Instruction Manual

MON 86 — V1.4

8086 Monitor

For Use with the SCP 300 CPU Support Board

Seattle Computer Products, Inc.
1114 Industry Drive, Seattle, WA. 98188
(206) 575-1830

Rev. 1.4A

CONTENTS

Getting Started

Connect an RS-232 terminal to the cable coming from J1 of the CPU support card. The terminal should be set for full duplex at one of the following rates: 19200, 9600, 1200, 300, 150, or 110 baud. The software-selected baud rate feature of the CPU Support card is used to automatically determine the baud rate of the terminal. By hitting the carriage return no more than four times, the sign-on message should appear. If it does not, reset the computer and try again. If it still does not sign on, check all connections carefully.

If Sense Switch 0 is a one (position 1 of S2 is closed), then the monitor will NOT sign on after baud rate selection but instead will automatically boot the disk. This is equivalent to the Boot command with no parameters.

Directly below the sign-on message there will be a greater-than symbol, ">". This is the Monitor prompt, and indicates that the Monitor is ready to accept a command. The input buffer allows commands of up to 80 characters in length. While typing the command line, <backspace> and <rubout> or <delete> may be used back up to correct a mistake, while "@" cancels the line and re-issues the prompt. Typing <carriage return> either causes the command to be executed or an error to be reported. Most errors are syntax errors, and an arrow followed by the word "Error" will appear under the first bad character. If an error occurs, no part of the command is executed (except during boot or flag replacement - see Boot and Register commands).

Monitor commands are available to display, alter and search memory; to do inputs and outputs; to boot the disk; and to aid in debugging 8086 programs. The debugging commands allow the user to execute a program in a controlled manner, observing its behavior. This controlled execution may be done either by single-stepping or through execution with breakpoints.

Single-stepping is done with the Monitor's Trace command. By using 8086 hardware trace mode, a single instruction can be executed, and the resulting effects on the registers or memory displayed. Even ROM may be traced, and every instruction is traced correctly (unlike 8080 or Z80 debuggers).

Execution with breakpoints (Go command) allows the user to quickly execute previously tested program portions but stops program execution if a breakpoint is reached. Breakpoints require more care than single-stepping since they can only be used in RAM at the address of the first byte of an 8086 opcode.

Both methods of "controlled execution" allow the user to modify or examine CPU registers. A "register save area" is maintained in memory: just before execution, all registers are set with values from this area; and when control is returned to the monitor, all registers are saved back in this area. The Register command allows this area to be displayed or modified.

Execution of any command may be aborted by typing Control-C. Typing Control-S during output will cause the display to pause so it may be read before scrolling away; any key (except Control-C) may be typed to continue.

If a user program is executing as a result of a Boot or Go command and interrupts are enabled, then the console may interrupt the program and return control to the Monitor. Typing any key will cause the interrupt, save program status, and print a register dump; except that Control-C will inhibit the register dump. Note that complete program status is always saved, and execution may be continued with a Go or Trace command.

The Monitor requires .5K of memory at address zero. Specifically, interrupt vectors are kept at locations 4-7, 0CH-0FH, and 64H-67H, while scratch pad ram, input buffer, and stack use less than 256 bytes beginning at 100H. User programs must not modify these locations if the Monitor is to be used for debugging.

Parameters

All commands of the Monitor accept one or more parameters on the line following the command letter. These parameters MAY be separated from each other and the command letter by spaces or commas, but one these delimiters is REQUIRED only to separate consecutive hex values. Most parameters are one of the following types:

<BYTE>, <HEX4>, <ADDRESS> - A hexadecimal number with no more than 2, 4, or 5 digits, respectively. Thus, <BYTE> becomes an 8-bit value, <HEX4> a 16-bit value, and <ADDRESS> a 20-bit value. If too many digits are entered or a non-hex character is typed, the error arrow will point to the mistake. Hex A-F must be in upper case.

<RANGE> - A <RANGE> is either <ADDRESS> <ADDRESS> or <ADDRESS> L <HEX4>. The first form specifies the first and last addresses affected by the command. The second form specifies a starting address and a length. For either form, the maximum length (first address - last address + 1) cannot exceed 10000H, and this limit may be as low as 0FFF1H due to limitations of working within a segment. (Specifically, [starting address modulo 16] + length must be <= 10000H.) An "RG Error" results if the length is too large. To specifiy a length of 10000H with only four digits, use a length of zero. Note that the "L" in this form must be upper case.

<LIST> - This is always the last parameter on a line and may extend to the end of the input buffer. It is actually a series of one or more parameters, each of which is either a <BYTE> or a <STRING>.

A <STRING> is any number of characters (except control characters) enclosed by either single (') or double (") quotes. Since the opening and closing quotes must be the same, the other type may appear in the string freely. If the same quote as opened the string needs to appear within it, it must be given as two adjacent quotes. The ASCII values of the characters in the string are used as a list of bytes.

Commands

A command is executed by typing the first letter of its name (upper case only) followed by any parameters. If the first letter on the line is not recognized as a command, the error arrow will point to it. Commands are listed below in alphabetical order, with the forms of all parameters shown.

B
B <ADDRESS> . . . <ADDRESS>

Boot - Loads the first sector of track 0 of the disk into memory starting at 200H. Up to ten 5-digit addresses may be specified; too many will cause a "BP Error". After the sector is loaded, breakpoints will be set at these locations. Then all registers will be set from the register save area, except that the Code Segment will be set to zero, and the Instruction Pointer will be set to 200H - thus a jump will be made to 200H. The user stack pointer MUST be valid for this command to work. See Go command for more information.

This command works in three steps. First, the disk sector is loaded. Next, the Code Segment and Instruction Pointer are set in the register save area. Finally, a Go command is executed. The result is that an error in a breakpoint address will not be found until AFTER the sector is loaded and the register save area changed. Thus it is not necessary to use another Boot command to correct the error; a Go command with the corrected breakpoints will do.

The example below shows how Boot can help test an experimental 8086 program. The program to be tested fits into one 128-byte sector and has been placed on track 0, sector 1 of a disk. The program is loaded with the Boot command but execution does not begin because a breakpoint is set at 200H, the first byte of loaded program. Before testing, the program is moved to 400H, just above the interrupt table, and CS and IP are adjusted.

```
SCF 8086 Monitor 1.4
>B200

AX=0000  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0000  IP=0200   NV UP EI PL NZ NA PO NC
> M 200 L80 400
>RCS
CS 0000
:40
>R IP
IP 0200
:0
>R
AX=0000  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV UP EI PL NZ NA PO NC
>
```

D <ADDRESS>
D <RANGE>
Dump - Displays memory contents in hex and ASCII. If only a starting address is specified, 80H bytes are dumped; otherwise the specified range is displayed. To help pinpoint addresses, each line (except possibly the first) begins on a 16-byte boundary, and each 8-byte boundary is marked with a "-". Non-printing characters are shown as a "." in the ASCII dump.

```
>D400 L29
00400 FF FB FF FF F7 7F FF FF-FF FE 7F FF FF FF FF FF   .{..w....~......
00410 DD FB DF FF CF FF FE DF-FF FF 7F FB FB FD FF F7   ]{_.O.~_...{{}.w
00420 BF FF BF FF BF BF 6F FF-FF                        ?.?.??o..
>
>D445 463
00445 FF DF 7F-F9 FF 7E FF FE FF FF FF                  ._.y.~.~...
00450 FF FF FF FF FF FF FF DF-FF D7 FF FF FF FF FF FF   ......._.W......
00460 9F FF FA FF                                       ..z.
>
>D80
00080 FF DF FF FF DF FF FF FF-F5 FF FF FD FF F5 FF 7F   ._.._...u..}.u..
00090 CE FF FF FB FF FB FF FF-7F FF FE FA FD FA FF FB   N..{.{....~z}z.{
000A0 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF DF DF   ..............__
000B0 FF FF FB BF FF FF 5F EF-FF FF FA FF FF DF F7 FD   ..{?.._o..z.._w}
000C0 FF FF FF FF FF FF FF FF-7F FF FF FF FF E6 FF FF   .............f..
000D0 FF DF FF FF FF DC FB 7F-FE FF FF FF FF DB ED FF   ._...\{.~....[m.
000E0 FF FF FF FF FB FF FF FF-FF FF FF FF FF FF 5F FF   ....{........._.
000F0 DF F7 FF DE FF FF FF BD-BF BF F9 FB DF FF DF DF   _w.^...=??y{_.__
```

E <ADDRESS> <LIST>
E <ADDRESS>

Enter - In the first form, the list of bytes is entered at the specified address, with the command being executed and completed upon hitting <carriage return>. If an error occurs, NO locations are changed.

