memory OK, and non-zero for bad memory parity
;   Memory is left at all zeros
VidPMem
;
        jmp     (a0)
;-------------------------------------------------------------------------
;
; Function - Perform timer chip tests
;
; On entry expects
;   d0 = timer number to test, byte (0 to 2)
; On exit
;   d0 = 0 for timer OK, and non-zero for bad timer
;   a0 = Detailed error table
Timers
;
        rts
;-------------------------------------------------------------------------
;
; Function - Perform RS232 port tests
;
; On entry expects
;   nothing expected
```

```
; On exit
;    d0 = 0 for timer OK, and non-zero for bad timer
;    a0 = Detailed error table
RS232
;
          rts
;_____________________________________________________________________
;
; Function - Check for expansion slot and slot 4 interrupts
;
; On entry expects
;   nothing expected
; On exit
;    d0 = 0 for interrupts OK, and non-zero for stray interrupts coming in
;    a0 = Detailed error table
IExpansion
;
          rts
;_____________________________________________________________________
;
; Function - COPS test, turns on the port, brings in any codes, reads the clock,
;     uses special register read commands to verify COPS (Checksum?), sends
;     keyboard reset command and gets keyboard I.D. to check against previous
;     I.D.  Handles COPS error codes coming in.
;
; On entry expects
;   nothing expected
; On exit
;    d0 = 0 for COPS OK, and non-zero for bad values from COPS
;    a0 = Detailed error table
COPSTest
;
          rts
;
;_____________________________________________________________________
;
; Function - Find memory on other boards.
;
; On entry expects
;   nothing expected
; On exit
;    d0 = 0 for found memory OK, and non-zero for non memory found
;    a0 = Detailed error table
;  Places memory data in table in video memory
SizeMemory
;
          rts
;
;_____________________________________________________________________
;
; Function - MMU read/write & address test.
;
; On entry expects
;   Expects that a memory board exists
; On exit
;    d0 = 0 for MMU OK, and non-zero for bad MMU Ram
;    a0 = Detailed error table
;  Leaves MMU in a state that ......
MMUBasics
;
          rts
;_____________________________________________________________________
;
; Function -  Main memory pattern tests.
;
; On entry expects
;   Expects that a memory board exists
; On exit
;    d0 =* 0 for MMU OK, and non-zero for bad Ram
;    a0 = Detailed error table
;  Leaves Memory written to all zeros.
MEMPatterns
;
          rts
;_____________________________________________________________________
;
; Function - Main memory parity circuit test.
;
; On entry expects
;   Expects that a memory board exists
; On exit
;    d0 = 0 for MMU OK, and non-zero for bad MMU Ram
;    a0 = Detailed error table
;  Leaves MMU in a state that ......
MEMParity
;
          rts
;_____________________________________________________________________
;
; Function - MMU functional test.
;
; On entry expects
;   Expects that a memory board exists
; On exit
;    d0 = 0 for MMU OK, and non-zero for bad MMU Ram
;    a0 = Detailed error table
```

```
;  Leaves MMU in a state that ......
MMUFunctional
;
          rts
;_____________________________________________________________________
;
; Function - IWM, floppy driver chip, test.
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for IWM OK, and non-zero for bad IWM
;   a0 = Detailed error table
;  Leaves IWM in a state that ......
IWMChip
;
          rts
;_____________________________________________________________________
;
; Function - Built-in hard disk port test
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for port OK, and non-zero for bad port
;   a0 = Detailed error table
;  Leaves port in a state that ......
BuiltIn
;
          rts
;_____________________________________________________________________
;
; Function - Execute expansion card 1 status program
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for card OK, and non-zero for bad card
;   a0 = Detailed error table
Test1Expansion
;
          rts
;
;_____________________________________________________________________
;
; Function - Execute expansion card 2 status program
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for card OK, and non-zero for bad card
;   a0 = Detailed error table
Test2Expansion
;
          rts
;
;_____________________________________________________________________
;
; Function - Execute expansion card 3 status program
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for card OK, and non-zero for bad card
;   a0 = Detailed error table
Test3Expansion
;
          rts
;
;_____________________________________________________________________
;
; Function - Execute expansion card 4 status program
;
; On entry expects
;   Expects nothing
; On exit
;   d0 = 0 for card OK, and non-zero for bad card
;   a0 = Detailed error table
Test4Expansion
;
          rts
;
;_____________________________________________________________________
;
          .PROC    MakeDesk,0
          .ref     AIcon_Draw,ADialog,DeskTop,Paint_String,Paint_Ch
;
          move.L   (SP)+,A0          ;save return address
          movem.l  d0-d7/a0-a6,-(sp)  ;Save everyone
;
          jsr      DeskTop           ;Make a blank desktop
;
          move.W   #60,-(SP)         ;x1
          move.W   #30,-(SP)         ;y1
          move.W   #640,-(SP)        ;x2
          move.W   #90,-(SP)         ;y2
```

```
        jsr     ADialog             ;Draw dialog box for main screen
;
        move.W  #90,-(SP)           ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #8,-(SP)            ;Icon code, LISA picture
        jsr     AIcon_Draw          ;Draw LISA picture in box
;
        move.W  #170,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #1,-(SP)            ;Icon code, Big board picture
        jsr     AIcon_Draw          ;Draw Big board picture in box
        move.W  #172,-(SP)          ;x1
        move.W  #60,-(SP)           ;y1
        lea     CPU,a1
        move.L  a1,-(SP)            ;string address
        jsr     Paint_String
;
        move.W  #250,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #2,-(SP)            ;Icon code, memory board picture
        jsr     AIcon_Draw          ;Draw memory board picture in box
;
        move.W  #330,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #3,-(SP)            ;Icon code, Expansion card 1 picture
        jsr     AIcon_Draw          ;Draw Expansion card picture in box
        move.W  #340,-(SP)          ;x1
        move.W  #60,-(SP)           ;y1
        move.w  #'1',-(sp)          ;Character
        jsr     Paint_Ch            ;Place character on the screen
;
        move.W  #410,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #3,-(SP)            ;Icon code, Expansion card 2 picture
        jsr     AIcon_Draw          ;Draw Expansion card picture in box
        move.W  #420,-(SP)          ;x1
        move.W  #60,-(SP)           ;y1
        move.w  #'2',-(sp)          ;Character
        jsr     Paint_Ch            ;Place character on the screen
;
        move.W  #490,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #3,-(SP)            ;Icon code, Expansion card 3 picture
        jsr     AIcon_Draw          ;Draw Expansion card picture in box
        move.W  #500,-(SP)          ;x1
        move.W  #60,-(SP)           ;y1
        move.w  #'3',-(sp)          ;Character
        jsr     Paint_Ch            ;Place character on the screen
;
        move.W  #570,-(SP)          ;x1
        move.W  #50,-(SP)           ;y1
        move.W  #1,-(SP)            ;Icon code, Expansion card 4
        jsr     AIcon_Draw          ;Draw Expansion card picture in box
        move.W  #580,-(SP)          ;x1
        move.W  #60,-(SP)           ;y1
        move.w  #'4',-(sp)          ;Character
        jsr     Paint_Ch            ;Place character on the screen
;
        movem.l (sp)+,d0-d7/a0-a6   ;Restore the world
        jmp     (a0)
;
;
;
CPU     .Byte   4
        .ASCII  'CPU '
        .Byte   0
;
        .END
```