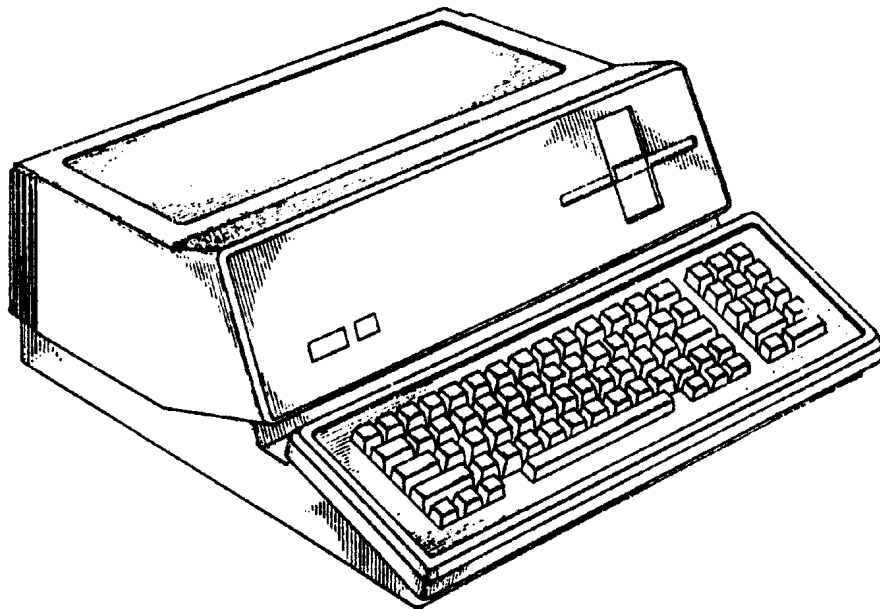


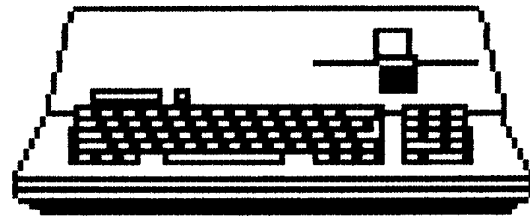


Apple /// Computer Information



APPLE /// SOS BOOTSTRAP LOADER SOURCE

ADDED BY DAVID T CRAIG • 2006



Apple ///
Apple ///+

Apple /// SOS Technical Information

SOS 1.3

Floppy Bootstrap Loader Source Code Listing

This listing shows the code which is found at the beginning of a SOS boot disk. When the Apple /// computer starts the computer's ROM loads this code from the floppy disk and executes the code. This code loads the Apple ///'s operating system, SOS.