The second form puts the Monitor into "Enter Mode", starting at the specified address. After hitting <carriage return>, the address and its current contents will be displayed. The user now has several options:

1) Replace the displayed value with a new value. Simply type in the new value in hex, using <backspace> or <delete> to correct mistakes. If an illegal hex digit is typed or more than two digits are typed, the bell will sound and the character will not be echoed. After entering the new value, type either <space>, "-", or <carriage return>, as defined below.

2) Type <space> to display and possibly replace the next memory location. Every 8-byte boundary will start a new line with the current address.

3) Type "-" to backup to the preceding memory location. This will always start a new line with the address. The "-" will not be echoed.

4) Type <carriage return> to terminate the command.

```
>E500 24,9,A 'Test',0
>D 500 L10
00500 24 09 0A 54 65 73 74 00-00 20 00 00 00 40 01 00   $..Test.. ...@..
>
>E508
00508 00.
00507 00.
00506 74.     00.49
00508 00.4E   20.47   00.0    00.0    00.0    40.0    01.0    00.
00510 60.     01.     01.76   00.
>D500 513
00500 24 09 0A 54 65 73 74 49-4E 47 00 00 00 00 00 00   $..TestING......
00510 60 01 76 00                                        `.v.
>
```

F <RANGE> <LIST>

Fill - The specified range is filled with the values in the list. If the list is larger than the range, not all values will be used; if the range is larger, the list will be repeated as many times as necessary to fill it. All memory in <RANGE> must be valid for this command to work properly. If bad or non-existent memory is encountered, the error will be propagated into all succeeding locations.

```
>F400 L28 "Help" A D
>D400 L30
00400 48 65 6C 70 0A 0D 48 65-6C 70 0A 0D 48 65 6C 70   Help..Help..Help
00410 0A 0D 48 65 6C 70 0A 0D-48 65 6C 70 0A 0D 48 65   ..Help..Help..He
00420 6C 70 0A 0D 48 65 6C 70-FF 7F FF FF FF FF F7 FF   lp..Help......w.
>
```

G
G <ADDRESS> . . . <ADDRESS>

Go - Sets all registers from the register save area. Since this includes the Code Segment and Instruction Pointer, this implies a jump to the program under test.

This command allows setting up ten breakpoints. Attempting to set more than ten will cause a "BP Error". Breakpoints may be set only at an address containing the first byte of an 8086 opcode. A breakpoint is set by placing an interrupt opcode (0CCH) at the specified address. When that opcode is executed, all registers are saved and displayed, and all breakpoints locations are restored to their original value. If control is not returned to the Monitor by a breakpoint or interrupt, the breakpoints will not be cleared.

The user stack pointer must be valid and have 6 bytes available for this command to work. The jump to the user program is made with an IRET instruction with the user stack pointer set and user Flags, Code Segment register, and Instruction Pointer on the user stack. Thus if the user stack is not valid, the system will "crash".

The program below is an infinite loop of 16 INC AX instructions followed by a jump to its start. First breakpoints are used to execute a few instructions. Then a Go without breakpoints allows continuous, full-speed execution which is terminated by an interrupt from the keyboard - in this case, typing the space bar.

```
>F400 L10 40
>E410 EB EE
>D400 L12
00400 40 40 40 40 40 40 40 40-40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
00410 EB EE                                              kn
>
>G410

AX=0010  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0010   NV UP EI PL NZ AC PO NC
>G400 412

AX=0010  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV UP EI PL NZ AC PO NC
>G

AX=4590  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV UP EI PL NZ AC PE NC
>
```

I <HEX4>

Input - Inputs a byte from the specified port and displays it. A 16-bit port address is allowed.

M <RANGE> <ADDRESS>

Move - Moves the block of memory specified by <RANGE> to <ADDRESS>. Overlapping moves are always performed without loss of data, i.e., data is moved before it is overwritten. To do this, all moves from higher addresses to lower ones are done front-to-back, while moves from lower addresses to higher ones are done back-to-front.

```
>M400 L10 420
>D400 42F
00400 54 45 53 54 49 4E 47 FF-F7 FF FF F6 FF FF FE FF   TESTING.w..v..~.
00410 FF FF FE FF FF FF FF FF-FE FF FF FF FF FF FF FF   ..~.....~.......
00420 54 45 53 54 49 4E 47 FF-F7 FF FF F6 FF FF FE FF   TESTING.w..v..~.
>
>M404 40F 405
>D400 L10
00400 54 45 53 54 49 49 4E 47-FF F7 FF FF F6 FF FF FE   TESTIING.w..v..~
>
>M405 410 404
>D400L10
00400 54 45 53 54 49 4E 47 FF-F7 FF FF F6 FF FF FE FF   TESTING.w..v..~.
>
```

O <HEX4> <BYTE>

Output - <BYTE> is sent to the specified output port. A 16-bit port address is allowed.

R
R <REGISTER NAME>

Register - with no parameters, this command dumps the register save area.

Giving a register name as a parameter allows that register to be displayed and modified. The register name may be AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP, PC, or F (upper case only); anything else will result in an "BR Error". IP and PC both refer to the Instruction Pointer and F refers to the Flag register. For all exept the Flag register, the current 16-bit value will be printed in hex, then a colon will appear as a prompt for the replacement value. Typing <carriage return> leaves the register unchanged; otherwise type a <HEX4> to replace.

The Flag register uses a system of two-letter mnemonics for each flag, as shown below:

FLAG	CLEAR	SET
Overflow	NV No Overflow	OV Overflow
Direction	UP Up (Incrementing)	DN Down (Decrementing)
Interrupt	DI Disabled Interrupts	EI Enabled Interrupts
Sign	PL Plus	NG Negative
Zero	NZ Not Zero	ZR Zero
Auxillary Carry	NA No Auxillary Carry	AC Auxillary Carry
Parity	PO Parity Odd	PE Parity Even
Carry	NC No Carry	CY Carry

Whenever the Flag register is displayed, all flags are displayed in this order. When the F register is specified with the R command, the flags are displayed and then the Monitor waits for any replacements to be made. Any number of two-letter flag codes may be typed, and only those flags entered will be modified. If a flag has more than one code in the list, a "DF Error" (Double Flag) will result. If any code is not recognized, a "BF Error" (Bad Flag) will occur. In either case, those flags up to the error have been changed, and those after the error have not.

After reset, all registers are set to zero except the segment registers, which are set to 40H, and the Stack Pointer, which is set to 0C00H. Flags are all cleared except for interrupts. Execution on a Trace or Go command would thus begin at 400H, which is the first location after the interrupt table.

```
>R
AX=0000  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV UP EI PL NZ AC PE NC
>R AX
AX 0000
:106
>RCS
CS 0040
:
>RF
NV UP EI PL NZ AC PE NC -ZR DN
>R
AX=0106  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV DN EI PL ZR AC PE NC
>
```

S <RANGE> <LIST>

Search - The range is searched for a byte or string of bytes specified by <LIST>. For each occurence the first address of the match is displayed.

```
>S400 L8000 'Help'
00400
00406
0040C
00412
00418
0041E
00424
>D400 L28
00400 48 65 6C 70 0A 0D 48 65-6C 70 0A 0D 48 65 6C 70   Help..Help..Help
00410 0A 0D 48 65 6C 70 0A 0D-48 65 6C 70 0A 0D 48 65   ..Help..Help..He
00420 6C 70 0A 0D 48 65 6C 70                           lp..Help
>
```

T
T <HEX4>

Trace - The number of instructions specified (default 1) are traced. After each instruction, the complete contents of the registers and flags are displayed. (For the meaning of the flag symbols, see Register command.) Since this command uses the hardware trace mode of the 8086, even ROM may be traced.

```
>R
AX=0106  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0000   NV DN EI PL ZR AC PE NC
>T

AX=0107  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0001   NV DN EI PL NZ NA PO NC
>T

AX=0108  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0002   NV DN EI PL NZ NA PO NC
>T4

AX=0109  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0003   NV DN EI PL NZ NA PE NC

AX=010A  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0004   NV DN EI PL NZ NA PE NC

AX=010B  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0005   NV DN EI PL NZ NA PO NC

AX=010C  BX=0000  CX=0000  DX=0000  SP=0C00  BP=0000  SI=0000  DI=0000
DS=0040  ES=0040  SS=0040  CS=0040  IP=0006   NV DN EI PL NZ NA PE NC
>
```