Source Code Listing
for

Apple ///

**SOS Floppy
Bootstrap Loader**

David T. Craig
736 Edgewater
Wichita, Kansas 67230

10/31/89 9:45

HD:Apple ///:SOS Floppy Bootstrap Loader

Page 1

```

0000| ;*****
0000| ; APPLE /// BOOTSTRAP LOADER FOR FLOPPY DISK
0000| ; - Disassembled 10-March-1988 by Scott Stinson
0000| ;*****
0000|
0000| .ABSOLUTE
0000| .PROC BOOTSTRAPLOADER
0000| .ORG 0A000
A000|
A000| ;-----
A000| ; EQUATES
A000| ;-----
A000|
A000| ; ZERO PAGE LOCATIONS
A000| ;-----
A000|
A000| 0082 IBDRVN .EQU 82 ; DRIVE NUMBER
A000| 0083 IBTRK .EQU 83 ; TRACK NUMBER
A000| 0084 IBSECT .EQU 84 ; SECTOR NUMBER
A000| 0085 IBBUFP .EQU 85 ; BUFFER POINTER
A000| 0087 IBCMD .EQU 87 ; COMMAND NUMBER
A000| 00E3 IBBUFP TMP .EQU 0E3 ; BUFFER POINTER TEMPORARY
A000| 00E5 FILECNT .EQU 0E5 ; FILE COUNT
A000| 00E7 INDXBLCNT .EQU 0E7 ; INDEX BLOCK COUNT
A000| 00E8 SOSJMPADR .EQU 0E8 ; SOS JUMP ADDRESS
A000|
A000| ;-----
A000| ; HARDWARE I/O ADDRESSES
A000| ;-----
A000|
A000| 0628 SCREENLOC .EQU 0628 ; SCREEN LOCATION
A000| C010 KBDSTROBE .EQU 0C010 ; KEYBOARD STROBE
A000| C040 IOBEEP .EQU 0C040 ; I/O BEEP
A000|
A000| ;-----
A000| ; GENERAL EQUATES
A000| ;-----
A000|
A000| 0040 RETINT .EQU 40 ; RETURN FROM INTERRUPT
A000| 0C00 IDXBLK1 .EQU 0C00 ; INDEX BLOCK 1
A000| 0D00 IDXBLK2 .EQU 0D00 ; INDEX BLOCK 2
A000| 1E00 LOADADR .EQU 1E00 ; LOADING ADDRESS
A000| 1E08 OFFSET .EQU 1E08 ; OFFSET
A000| 2000 FIRSTPAGE .EQU 2000 ; FIRST PAGE
A000| A200 MAINBUFP .EQU 0A200 ; MAIN BUFFER
A000| F000 REGRWTS .EQU 0F000 ; READ/WRITE SECTOR ROUTINE
A000| F4A0 SECTABL .EQU 0F4A0 ; SECTOR TABLE
A000| FFCA NMIVECTOR .EQU 0FFCA ; NON-MASKABLE INTERRUPT VECTOR
A000| FFDF EREG .EQU 0FFDF ; ENVIRONMENT REGISTER
A000| FFEF BREG .EQU 0FEFF ; BANK REGISTER
A000|
A000| ;-----
A000| ; ENTRY POINT
A000| ;-----
A000|
A000| ENTRY SEI ; SET INTERRUPT DISABLE
A001| D8 CLD ; CLEAR DECIMAL FLAG
A002| A9 77 LDA #77 ; LOAD ACCUMULATOR WITH $77
A004| 8D DFFF STA EREG ; STORE IN ENVIRONMENT REGISTER
A007| ; SET 2 MHZ, I/O SPACE ENABLED, SCREEN ENABLED,
A007| ; RESET ENABLED, WRITE PROTECT NOT ENABLED,
A007| ; PRIMARY STACK, AND ROM SELECTED
A007| A2 FF LDX #0FF ; LOAD ACCUMULATOR WITH $FF
A009| 9A TXS ; TRANSFER X-REGISTER TO STACK POINTER
A00A| 2C 10C0 BIT KBDSTROBE ; CLEAR KEYBOARD
A00D| A9 40 LDA #RETINT ; LOAD ACCUMULATOR WITH RETURN FROM INTERRUPT
A00F| 8D CAFF STA NMIVECTOR ; STORE IN NON-MASKABLE INTERRUPT VECTOR
A012| A9 07 LDA #07 ; LOAD ACCUMULATOR WITH $07
A014| 8D EFFF STA BREG ; STORE IN BANK REGISTER
A017| A9 00 LDA #00 ; LOAD ACCUMULATOR WITH $00
A019| CE EFFF $010 DEC BREG ; DECREMENT BANK REGISTER
A01C| 8D 0020 STA FIRSTPAGE ; STORE IN FIRST PAGE OF BANK
A01F| AE 0020 LDX FIRSTPAGE ; LOAD X-REGISTER WITH FIRST PAGE BYTE
A022| D0F5 BNE $010 ; BRANCH IF BYTE IS NOT EQUAL TO $00
A024|
A024| ;-----
A024| ; This section reads in the SOS directory.
A024| ;-----
A024|
A024| A9 00 READSOSDIR LDA #00 ; LOAD ACCUMULATOR WITH $00-BLOCK HIGH BYTE
A026| 85 85 STA IBBUFP ; STORE IN BUFFER POINTER LOW BYTE
A028| A2 A2 LDX #0A2 ; LOAD X-REGISTER WITH $A2
A02A| 86 86 STX IBBUFP+1 ; STORE IN BUFFER POINTER HIGH BYTE
A02C| A2 02 LDX #02 ; LOAD X-REGISTER WITH $02-BLOCK LOW BYTE
A02E| A4 85 RDSOSDIRLP LDY IBBUFP ; LOAD Y-REGISTER WITH BUFFER POINTER LOW BYTE
A030| 84 E3 STY IBBUFP TMP ; STORE IN BUFFER POINTER TEMPORARY LOW BYTE
A032| A4 86 LDY IBBUFP+1 ; LOAD Y-REGISTER WITH BUFFER POINTER HIGH BYTE
A034| 84 E4 STY IBBUFP TMP+1 ; STORE IN BUFFER POINTER TEMPORARY HIGH BYTE
A036| 20 1DA1 JSR READBLK ; JUMP TO READ A BLOCK FROM FLOPPY DISK DRIVE
    
```