8086 Monitor Assembly Listing

```
0000                     ; Seattle Computer Products 8086 Monitor version 1.4  2/18/80
0000                     ;   by Tim Paterson
0000                     ; This software is not copyrighted.
0000
0000
0000                     ;To select a disk boot, set one of the following equates
0000                     ;to 1, the rest to 0.
0000
0000                     CROMEMCO4FDC:   EQU     0          ;1 for 4FDC, 0 for others
0000                     NORTHSTARSD:    EQU     1          ;North Star single density?
0000                     TARBELL:        EQU     0          ;Tarbell (single or double)?
0000                     OTHER:          EQU     0          ;User-defined disk
0000
0000                     PUTBASE:EQU     100H
0000                     LOAD:   EQU     200H
0000                             ORG     7F0H
07F0                             PUT     PUTBASE+7F0H
07F0 EA 00 00 80 FF              JMP     0,0FF80H           ;Power-on jump to monitor
07F5
07F5                     ;Baud Rate Table. The 9513 divides 2MHz by these values.
07F5                     ;They are for 9600, 1200, 300, 150, 110 baud
07F5
07F5 0D 00 68 00 A0 01   BAUD:   DW      13,104,416,832,1144
     40 03 78 04
07FF
07FF                             ORG     100H               ;RAM area base address
0100
0100                     ;System Equates
0100
0100                     BASE:   EQU     0F0H               ;CPU Support base port address
0100                     STAT:   EQU     BASE+7             ;UART status port
0100                     DATA:   EQU     BASE+6             ;UART data port
0100                     DAV:    EQU     2                  ;UART data available bit
0100                     TBMT:   EQU     1                  ;UART transmitter ready bit
0100                     BUFLEN: EQU     80                 ;Maximum length of line input buffer
0100                     BPMAX:  EQU     10                 ;Maximum number of breakpoints
0100                     BPLEN:  EQU     BPMAX+BPMAX        ;Length of breakpoint table
0100                     REGTABLEN:EQU   14                 ;Number of registers
0100                     SEGDIF: EQU     800H               ;-0FF800H (ROM address)
0100                     PROMPT: EQU     ">"
0100                     CAN:    EQU     "@"
0100
0100                     ;RAM area.
0100
0100                     BRKCNT: DS      2                  ;Number of breakpoints
0102                     TCOUNT: DS      2                  ;Number of steps to trace
0104                     BPTAB:  DS      BPLEN              ;Breakpoint table
0118                     LINEBUF:DS      BUFLEN+1           ;Line input buffer
0169                             ALIGN
016A                             DS      50                 ;Working stack area
019C                     STACK:
019C
019C                     ;Register save area
019C
019C                     AXSAVE: DS      2
019E                     BXSAVE: DS      2
01A0                     CXSAVE: DS      2
01A2                     DXSAVE: DS      2
01A4                     SPSAVE: DS      2
01A6                     BPSAVE: DS      2
01A8                     SISAVE: DS      2
01AA                     DISAVE: DS      2
01AC                     DSSAVE: DS      2
01AE                     ESSAVE: DS      2
01B0                     RSTACK:         ;Stack set here so registers can be saved by pushing
01B0                     SSSAVE: DS      2
01B2                     CSSAVE: DS      2
01B4                     IPSAVE: DS      2
```

```
01B6                    FSAVE:  DS      2
01B8
01B8                    ;Start of Monitor code
01B8
01B8                            ORG     0
0000                            PUT     PUTBASE
0000
0000                    ;One-time initialization
0000
0000 FC                         UP
0001 33 C0                      XOR     AX,AX
0003 8E D0                      MOV     SS,AX
0005 8E D8                      MOV     DS,AX
0007 8E C0                      MOV     ES,AX
0009 BF 9C 01                   MOV     DI,AXSAVE
000C B9 0E 00                   MOV     CX,14
000F F3                         REP
0010 AB                         STOW                    ;Set register images to zero
0011 80 0E B7 01 02             OR      B,[FSAVE+1],2   ;Enable interrupts
0016 B1 04                      MOV     CL,4
0018 B0 40                      MOV     AL,40H
001A BF AC 01                   MOV     DI,DSSAVE
001D F3                         REP
001E AB                         STOW                    ;Set segment reg. images to 40H
001F C6 06 A5 01 0C             MOV B   [SPSAVE+1],0CH  ;Set user stack to 400H+0C00H
0024 BC 9C 01                   MOV     SP,STACK
0027                    ;Prepare 9513
0027 B0 17                      MOV     AL,17H
0029 E6 F5                      OUT     BASE+5          ;Select Master Mode register
002B B0 F3                      MOV     AL,0F3H
002D E6 F4                      OUT     BASE+4          ;Low byte of Master Mode
002F B8 84 05                   MOV     AX,584H         ;Output 84H to BASE+4
0032 E7 F4                      OUTW    BASE+4          ;and 05H to BASE+5
0034                    ;Master Mode now set to 84F3H:
0034                    ;       Scaler set to BCD division
0034                    ;       Enable data pointer increment
0034                    ;       8-bit data bus
0034                    ;       FOUT=100Hz, dividing F5 by 4 (F5=4MHz/10000)
0034                    ;       Both alarm comparators disabled
0034                    ;       Time-of-day enabled
0034                    ;Counter 5 selected
0034
0034                    ;Initialize loop. Ports BASE through BASE+7 are initialized
0034                    ;from table. Each table entry has number of bytes followed by
0034                    ;data.
0034
0034 BE 33 07                   MOV     SI,INITTABLE    ;Initialization table
0037 BA F0 00                   MOV     DX,BASE         ;DX has (variable) port no.
003A                    INITPORT:
003A 2E                         SEG     CS
003B AC                         LODB                    ;Get byte count
003C 8A C8                      MOV     CL,AL
003E E3 05                      JCXZ    NEXTPORT        ;No init. for some ports
0040                    INITBYTE:
0040 2E                         SEG     CS
0041 AC                         LODB                    ;Get init. data
0042 EE                         OUT     DX              ;Send to port
0043 E2 FB                      LOOP    INITBYTE        ;As many bytes as required
0045                    NEXTPORT:
0045 42                         INC     DX              ;Prepare for next port
0046 80 FA F8                   CMP     DL,BASE+8       ;Check against limit
0049 75 EF                      JNZ     INITPORT
004B
004B                    ;Initialization complete except for determining baud rate.
004B                    ;Both 8259As are ready to accept interrupts, the 9513 is
004B                    ;providing 19.2k baud X 16 to the 8251A which is set for
004B                    ;16X clock and one stop bit.
004B
004B E8 19 00                   CALL    CHECKB          ;Check for correct baud rate
004E                    ;CHECKB does not return if baud rate is correct
```

```
004E
004E                        ;Intial baud rate (19.2k) was wrong, so run auto-baud routine
004E
004E                        INITBAUD:
004E BE F5 07                        MOV     SI,BAUD
0051                        ;First set up 9513 for slower baud rates (<=9600).
0051                        ;Counter 5 mode register has already been selected.
0051 B8 23 E8                        MOV     AX,0E823H           ;Output 23H to BASE+4
0054 E7 F4                           OUTW    BASE+4              ;and 0E8H to BASE+5
0056                        ;23H to BASE+4 sets lower half of Counter 5 mode register.
0056                        ;Reload from Load, count down repetively in binary,
0056                        ;toggle output.
0056                        ;0E8H to BASE+5 disables data pointer sequencing
0056
0056 B0 0D                           MOV     AL,0DH
0058 E6 F5                           OUT     BASE+5              ;Select Counter 5 load reg.
005A                        INITB:
005A 2E                              SEG     CS
005B AD                              LODW                        ;Get divisor
005C E6 F4                           OUT     BASE+4              ;Output low byte
005E 8A C4                           MOV     AL,AH
0060 E6 F4                           OUT     BASE+4              ;Output high byte
0062 E8 02 00                        CALL    CHECKB              ;Check if baud rate correct
0065 EB F3                           JP      INITB               ;Try next rate if not
0067                        CHECKB:
0067 E8 98 00                        CALL    IN                  ;First byte could be messed up
006A E8 95 00                        CALL    IN                  ;Get carriage return
006D 3C 0D                           CMP     AL,13               ;Correct?
006F 74 01                           JZ      MONITOR             ;Don't return if correct
0071 C3                              RET                         ;Didn't get it yet
0072
0072                        ;Initialization complete, including baud rate.
0072
0072                        MONITOR:
0072                        ; Do auto boot if sense switch 0 is on.
0072 BF 18 01                        MOV     DI,LINEBUF
0075 C6 05 0D                        MOV     B,[DI],13           ;No breakpoints after boot
0078 E4 FF                           IN      BASE+0FH            ;Sense switch port
007A A8 01                           TEST    AL,1
007C 74 03                           JZ      DOMON
007E E9 F5 06                        JMP     BOOT
0081                        DOMON:
0081 BE 51 07                        MOV     SI,HEADER
0084 E8 8B 00                        CALL    PRINTMES
0087                        COMMAND:
0087                        ;Re-establish initial conditions
0087 FC                              UP
0088 33 C0                           XOR     AX,AX
008A 8E D8                           MOV     DS,AX
008C 8E C0                           MOV     ES,AX
008E BC 9C 01                        MOV     SP,STACK
0091 C7 06 64 00 BB 06               MOV     [64H],INT           ;Set UART interrupt vector
0097 8C 0E 66 00                     MOV     [66H],CS
009B B0 3E                           MOV     AL,PROMPT
009D E8 C8 00                        CALL    OUT
00A0 E8 1E 00                        CALL    INBUF               ;Get command line
00A3                        ;From now and throughout command line processing, DI points
00A3                        ;to next character in command line to be processed.
00A3 E8 7F 00                        CALL    SCANB               ;Scan off leading blanks
00A6 74 DF                           JZ      COMMAND             ;Null command?
00A8 8A 05                           MOV     AL,[DI]             ;AL=first non-blank character
00AA                        ;Prepare command letter for table lookup
00AA 2C 42                           SUB     AL,"B"              ;Low end range check
00AC 72 10                           JC      ERR1
00AE 3C 13                           CMP     AL,"T"+1-"B"        ;Upper end range check
00B0 73 0C                           JNC     ERR1
00B2 47                              INC     DI
00B3 D0 E0                           SHL     AL                  ;Times two
00B5 98                              CBW                         ;Now a 16-bit quantity
00B6 93                              XCHG    BX,AX               ;In BX we can address with it
```

```
00B7 2E                         SEG     CS
00B8 FF 97 7D 01                CALL    [BX+COMTAB]     ;Execute command
00BC EB C9                      JP      COMMAND         ;Get next command
00BE E9 A8 02           ERR1:   JMP     ERROR
00C1
00C1                    ;Get input line
00C1
00C1                    INBUF:
00C1 BF 18 01                   MOV     DI,LINEBUF      ;Next empty buffer location
00C4 33 C9                      XOR     CX,CX           ;Character count
00C6                    GETCH:
00C6 E8 39 00                   CALL    IN              ;Get input character
00C9 3C 20                      CMP     AL,20H          ;Check for control characters
00CB 72 1B                      JC      CONTROL
00CD 3C 7F                      CMP     AL,7FH          ;RUBOUT is a backspace
00CF 74 0E                      JZ      BACKSP
00D1 E8 94 00                   CALL    OUT             ;Echo character
00D4 3C 40                      CMP     AL,CAN          ;Cancel line?
00D6 74 25                      JZ      KILL
00D8 AA                         STOB                    ;Put in input buffer
00D9 41                         INC     CX              ;Bump character count
00DA 83 F9 50                   CMP     CX,BUFLEN       ;Buffer full?
00DD 76 E7                      JBE     GETCH           ;Drop in to backspace if full
00DF                    BACKSP:
00DF E3 E5                      JCXZ    GETCH           ;Can't backspace over nothing
00E1 4F                         DEC     DI              ;Drop pointer
00E2 49                         DEC     CX              ;and character count
00E3 E8 29 00                   CALL    BACKUP          ;Send physical backspace
00E6 EB DE                      JP      GETCH           ;Get next char.
00E8                    CONTROL:
00E8 3C 08                      CMP     AL,8            ;Check for backspace
00EA 74 F3                      JZ      BACKSP
00EC 3C 0D                      CMP     AL,13           ;Check for carriage return
00EE 75 D6                      JNZ     GETCH           ;Ignore all other control char.
00F0 AA                         STOB                    ;Put the car. ret. in buffer
00F1 BF 18 01                   MOV     DI,LINEBUF      ;Set up DI for command processing
00F4
00F4                    ;Output CR/LF sequence
00F4
00F4                    CRLF:
00F4 B0 0D                      MOV     AL,13
00F6 E8 6F 00                   CALL    OUT
00F9 B0 0A                      MOV     AL,10
00FB EB 6B                      JP      OUT
00FD
00FD                    ;Cancel input line
00FD
00FD                    KILL:
00FD E8 F4 FF                   CALL    CRLF
0100 EB 85                      JP      COMMAND
0102
0102                    ;Character input routine
0102
0102                    IN:
0102 FA                         DI                      ;Poll, don't interrupt
0103 E4 F7                      INB     STAT
0105 A8 02                      TEST    AL,DAV
0107 74 F9                      JZ      IN              ;Loop until ready
0109 E4 F6                      INB     DATA
010B 24 7F                      AND     AL,7FH          ;Only 7 bits
010D FB                         EI                      ;Interrupts OK now
010E C3                         RET
010F
010F                    ;Physical backspace - blank, backspace, blank
010F
010F                    BACKUP:
010F BE 73 07                   MOV     SI,BACMES
0112
0112                    ;Print ASCII message. Last char has bit 7 set
0112
```

```
0112                        PRINTMES:
0112 2E                             SEG     CS
0113 AC                             LODB                    ;Get char to print
0114 E8 51 00                       CALL    OUT
0117 D0 E0                          SHL     AL              ;High bit set?
0119 73 F7                          JNC     PRINTMES
011B C3                             RET
011C
011C                        ;Scan for parameters of a command
011C
011C                        SCANP:
011C E8 06 00                       CALL    SCANB           ;Get first non-blank
011F 82 3D 2C                       CMP     B,[DI],","      ;One comma between params OK
0122 75 0A                          JNE     EOLCHK          ;If not comma, we found param
0124 47                             INC     DI              ;Skip over comma
0125
0125                        ;Scan command line for next non-blank character
0125
0125                        SCANB:
0125 B0 20                          MOV     AL," "
0127 51                             PUSH    CX              ;Don't disturb CX
0128 B1 FF                          MOV     CL,-1           ;but scan as many as necessary
012A F3                             REPE
012B AE                             SCAB
012C 4F                             DEC     DI              ;Back up to first non-blank
012D 59                             POP     CX
012E                        EOLCHK:
012E 82 3D 0D                       CMP     B,[DI],13
0131 C3                             RET
0132
0132                        ;Print the 5-digit hex address of SI and DS
0132
0132                        OUTSI:
0132 8C DA                          MOV     DX,DS           ;Put DS where we can work with it
0134 B4 00                          MOV     AH,0            ;Will become high bits of DS
0136 E8 78 00                       CALL    SHIFT4          ;Shift DS four bits
0139 03 D6                          ADD     DX,SI           ;Compute absolute address
013B EB 09                          JP      OUTADD          ;Finish below
013D
013D                        ;Print 5-digit hex address of DI and ES
013D                        ;Same as OUTSI above
013D
013D                        OUTDI:
013D 8C C2                          MOV     DX,ES
013F B4 00                          MOV     AH,0
0141 E8 6D 00                       CALL    SHIFT4
0144 03 D7                          ADD     DX,DI
0146                        ;Finish OUTSI here too
0146                        OUTADD:
0146 82 D4 00                       ADC     AH,0            ;Add in carry to high bits
0149 E8 12 00                       CALL    HIDIG           ;Output hex value in AH
014C
014C                        ;Print out 16-bit value in DX in hex
014C
014C                        OUT16:
014C 8A C6                          MOV     AL,DH           ;High-order byte first
014E E8 02 00                       CALL    HEX
0151 8A C2                          MOV     AL,DL           ;Then low-order byte
0153
0153                        ;Output byte in AL as two hex digits
0153
0153                        HEX:
0153 8A E0                          MOV     AH,AL           ;Save for second digit
0155                        ;Shift high digit into low 4 bits
0155 51                             PUSH    CX
0156 B1 04                          MOV     CL,4
0158 D2 E8                          SHR     AL,CL
015A 59                             POP     CX
015B
015B E8 02 00                       CALL    DIGIT           ;Output first digit
```

```
015E                    HIDIG:
015E 8A C4                      MOV     AL,AH           ;Now do digit saved in AH
0160                    DIGIT:
0160 24 0F                      AND     AL,0FH          ;Mask to 4 bits
0162                    ;Trick 6-byte hex conversion works on 8086 too.
0162 04 90                      ADD     AL,90H
0164 27                         DAA
0165 14 40                      ADC     AL,40H
0167 27                         DAA
0168
0168                    ;Console output of character in AL
0168
0168                    OUT:
0168 50                         PUSH    AX              ;Character to output on stack
0169                    OUT1:
0169 E4 F7                      INB     STAT
016B 24 01                      AND     AL,TBMT
016D 74 FA                      JZ      OUT1            ;Wait until ready
016F 58                         POP     AX
0170 E6 F6                      OUTB    DATA
0172 C3                         RET
0173
0173                    ;Output one space
0173
0173                    BLANK:
0173 B0 20                      MOV     AL," "
0175 EB F1                      JP      OUT
0177
0177                    ;Output the number of blanks in CX
0177
0177                    TAB:
0177 E8 F9 FF                   CALL    BLANK
017A E2 FB                      LOOP    TAB
017C C3                         RET
017D
017D                    ;Command Table. Command letter indexes into table to get
017D                    ;address of command. PERR prints error for no such command.
017D
017D                    COMTAB:
017D 76 07                      DW      BOOT            ;B
017F 68 03                      DW      PERR            ;C