DAVID J. CRAIG



ROM



```

A039| A0 02          LDY    #02          ; LOAD Y-REGISTER WITH $02
A03B| B1 E3          LDA    @IBBUPTMP,Y ; LOAD ACCUMULATOR WITH NEXT BLOCK TO READ LOW
A03D|                ; BYTE
A03D| AA            TAX          ; TRANSFER ACCUMULATOR TO X-REGISTER
A03E| C8            INY          ; INCREMENT Y-REGISTER
A03F| B1 E3          LDA    @IBBUPTMP,Y ; LOAD ACCUMULATOR WITH NEXT BLOCK TO READ HIGH
A041|                ; BYTE
A041| D0EB          BNE    RDSOSDIRLP ; BRANCH IF NEXT BLOCK TO READ HIGH BYTE IS NOT
A043|                ; EQUAL TO ZERO
A043| E0 00          CPX    #00          ; CHECK TO SEE IF NEXT BLOCK TO READ LOW BYTE IS
A045|                ; ZERO
A045| D0E7          BNE    RDSOSDIRLP ; BRANCH IF NEXT BLOCK TO READ LOW BYTE IS NOT
A047|                ; EQUAL TO ZERO
A047|
A047|                ;-----
A047|                ; This section searches the SOS directory for the SOS.KERNEL file.
A047|                ;-----
A047| AD 25A2        SRCHSOSKER LDA  MAINBUFF+25 ; LOAD ACCUMULATOR WITH FILE COUNT LOW BYTE
A04A| 85 E5          STA  FILECNT    ; STORE IN FILE COUNT LOW BYTE
A04C| AD 26A2        LDA  MAINBUFF+26 ; LOAD ACCUMULATOR WITH FILE COUNT HIGH BYTE
A04F| 85 E6          STA  FILECNT+1 ; STORE IN FILE COUNT HIGH BYTE
A051| 05 E5          ORA  FILECNT    ; OR ACCUMULATOR WITH FILE COUNT LOW BYTE
A053| D003          BNE    $010    ; BRANCH IF FILE COUNT IS NOT EQUAL TO ZERO
A055| 4C 56A1        JMP    WRNTFNDERR ; JUMP TO WRITE NOT FOUND ERROR MESSAGE TO
A058|                ; SCREEN
A058| A5 E5          $010 LDA  FILECNT    ; LOAD ACCUMULATOR WITH FILE COUNT LOW BYTE
A05A| D002          BNE    $020    ; BRANCH IF NOT EQUAL TO $00
A05C| C6 E6          DEC  FILECNT+1 ; DECREMENT FILE COUNT HIGH BYTE
A05E| C6 E5          $020 DEC  FILECNT    ; DECREMENT FILE COUNT LOW BYTE
A060| A9 2B          LDA  #2B          ; LOAD ACCUMULATOR WITH $28
A062| 85 85          STA  IBBUFF    ; STORE IN BUFFER POINTER LOW BYTE
A064| A9 A2          LDA  #0A2         ; LOAD ACCUMULATOR WITH $A2
A066| 85 86          STA  IBBUFF+1   ; STORE IN BUFFER POINTER HIGH BYTE
A068| AE 24A2        LDX  MAINBUFF+24 ; LOAD X-REGISTER WITH ENTRIES PER BLOCK
A06B| CA            DEX          ; DECREMENT X-REGISTER
A06C| A0 00          SRCHLP  LDY    #00          ; LOAD Y-REGISTER WITH $00
A06E| B1 85          LDA    @IBBUFP,Y ; LOAD ACCUMULATOR WITH STORAGE TYPE AND NAME
A070|                ; LENGTH BYTE
A070| F01A          BEQ    $020    ; BRANCH IF EQUAL TO ZERO
A072| 29 0F          AND    #0F         ; MASK OFF BITS 4,5,6,7
A074| CD 92A1        CMP  FLNMLEN    ; COMPARE WITH FILE NAME LENGTH
A077| D013          BNE    $020    ; BRANCH IF NOT EQUAL TO ZERO
A079| A8            TAY          ; TRANSFER NAME LENGTH TO Y-REGISTER
A07A| B1 85          $010 LDA  @IBBUFP,Y ; LOAD ACCUMULATOR WITH FILE NAME BYTE