0181 0D 02                      DW      DUMP            ;D
0183 88 03                      DW      ENTER           ;E
0185 97 02                      DW      FILL            ;F
0187 6A 06                      DW      GO              ;G
0189 68 03                      DW      PERR            ;H
018B 4C 06                      DW      INPUT           ;I
018D 68 03                      DW      PERR            ;J
018F 68 03                      DW      PERR            ;K
0191 68 03                      DW      PERR            ;L
0193 6A 02                      DW      MOVE            ;M
0195 68 03                      DW      PERR            ;N
0197 59 06                      DW      OUTPUT          ;O
0199 68 03                      DW      PERR            ;P
019B 68 03                      DW      PERR            ;Q
019D 2F 04                      DW      REG             ;R
019F BA 02                      DW      SEARCH          ;S
01A1 6A 05                      DW      TRACE           ;T
01A3
01A3                    ;Given 20-bit address in AH:DX, breaks it down to a segment
01A3                    ;number in AX and a displacement in DX. Displacement is
01A3                    ;always zero except for least significant 4 bits.
01A3
01A3                    GETSEG:
01A3 8A C2                      MOV     AL,DL
01A5 24 0F                      AND     AL,0FH          ;AL has least significant 4 bits
01A7 E8 07 00                   CALL    SHIFT4          ;4-bit left shift of AH:DX
01AA 8A D0                      MOV     DL,AL           ;Restore lowest 4 bits
01AC 8A C6                      MOV     AL,DH           ;Low byte of segment number
01AE 32 F6                      XOR     DH,DH           ;Zero high byte of displacement
```

```
01B0 C3                         RET
01B1
01B1                    ;Shift AH:DX left 4 bits
01B1
01B1                    SHIFT4:
01B1 D1 E2                      SHL     DX
01B3 D0 D4                      RCL     AH      ;1
01B5 D1 E2                      SHL     DX
01B7 D0 D4                      RCL     AH      ;2
01B9 D1 E2                      SHL     DX
01BB D0 D4                      RCL     AH      ;3
01BD D1 E2                      SHL     DX
01BF D0 D4                      RCL     AH      ;4
01C1 C3                 RET2:   RET
01C2
01C2                    ;RANGE - Looks for parameters defining an address range.
01C2                    ;The first parameter is a hex number of 5 or less digits
01C2                    ;which specifies the starting address. The second parameter
01C2                    ;may specify the ending address, or it may be preceded by
01C2                    ;"L" and specify a length (4 digits max), or it may be
01C2                    ;omitted and a length of 128 bytes is assumed. Returns with
01C2                    ;segment no. in AX and displacement (0-F) in DX.
01C2
01C2                    RANGE:
01C2 B9 05 00                   MOV     CX,5            ;5 digits max
01C5 E8 22 01                   CALL    GETHEX          ;Get hex number
01C8 50                         PUSH    AX              ;Save high 4 bits
01C9 52                         PUSH    DX              ;Save low 16 bits
01CA E8 4F FF                   CALL    SCANP           ;Get to next parameter
01CD 82 3D 4C                   CMP     B,[DI],"L"      ;Length indicator?
01D0 74 1C                      JE      GETLEN
01D2 BA 80 00                   MOV     DX,128          ;Default length
01D5 E8 30 01                   CALL    HEXIN           ;Second parameter present?
01D8 72 1B                      JC      RNGRET          ;If not, use default
01DA B9 05 00                   MOV     CX,5            ;5 hex digits
01DD E8 0A 01                   CALL    GETHEX          ;Get ending address
01E0 8B CA                      MOV     CX,DX           ;Low 16 bits of ending addr.
01E2 5A                         POP     DX              ;Low 16 bits of starting addr.
01E3 5B                         POP     BX              ;BH=hi 4 bits of start addr.
01E4 2B CA                      SUB     CX,DX           ;Compute range
01E6 1A E7                      SBB     AH,BH           ;Finish 20-bit subtract
01E8 75 1D                      JNZ     RNGERR          ;Range must be less than 64K
01EA 93                         XCHG    AX,BX           ;AH=starting, BH=ending hi 4 bits
01EB 41                         INC     CX              ;Range must include ending location
01EC EB 0B                      JP      RNGCHK          ;Finish range testing and return
01EE                    GETLEN:
01EE 47                         INC     DI              ;Skip over "L" to length
01EF B9 04 00                   MOV     CX,4            ;Length may have 4 digits
01F2 E8 F5 00                   CALL    GETHEX          ;Get the range
01F5                    RNGRET:
01F5 8B CA                      MOV     CX,DX           ;Length
01F7 5A                         POP     DX              ;Low 16 bits of starting addr.
01F8 58                         POP     AX              ;AH=hi 4 bits of starting addr.
01F9
01F9                    ;RNGCHK verifies that the range lies entirely within one segment.
01F9                    ;CX=0 means count=10000H. Range is within one segment only if
01F9                    ;adding the low 4 bits of the starting address to the count is
01F9                    ;<=10000H, because segments can start only on 16-byte boundaries.
01F9
01F9                    RNGCHK:
01F9 8B DA                      MOV     BX,DX           ;Low 16 bits of start addr.
01FB 81 E3 0F 00                AND     BX,0FH          ;Low 4 bits of starting addr.
01FF E3 04                      JCXZ    MAXRNG          ;If count=10000H then BX must be 0
0201 03 D9                      ADD     BX,CX           ;Must be <=10000H
0203 73 9E                      JNC     GETSEG          ;OK if strictly <
0205                    MAXRNG:
0205                    ;If here because of JCXZ MAXRNG, we are testing if low 4 bits
0205                    ;(in BX) are zero. If we dropped straight in, we are testing
0205                    ;for BX+CX=10000H (=0). Either way, zero flag set means
0205                    ;withing range.
```

```
0205 74 9C                      JZ      GETSEG
0207                    RNGERR:
0207 B8 52 47                   MOV     AX,4700H+"R"    ;RG ERROR
020A E9 1F 03                   JMP     ERR
020D
020D                    ;Dump an area of memory in both hex and ASCII
020D
020D                    DUMP:
020D E8 B2 FF                   CALL    RANGE           ;Get range to dump
0210 50                         PUSH    AX              ;Save segment
0211 E8 4E 01                   CALL    GETEOL          ;Check for errors
0214 1F                         POP     DS              ;Set segment
0215 8B F2                      MOV     SI,DX           ;SI has displacement in segment
0217                    ROW:
0217 E8 18 FF                   CALL    OUTSI           ;Print address at start of line
021A 56                         PUSH    SI              ;Save address for ASCII dump
021B                    BYTE:
021B E8 55 FF                   CALL    BLANK           ;Space between bytes
021E                    BYTE1:
021E AC                         LODB                    ;Get byte to dump
021F E8 31 FF                   CALL    HEX             ;and display it
0222 5A                         POP     DX              ;DX has start addr. for ASCII dump
0223 49                         DEC     CX              ;Drop loop count
0224 74 17                      JZ      ASCII           ;If through do ASCII dump
0226 8B C6                      MOV     AX,SI
0228 A8 0F                      TEST    AL,0FH          ;On 16-byte boundary?
022A 74 0C                      JZ      ENDROW
022C 52                         PUSH    DX              ;Didn't need ASCII addr. yet
022D A8 07                      TEST    AL,7            ;On 8-byte boundary?
022F 75 EA                      JNZ     BYTE
0231 B0 2D                      MOV     AL,"-"          ;Mark every 8 bytes
0233 E8 32 FF                   CALL    OUT
0236 EB E6                      JP      BYTE1
0238                    ENDROW:
0238 E8 02 00                   CALL    ASCII           ;Show it in ASCII
023B EB DA                      JP      ROW             ;Loop until count is zero
023D                    ASCII:
023D 51                         PUSH    CX              ;Save byte count
023E 8B C6                      MOV     AX,SI           ;Current dump address
0240 8B F2                      MOV     SI,DX           ;ASCII dump address
0242 2B C2                      SUB     AX,DX           ;AX=length of ASCII dump
0244                    ;Compute tab length. ASCII dump always appears on right side
0244                    ;screen regardless of how many bytes were dumped. Figure 3
0244                    ;characters for each byte dumped and subtract from 51, which
0244                    ;allows a minimum of 3 blanks after the last byte dumped.
0244 8B D8                      MOV     BX,AX
0246 D1 E0                      SHL     AX              ;Length times 2
0248 03 C3                      ADD     AX,BX           ;Length times 3
024A B9 33 00                   MOV     CX,51
024D 2B C8                      SUB     CX,AX           ;Amount to tab in CX
024F E8 25 FF                   CALL    TAB
0252 8B CB                      MOV     CX,BX           ;ASCII dump length back in CX
0254                    ASCDMP:
0254 AC                         LODB                    ;Get ASCII byte to dump
0255 24 7F                      AND     AL,7FH          ;ASCII uses 7 bits
0257 3C 7F                      CMP     AL,7FH          ;Don't try to print RUBOUT
0259 74 04                      JZ      NOPRT
025B 3C 20                      CMP     AL," "          ;Check for control characters
025D 73 02                      JNC     PRIN
025F                    NOPRT:
025F B0 2E                      MOV     AL,"."          ;If unprintable character
0261                    PRIN:
0261 E8 04 FF                   CALL    OUT             ;Print ASCII character
0264 E2 EE                      LOOP    ASCDMP          ;CX times
0266 59                         POP     CX              ;Restore overall dump length
0267 E9 8A FE                   JMP     CRLF            ;Print CR/LF and return
026A
026A                    ;Block move one area of memory to another. Overlapping moves
026A                    ;are performed correctly, i.e., so that a source byte is not
026A                    ;overwritten until after it has been moved.
```

```
026A
026A                    MOVE:
026A E8 55 FF                   CALL    RANGE           ;Get range of source area
026D 51                         PUSH    CX              ;Save length
026E 50                         PUSH    AX              ;Save segment
026F 8B F2                      MOV     SI,DX           ;Set source displacement
0271 B9 05 00                   MOV     CX,5            ;Allow 5 digits
0274 E8 73 00                   CALL    GETHEX          ;in destination address
0277 E8 E8 00                   CALL    GETEOL          ;Check for errors
027A E8 26 FF                   CALL    GETSEG          ;Convert dest. to seg/disp
027D 8B FA                      MOV     DI,DX           ;Set dest. displacement
027F 5B                         POP     BX              ;Source segment
0280 8E DB                      MCV     DS,BX
0282 8E C0                      MOV     ES,AX           ;Destination segment
0284 59                         POP     CX              ;Length
0285 3B FE                      CMP     DI,SI           ;Check direction of move
0287 1B C3                      SBB     AX,BX           ;Extend the CMP to 32 bits
0289 72 07                      JB      COPYLIST        ;Move forward into lower mem.
028B                    ;Otherwise, move backward. Figure end of source and destination
028B                    ;areas and flip direction flag.
028B 49                         DEC     CX
028C 03 F1                      ADD     SI,CX           ;End of source area
028E 03 F9                      ADD     DI,CX           ;End of destination area
0290 FD                         DOWN                    ;Reverse direction
0291 41                         INC     CX
0292                    COPYLIST:
0292 A4                         MOVB    ;Do at least 1 - Range is 1-10000H not 0-FFFFH
0293 49                         DEC     CX
0294 F3                         REP
0295 A4                         MOVB                    ;Block move
0296 C3                         RET
0297
0297                    ;Fill an area of memory with a list values. If the list
0297                    ;is bigger than the area, don't use the whole list. If the
0297                    ;list is smaller, repeat it as many times as necessary.
0297
0297                    FILL:
0297 E8 28 FF                   CALL    RANGE           ;Get range to fill
029A 51                         PUSH    CX              ;Save length
029B 50                         PUSH    AX              ;Save segment number
029C 52                         PUSH    DX              ;Save displacement
029D E8 B4 00                   CALL    LIST            ;Get list of values to fill with
02A0 5F                         POP     DI              ;Displacement in segment
02A1 07                         POP     ES              ;Segment
02A2 59                         POP     CX              ;Length
02A3 3B D9                      CMP     BX,CX           ;BX is length of fill list
02A5 BE 18 01                   MOV     SI,LINEBUF      ;List is in line buffer
02A8 E3 02                      JCXZ    BIGRNG
02AA 73 E6                      JAE     COPYLIST        ;If list is big, copy part of it
02AC                    BIGRNG:
02AC 2B CB                      SUB     CX,BX           ;How much bigger is area than list?
02AE 87 D9                      XCHG    CX,BX           ;CX=length of list
02B0 57                         PUSH    DI              ;Save starting addr. of area
02B1 F3                         REP
02B2 A4                         MOVB                    ;Move list into area
02B3 5E                         POP     SI
02B4                    ;The list has been copied into the beginning of the
02B4                    ;specified area of memory. SI is the first address
02B4                    ;of that area, DI is the end of the copy of the list
02B4                    ;plus one, which is where the list will begin to repeat.
02B4                    ;All we need to do now is copy [SI] to [DI] until the
02B4                    ;end of the memory area is reached. This will cause the
02B4                    ;list to repeat as many times as necessary.
02B4 8B CB                      MOV     CX,BX           ;Length of area minus list
02B6 06                         PUSH    ES              ;Different index register
02B7 1F                         POP     DS              ;requires different segment reg.
02B8 EB D8                      JP      COPYLIST        ;Do the block move
02BA
02BA                    ;Search a specified area of memory for given list of bytes.
02BA                    ;Print address of first byte of each match.
```

```
02BA
02BA                  SEARCH:
02BA E8 05 FF                 CALL    RANGE           ;Get area to be searched
02BD 51                       PUSH    CX              ;Save count
02BE 50                       PUSH    AX              ;Save segment number
02BF 52                       PUSH    DX              ;Save displacement
02C0 E8 91 00                 CALL    LIST            ;Get search list
02C3 4B                       DEC     BX              ;No. of bytes in list-1
02C4 5F                       POP     DI              ;Displacement within segment
02C5 07                       POP     ES              ;Segment
02C6 59                       POP     CX              ;Length to be searched
02C7 2B CB                    SUB     CX,BX           ;  minus length of list
02C9                  SCAN:
02C9 BE 18 01                 MOV     SI,LINEBUF      ;List kept in line buffer
02CC AC                       LODB                    ;Bring first byte into AL
02CD                  DOSCAN:
02CD AE                       SCAB                    ;Search for first byte
02CE E0 FD                    LOOPNE  DOSCAN          ;Do at least once by using LOOP
02D0 75 4A                    JNZ     RET             ;Exit if not found
02D2 53                       PUSH    BX              ;Length of list minus 1
02D3 87 CB                    XCHG    BX,CX
02D5 57                       PUSH    DI              ;Will resume search here
02D6 F3                       REPE
02D7 A6                       CMPB                    ;Compare rest of string
02D8 8B CB                    MOV     CX,BX           ;Area length back in CX
02DA 5F                       POP     DI              ;Next search location
02DB 5B                       POP     BX              ;Restore list length
02DC 75 08                    JNZ     TEST            ;Continue search if no match
02DE 4F                       DEC     DI              ;Match address
02DF E8 5B FE                 CALL    OUTDI           ;Print it
02E2 47                       INC     DI              ;Restore search address
02E3 E8 0E FE                 CALL    CRLF
02E6                  TEST:
02E6 E3 34                    JCXZ    RET
02E8 EB DF                    JP      SCAN            ;Look for next occurrence
02EA
02EA                  ;Get the next parameter, which must be a hex number.
02EA                  ;CX is maximum number of digits the number may have.
02EA
02EA                  GETHEX:
02EA E8 2F FE                 CALL    SCANP           ;Scan to next parameter
02ED                  GETHEX1:
02ED 33 D2                    XOR     DX,DX           ;Initialize the number
02EF 8A E6                    MOV     AH,DH
02F1 E8 14 00                 CALL    HEXIN           ;Get a hex digit
02F4 72 73                    JC      ERROR           ;Must be one valid digit
02F6 8A D0                    MOV     DL,AL           ;First 4 bits in position
02F8                  GETLP:
02F8 47                       INC     DI              ;Next char in buffer
02F9 49                       DEC     CX              ;Digit count
02FA E8 0B 00                 CALL    HEXIN           ;Get another hex digit?
02FD 72 1D                    JC      RET             ;All done if no more digits
02FF E3 68                    JCXZ    ERROR           ;Too many digits?
0301 E8 AD FE                 CALL    SHIFT4          ;Multiply by 16
0304 0A D0                    OR      DL,AL           ;and combine new digit
0306 EB F0                    JP      GETLP           ;Get more digits
0308
0308                  ;Check if next character in the input buffer is a hex digit
0308                  ;and convert it to binary if it is. Carry set if not.
0308
0308                  HEXIN:
0308 8A 05                    MOV     AL,[DI]
030A
030A                  ;Check if AL has a hex digit and convert it to binary if it
030A                  ;is. Carry set if not.
030A
030A                  HEXCHK:
030A 2C 30                    SUB     AL,"0"          ;Kill ASCII numeric bias
030C 72 0E                    JC      RET
030E 3C 0A                    CMP     AL,10
```

```
0310 F5                         CMC
0311 73 09                      JNC     RET             ;OK if 0-9
0313 2C 07                      SUB     AL,7            ;Kill A-F bias
0315 3C 0A                      CMP     AL,10
0317 72 03                      JC      RET
0319 3C 10                      CMP     AL,16
031B F5                         CMC
031C C3                 RET:    RET
031D
031D                    ;Process one parameter when a list of bytes is
031D                    ;required. Carry set if parameter bad. Called by LIST
031D
031D                    LISTITEM:
031D E8 FC FD                   CALL    SCANP           ;Scan to parameter
0320 E8 E5 FF                   CALL    HEXIN           ;Is it in hex?
0323 72 0B                      JC      STRINGCHK       ;If not, could be a string
0325 B9 02 00                   MOV     CX,2            ;Only 2 hex digits for bytes
0328 E8 BF FF                   CALL    GETHEX          ;Get the byte value
032B 88 17                      MOV     [BX],DL         ;Add to list
032D 43                         INC     BX
032E F8                 GRET:   CLC                     ;Parameter was OK