A07C| D9 92A1        CMP  FLNME-1,Y  ; COMPARE WITH FILE NAME BYTE
A07F| D00B          BNE    $020    ; BRANCH IF NOT EQUAL
A081| 88            DEY          ; DECREMENT NAME LENGTH
A082| D0F6          BNE    $010    ; BRANCH IF NAME LENGTH NOT EQUAL TO ZERO
A084| B1 85          LDA    @IBBUFP,Y ; LOAD ACCUMULATOR WITH STORAGE TYPE AND NAME
A086|                ; LENGTH BYTE
A086| 29 F0          AND    #0F0       ; MASK OFF BITS 0,1,2,3
A088| C9 20          CMP  #20        ; COMPARE WITH $20 FOR SAPLING FILE
A08A| F032          BEQ    READIDXBLK ; BRANCH IF EQUAL TO READ INDEX BLOCK
A08C| 08            $020 PHP          ; PUSH PROCESSOR STATUS ON STACK
A08D| CA            DEX          ; DECREMENT ENTRIES PER BLOCK
A08E| F010          BEQ    $030    ; BRANCH IF ENTRIES PER BLOCK IS EQUAL TO ZERO
A090| 18            CLC          ; CLEAR CARRY
A091| A5 85          LDA  IBBUFF    ; LOAD ACCUMULATOR WITH BUFFER POINTER LOW BYTE
A093| 6D 23A2        ADC  MAINBUFF+23 ; ADD ENTRY LENGTH LOW BYTE
A096| 85 85          STA  IBBUFF    ; STORE IN BUFFER POINTER LOW BYTE
A098| A5 86          LDA  IBBUFF+1   ; LOAD ACCUMULATOR WITH BUFFER POINTER HIGH BYTE
A09A| 69 00          ADC  #00        ; ADD $00
A09C| 85 86          STA  IBBUFF+1   ; STORE IN BUFFER POINTER HIGH BYTE
A09E| D009          BNE    $040    ; BRANCH ALWAYS
A0A0| A9 04          $030 LDA  #04          ; LOAD ACCUMULATOR WITH $04
A0A2| 85 85          STA  IBBUFF    ; STORE IN BUFFER POINTER LOW BYTE
A0A4| E6 86          LDA  IBBUFF+1   ; INCREMENT BUFFER POINTER HIGH BYTE
A0A6| AE 24A2        LDX  MAINBUFF+24 ; LOAD X-REGISTER WITH ENTRIES PER BLOCK
A0A9| 28            $040 PLP          ; PULL PROCESSOR STATUS FROM STACK
A0AA| F0C0          BEQ    SRCHLP   ; BRANCH IF NOT EQUAL TO ZERO
A0AC| 38            SEC          ; SET CARRY
A0AD| A5 E5          LDA  FILECNT    ; LOAD ACCUMULATOR WITH FILE COUNT LOW BYTE
A0AF| E9 01          SBC  #01        ; SUBTRACT $01
A0B1| 85 E5          STA  FILECNT    ; STORE IN FILE COUNT LOW BYTE
A0B3| A5 E6          LDA  FILECNT+1 ; LOAD ACCUMULATOR WITH FILE COUNT HIGH BYTE
A0B5| E9 00          SBC  #00        ; SUBTRACT $00
A0B7| 85 E6          STA  FILECNT+1 ; STORE IN FILE COUNT HIGH BYTE
A0B9| B0B1          BCS  SRCHLP   ; BRANCH IF MORE FILE ENTRIES
A0BB| 4C 56A1        JMP    WRNTFNDERR ; JUMP TO WRITE NOT FOUND ERROR MESSAGE TO
A0BE|                ; SCREEN
A0BE|
A0BE|                ;-----
A0BE|                ; This section reads in the index block of the SOS.KERNEL file.
A0BE|                ;-----
A0BE|
A0BE| A0 11          READIDXBLK LDY  #11          ; LOAD Y-REGISTER WITH $11
A0C0| B1 85          LDA  @IBBUFP,Y ; LOAD KEY POINTER LOW BYTE
A0C2| AA            TAX          ; TRANSFER ACCUMULATOR TO X-REGISTER-BLOCK LOW
A0C3|                ; BYTE

```

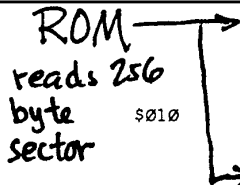
```

A0C3| C8                INY                ; INCREMENT Y-REGISTER
A0C4| B1 85            LDA                @IBBUF, Y ; LOAD KEY POINTER HIGH BYTE
A0C6| A0 00            LDY                #00          ; LOAD Y-REGISTER WITH $00
A0C8| 84 85            STY                IBBUF          ; STORE IN BUFFER POINTER LOW BYTE
A0CA| A0 0C            LDY                #0C          ; LOAD Y-REGISTER WITH $0C
A0CC| 84 86            STY                IBBUF+1        ; STORE IN BUFFER POINTER HIGH BYTE
A0CE| 20 1DA1           JSR                READBLK        ; JUMP TO READ A BLOCK FROM FLOPPY DISK DRIVE
A0D1|
A0D1| ;-----
A0D1| ; This section reads in the first block of the SOS.KERNEL file.
A0D1| ;-----
A0D1| AE 000C            RDISOSKER LDX        IDXBLK1        ; LOAD X-REGISTER WITH INDEX BLOCK LOW BYTE
A0D4| AD 000D           LDA                IDXBLK2        ; LOAD ACCUMULATOR WITH INDEX BLOCK HIGH BYTE
A0D7| A0 00            LDY                #00          ; LOAD Y-REGISTER WITH $00
A0D9| 84 85            STY                IBBUF          ; STORE IN BUFFER POINTER LOW BYTE
A0DB| A0 1E            LDY                #1E          ; LOAD Y-REGISTER WITH $1E
A0DD| 84 86            STY                IBBUF+1        ; STORE IN BUFFER POINTER HIGH BYTE
A0DF| 20 1DA1           JSR                READBLK        ; JUMP TO READ A BLOCK FROM FLOPPY DISK DRIVE
A0E2|
A0E2| ;-----
A0E2| ; This section does a verification of the SOS.KERNEL file to make
A0E2| ; sure it is the proper SOS.KERNEL file. It checks for "SOS KRNL" in
A0E2| ; the first 8 bytes of the file.
A0E2| ;-----
A0E2| A0 08            FLVRFY      LDY                #08          ; LOAD Y-REGISTER WITH $08
A0E4| B9 FF1D           FLVRFYLP    LDA                LOADADR-1, Y ; LOAD ACCUMULATOR WITH BYTE FROM SOS.KERNEL
A0E7| D9 9CA1           CMP                FLVRFY-1, Y ; COMPARE WITH VERIFICATION BYTE
A0EA| F003            BEQ                #010          ; BRANCH IF EQUAL
A0EC| 4C 6AA1           JMP                WRINKERERR     ; JUMP TO WRITE INVALID KERNEL ERROR MESSAGE TO
A0EF|                                     ; SCREEN
A0EF| 88                $010        DEY                ; DECREMENT Y-REGISTER
A0F0| D0F2            BNE                FLVRFYLP      ; BRANCH IF NOT EQUAL TO ZERO TO CHECK REST OF 8
A0F2|                                     ; SOS.KERNEL BYTES
A0F2| ;-----
A0F2| ; This section reads in the SOS.KERNEL file.
A0F2| ;-----
A0F2| A9 01            RDSOSKER    LDA                #01          ; LOAD ACCUMULATOR WITH $01
A0F4| 85 E7            STA                INDXBLKCNT     ; STORE IN INDEX BLOCK COUNT
A0F6| A4 E7            RDSOSKELP  LDY                INDXBLKCNT     ; LOAD Y-REGISTER WITH INDEX BLOCK COUNT
A0F8| BE 000C           LDX                IDXBLK1, Y    ; LOAD X-REGISTER WITH BLOCK LOW BYTE
A0FB| B9 000D           LDA                IDXBLK2, Y    ; LOAD ACCUMULATOR WITH BLOCK HIGH BYTE
A0FE| D004            BNE                #010          ; BRANCH IF BLOCK HIGH BYTE IS NOT EQUAL TO ZERO
A100| E0 00            CPX                #010          ; CHECK TO SEE IF BLOCK LOW BYTE IS NOT EQUAL TO
A102|                                     ; ZERO
A104| F007            BEQ                JUMPSOSKER    ; BRANCH IF BLOCK LOW BYTE IS NOT EQUAL TO ZERO
A107| 20 1DA1           JSR                READBLK        ; JUMP TO READ A BLOCK FROM FLOPPY DISK DRIVE
A109| E6 E7            INC                INDXBLKCNT     ; INCREMENT INDEX BLOCK COUNT
A10B| D0EB            BNE                RDSOSKELP     ; BRANCH IF INDEX BLOCK COUNT IS NOT EQUAL TO
A10B|                                     ; ZERO TO READ MORE OF THE SOS.KERNEL
A10B| ;-----
A10B| ; This section jumps to the SOS.KERNEL loader.
A10B| ;-----
A10B| 18                JUMPSOSKER CLC                ; CLEAR CARRY
A10C| A9 0E            LDA                #0E          ; LOAD ACCUMULATOR WITH $0E
A10E| 6D 081E           ADC                OFFSET        ; ADD OFFSET LOW BYTE
A111| 85 E8            STA                SOSJMPADR     ; STORE IN SOS JUMP ADDRESS LOW BYTE
A113| A9 1E            LDA                #1E          ; LOAD ACCUMULATOR WITH $1E
A115| 6D 091E           ADC                OFFSET+1      ; ADD OFFSET HIGH BYTE
A118| 85 E9            STA                SOSJMPADR+1    ; STORE IN SOS JUMP ADDRESS HIGH BYTE
A11A| 6C E800           JMP                @SOSJMPADR     ; JUMP TO SOS.KERNEL LOADER
A11D|
A11D| ;-----
A11D| ; This section reads a block of data from the floppy disk drive.
A11D| ; On entry the x-register contains the block low byte and the
A11D| ; accumulator contains the block high byte.
A11D| ;-----
A11D| 86 83            READBLK     STX                IBTRK        ; STORE BLOCK LOW BYTE IN TRACK NUMBER
A11F| 4A                LSR                A                ; DIVIDE BLOCK BY 8 TO GET TRACK NUMBER
A120| 66 83            ROR                IBTRK
A122| 4A                LSR                A
A123| 66 83            ROR                IBTRK
A125| 4A                LSR                A
A126| 66 83            ROR                IBTRK
A128| 8A                TXA                ; TRANSFER X-REGISTER WHICH CONTAINS THE BLOCK
A129|                                     ; LOW BYTE TO ACCUMULATOR
A129| 29 07            AND                #07          ; MASK OFF BITS 3,4,5,6,7
A12B| AA                TAX                ; TRANSFER ACCUMULATOR TO X-REGISTER
A12C| BD A0F4           LDA                SECTABL, X    ; LOAD ACCUMULATOR WITH PROPER SECTOR TO READ
A12F| 85 84            STA                IBSECT        ; STORE IN SECTOR NUMBER
A131| A9 01            LDA                #01          ; LOAD ACCUMULATOR WITH $01
A133| 85 87            STA                IBCMD         ; STORE IN COMMAND NUMBER
A135| A9 00            LDA                #00          ; LOAD ACCUMULATOR WITH $00
A137| 85 82            STA                IBDRVN        ; STORE IN DRIVE NUMBER

```

```

A139| 20 00F0      JSR   REGRWTS      ; JUMP TO READ A SECTOR FROM FLOPPY DISK
A13C| 9005         BCC   $010          ; BRANCH IF NO DISK ERRORS OCCURED
A13E| A2 FF       LDX   #0FF         ; LOAD ACCUMULATOR WITH $FF
A140| 9A           TXS   #0           ; TRANSFER X-REGISTER TO STACK POINTER
A141| B03B         BCS   WRDISKERR     ; BRANCH TO WRITE DISK ERROR MESSAGE TO SCREEN
A143| E6 86       INC   IBBUFP+1     ; INCREMENT BUFFER POINTER HIGH BYTE
A145| E6 84       INC   IBSECT        ; INCREMENT SECTOR NUMBER
A147| E6 84       INC   IBSECT        ; INCREMENT SECTOR NUMBER
A149| 20 00F0      JSR   REGRWTS      ; JUMP TO READ A SECTOR FROM FLOPPY DISK
A14C| 9005         BCC   $020          ; BRANCH IF NO DISK ERRORS OCCURED
A14E| A2 FF       LDX   #0FF         ; LOAD ACCUMULATOR WITH $FF
A150| 9A           TXS   #0           ; TRANSFER X-REGISTER TO STACK POINTER
A151| B02B         BCS   WRDISKERR     ; BRANCH TO WRITE DISK ERROR MESSAGE TO SCREEN
A153| E6 86       INC   IBBUFP+1     ; INCREMENT BUFFER POINTER HIGH BYTE
A155| 60           RTS                    ; RETURN TO CALLER
A156|
A156| ;-----
A156| ; This section writes the not found error message to the screen.
A156| ;-----
A156|
A156| A2 1B       WRNTFNDERR LDX   #1B          ; LOAD X-REGISTER WITH $1B
A158| A0 21       LDY   #21          ; LOAD Y-REGISTER WITH $21
A15A| BD A4A1     LDA   $010          ; LOAD ACCUMULATOR WITH NOT FOUND ERROR MESSAGE
A15D|
A15D| 99 2806     STA   SCREENLOC,Y   ; WRITE IT TO THE SCREEN
A160| 88         DEY                    ; DECREMENT Y-REGISTER
A161| CA         DEX                    ; DECREMENT X-REGISTER
A162| D0F6       BNE   $010          ; BRANCH IF MORE CHARACTERS TO WRITE ON SCREEN
A164| AD 40C0     LDA   IOBEEP        ; BEEP SPEEKER
A167| 4C 67A1     JMP   $020          ; HANG FOREVER !!
A16A|
A16A| ;-----
A16A| ; This section writes the invalid kernel error message to the screen.
A16A| ;-----
A16A|
A16A| A2 13       WRINKERERR LDX   #13          ; LOAD X-REGISTER WITH $13
A16C| A0 1D       LDY   #1D          ; LOAD Y-REGISTER WITH $1D
A16E| BD BFA1     LDA   $010          ; LOAD ACCUMULATOR WITH INVALID KERNEL ERROR
A171|
A171| 99 2806     STA   SCREENLOC,Y   ; WRITE IT TO THE SCREEN
A174| 88         DEY                    ; DECREMENT Y-REGISTER
A175| CA         DEX                    ; DECREMENT X-REGISTER
A176| D0F6       BNE   $010          ; BRANCH IF MORE CHARACTERS TO WRITE ON SCREEN
A178| AD 40C0     LDA   IOBEEP        ; BEEP SPEEKER
A17B| 4C 7BA1     JMP   $020          ; HANG FOREVER !!
A17E|
A17E| ;-----
A17E| ; This section writes the disk error message to the screen.
A17E| ;-----
A17E|
A17E| A2 0A       WRDISKERR LDX   #0A          ; LOAD X-REGISTER WITH $0A
A180| A0 18       LDY   #18          ; LOAD Y-REGISTER WITH $18
A182| BD D2A1     LDA   $010          ; LOAD ACCUMULATOR WITH DISK ERROR MESSAGE BYTE
A185| 99 2806     STA   SCREENLOC,Y   ; WRITE IT TO THE SCREEN
A188| 88         DEY                    ; DECREMENT Y-REGISTER
A189| CA         DEX                    ; DECREMENT X-REGISTER
A18A| D0F6       BNE   $010          ; BRANCH IF MORE CHARACTERS TO WRITE ON SCREEN
A18C| AD 40C0     LDA   IOBEEP        ; BEEP SPEEKER
A18F| 4C 8FA1     JMP   $020          ; HANG FOREVER !!
A192|
A192| ;-----
A192| ; STORAGE FOR THE ERROR MESSAGE AND FILE VERIFICATION ROUTINES
A192| ;-----
A192|
A192| 0A         FLNMELEN  .BYTE  0A
A193| 53 4F 53 2E 4B 45 52  FLNME   .ASCII  "SOS.KERNEL"
A19A| 4E 45 4C
A19D| 53 4F 53 20 4B 52 4E  FLVERIFY .ASCII  "SOS KRNL"
A1A4| 4C
A1A5| 46 49 4C 45 20 27 53  NTFNDERR .ASCII  "FILE 'SOS.KERNEL' NOT FOUND"
A1AC| 4F 53 2E 4B 45 52 4E
A1B3| 45 4C 27 20 4E 4F 54
A1BA| 20 46 4F 55 4E 44
A1C0| 49 4E 56 41 4C 49 44  INVKEERR .ASCII  "INVALID KERNEL FILE"
A1C7| 20 4B 45 52 4E 45 4C
A1CE| 20 46 49 4C 45
A1D3| 44 49 53 4B 20 45 52  DISKERR   .ASCII  "DISK ERROR"
A1DA| 52 4F 52
A1DD|
A1DD| .END
    
```



SYMBOL TABLE DUMP

AB - Absolute LB - Label UD - Undefined MC - Macro
 RF - Ref DF - Def PR - Proc FC - Func
 PB - Public PV - Private CS - Consts

BOOTSTRA PR ---- | BREG AB FFEF | DISKERR LB A1D3 | ENTRY LB A000 | EREG AB FFDF |

10/31/89 9:45

HD:Apple ///:SOS Floppy Bootstrap Loader

Page 5

```

FILECNT AB 00E5 | FIRSTPAG AB 2000 | FLNME LB A193 | FLNMELEN LB A192 | FLVERIFY LB A19D |
FLVRFY LB A0E2 | FLVRFYLP LB A0E4 | IBBUFP AB 0085 | IBBUPTM AB 00E3 | IBCMD AB 0087 |
IBDRVN AB 0082 | IBSECT AB 0084 | IBTRK AB 0083 | IOBEEP AB C040 | JUMPSOSK LB A10B | KBDSTROB AB C010 |
INDXBLKC AB 00E7 | INVKEERR LB A1C0 | IOBEEP AB C040 | JUMPSOSK LB A10B | KBDSTROB AB C010 |
LOADADR AB 1E00 | MAINBUFF AB A200 | NMIVECTO AB FFCA | NTFNDERR LB A1A5 | OFFSET AB 1E08 |
RD1SOSKE LB A0D1 | RDSOSDIR LB A02E | RDSOSKEL LB A0F6 | RDSOSKER LB A0F2 | READBLK LB A11D |
READIDXB LB A0BE | READSOSD LB A024 | REGRWTS AB F000 | RETINT AB 0040 | SCREENLO AB 0628 |
SECTABL AB F4A0 | SOSJMPAD AB 00E8 | SRCHLP LB A06C | SRCHSOSK LB A047 | WRDISKER LB A17E |
WRINKERE LB A16A | WRNTFNDE LB A156 |
    
```

Assembly complete: 363 lines
 0 Errors flagged on this Assembly

 6502 OPCODE STATIC FREQUENCIES

```

ADC : 4 | ****
AND : 3 | ***
BCC : 2 | **
BCS : 3 | ***
BEQ : 6 | *****
BIT : 1 m *
BNE : 15 | *****
CLC : 2 | **
CLD : 1 m *
CMP : 4 | ****
CPX : 2 | **
DEC : 3 | ***
DEX : 5 | *****
DEY : 5 | *****
INC : 6 | *****
INY : 2 | **
JMP : 7 | *****
JSR : 6 | *****
LDA : 37 M *****
LDX : 12 | *****
LDY : 14 | *****
LSR : 3 | ***
ORA : 1 m *
PHP : 1 m *
PLP : 1 m *
ROR : 3 | ***
RTS : 1 m *
SBC : 2 | **
SEC : 1 m *
SEI : 1 m *
STA : 23 | *****
STX : 2 | **
STY : 6 | *****
TAX : 3 | ***
TAY : 1 m *
TXA : 1 m *
TXS : 3 | ***
    
```

Minimum frequency = 1
 Maximum frequency = 37

Average frequency = 5

Unused opcodes:

ASL BMI BPL BRK BVC BVS CLI CLV CPY EOR INX NOP PHA PLA ROL RTI
 SED TSX TYA

Program opcode usage: 66 %

 (1.00) That's all, Folks ...