032F C3                         RET
0330                    STRINGCHK:
0330 8A 05                      MOV     AL,[DI]         ;Get first character of param
0332 3C 27                      CMP     AL,"'"          ;String?
0334 74 06                      JZ      STRING
0336 3C 22                      CMP     AL,'"'          ;Either quote is all right
0338 74 02                      JZ      STRING
033A F9                         STC                     ;Not string, not hex - bad
033B C3                         RET
033C                    STRING:
033C 8A E0                      MOV     AH,AL           ;Save for closing quote
033E 47                         INC     DI
033F                    STRNGLP:
033F 8A 05                      MOV     AL,[DI]         ;Next char of string
0341 47                         INC     DI
0342 3C 0D                      CMP     AL,13           ;Check for end of line
0344 74 23                      JZ      ERROR           ;Must find a close quote
0346 3A C4                      CMP     AL,AH           ;Check for close quote
0348 75 05                      JNZ     STOSTRG         ;Add new character to list
034A 3A 25                      CMP     AH,[DI]         ;Two quotes in a row?
034C 75 E0                      JNZ     GRET            ;If not, we're done
034E 47                         INC     DI              ;Yes - skip second one
034F                    STOSTRG:
034F 88 07                      MOV     [BX],AL         ;Put new char in list
0351 43                         INC     BX
0352 EB EB                      JP      STRNGLP         ;Get more characters
0354
0354                    ;Get a byte list for ENTER, FILL or SEARCH. Accepts any number
0354                    ;of 2-digit hex values or character strings in either single
0354                    ;(') or double (") quotes.
0354
0354                    LIST:
0354 BB 18 01                   MOV     BX,LINEBUF      ;Put byte list in the line buffer
0357                    LISTLP:
0357 E8 C3 FF                   CALL    LISTITEM        ;Process a parameter
035A 73 FB                      JNC     LISTLP          ;If OK, try for more
035C 81 EB 18 01                SUB     BX,LINEBUF      ;BX now has no. of bytes in list
0360 74 07                      JZ      ERROR           ;List must not be empty
0362
0362                    ;Make sure there is nothing more on the line except for
0362                    ;blanks and carriage return. If there is, it is an
0362                    ;unrecognized parameter and an error.
0362
0362                    GETEOL:
0362 E8 C0 FD                   CALL    SCANB           ;Skip blanks
0365 75 02                      JNZ     ERROR           ;Better be a RETURN
0367 C3                         RET
0368
0368                    ;Command error. DI has been incremented beyond the
```

```
0368                    ;command letter so it must decremented for the
0368                    ;error pointer to work.
0368
0368                    PERR:
0368 4F                         DEC     DI
0369
0369                    ;Syntax error. DI points to character in the input buffer
0369                    ;which caused error. By subtracting from start of buffer,
0369                    ;we will know how far to tab over to appear directly below
0369                    ;it on the terminal. Then print "^ Error".
0369
0369                    ERROR:
0369 81 EF 17 01                SUB     DI,LINEBUF-1    ;How many char processed so far?
036D 8B CF                      MOV     CX,DI           ;Parameter for TAB in CX
036F E8 05 FE                   CALL    TAB             ;Directly below bad char
0372 BE 6A 07                   MOV     SI,SYNERR       ;Error message
0375
0375                    ;Print error message and abort to command level
0375
0375                    PRINT:
0375 E8 9A FD                   CALL    PRINTMES
0378 E9 0C FD                   JMP     COMMAND
037B
037B                    ;Short form of ENTER command. A list of values from the
037B                    ;command line are put into memory without using normal
037B                    ;ENTER mode.
037B
037B                    GETLIST:
037B E8 D6 FF                   CALL    LIST            ;Get the bytes to enter
037E 5F                         POP     DI              ;Displacement within segment
037F 07                         POP     ES              ;Segment to enter into
0380 BE 18 01                   MOV     SI,LINEBUF      ;List of bytes is in line buffer
0383 8B CB                      MOV     CX,BX           ;Count of bytes
0385 F3                         REP
0386 A4                         MOVB                    ;Enter that byte list
0387 C3                         RET
0388
0388                    ;Enter values into memory at a specified address. If the
0388                    ;line contains nothing but the address we go into "enter
0388                    ;mode", where the address and its current value are printed
0388                    ;and the user may change it if desired. To change, type in
0388                    ;new value in hex. Backspace works to correct errors. If
0388                    ;an illegal hex digit or too many digits are typed, the
0388                    ;bell is sounded but it is otherwise ignored. To go to the
0388                    ;next byte (with or without change), hit space bar. To
0388                    ;back up to a previous address, type "-". On
0388                    ;every 8-byte boundary a new line is started and the address
0388                    ;is printed. To terminate command, type carriage return.
0388                    ;   Alternatively, the list of bytes to be entered may be
0388                    ;included on the original command line immediately following
0388                    ;the address. This is in regular LIST format so any number
0388                    ;of hex values or strings in quotes may be entered.
0388
0388                    ENTER:
0388 B9 05 00                   MOV     CX,5            ;5 digits in address
038B E8 5C FF                   CALL    GETHEX          ;Get ENTER address
038E E8 12 FE                   CALL    GETSEG          ;Convert to seg/disp format
0391                    ;Adjust segment and displacement so we are in the middle
0391                    ;of the segment instead of the very bottom. This allows
0391                    ;backing up a long way.
0391 82 EC 08                   SUB     AH,8            ;Adjust segment 32K down
0394 80 C6 80                   ADD     DH,80H          ; and displacement 32K up
0397 50                         PUSH    AX              ;Save for later
0398 52                         PUSH    DX
0399 E8 89 FD                   CALL    SCANB           ;Any more parameters?
039C 75 DD                      JNZ     GETLIST         ;If not end-of-line get list
039E 5F                         POP     DI              ;Displacement of ENTER
039F 07                         POP     ES              ;Segment
03A0                    GETROW:
03A0 E8 9A FD                   CALL    OUTDI           ;Print address of entry
```

```
03A3 E8 CD FD              CALL    BLANK           ;Leave a space
03A6                GETBYTE:
03A6 26                    SEG     ES
03A7 8A 05                 MOV     AL,[DI]         ;Get current value
03A9 E8 A7 FD              CALL    HEX             ;And display it
03AC B0 2E                 MOV     AL,"."
03AE E8 B7 FD              CALL    OUT             ;Prompt for new value
03B1 B9 02 00              MOV     CX,2            ;Max of 2 digits in new value
03B4 BA 00 00              MOV     DX,0            ;Intial new value
03B7                GETDIG:
03B7 E8 48 FD              CALL    IN              ;Get digit from user
03BA 8A E0                 MOV     AH,AL           ;Save
03BC E8 4B FF              CALL    HEXCHK          ;Hex digit?
03BF 86 E0                 XCHG    AH,AL           ;Need original for echo
03C1 72 0C                 JC      NOHEX           ;If not, try special command
03C3 E8 A2 FD              CALL    OUT             ;Echo to console
03C6 8A F2                 MOV     DH,DL           ;Rotate new value
03C8 8A D4                 MOV     DL,AH           ;And include new digit
03CA E2 EB                 LOOP    GETDIG          ;At most 2 digits
03CC                ;We have two digits, so all we will accept now is a command.
03CC                WAIT:
03CC E8 33 FD              CALL    IN              ;Get command character
03CF                NOHEX:
03CF 3C 08                 CMP     AL,8            ;Backspace
03D1 74 19                 JZ      BS
03D3 3C 7F                 CMP     AL,7FH          ;RUBOUT
03D5 74 15                 JZ      BS
03D7 3C 2D                 CMP     AL,"-"          ;Back up to previous address
03D9 74 4D                 JZ      PREV
03DB 3C 0D                 CMP     AL,13           ;All done with command?
03DD 74 2F                 JZ      EOL
03DF 3C 20                 CMP     AL," "          ;Go to next address
03E1 74 31                 JZ      NEXT
03E3                ;If we got here, character was invalid. Sound bell.
03E3 B0 07                 MOV     AL,7
03E5 E8 80 FD              CALL    OUT
03E8 E3 E2                 JCXZ    WAIT            ;CX=0 means no more digits
03EA EB CB                 JP      GETDIG          ;Don't have 2 digits yet
03EC                BS:
03EC 82 F9 02              CMP     CL,2            ;CX=2 means nothing typed yet
03EF 74 C6                 JZ      GETDIG          ;Can't back up over nothing
03F1 FE C1                 INC     CL              ;Accept one more character
03F3 8A D6                 MOV     DL,DH           ;Rotate out last digit
03F5 8A F5                 MOV     DH,CH           ;Zero this digit
03F7 E8 15 FD              CALL    BACKUP          ;Physical backspace
03FA EB BB                 JP      GETDIG          ;Get more digits
03FC
03FC                ;If new value has been entered, convert it to binary and
03FC                ;put into memory. Always bump pointer to next location
03FC
03FC                STORE:
03FC 82 F9 02              CMP     CL,2            ;CX=2 means nothing typed yet
03FF 74 0B                 JZ      NOSTO           ;So no new value to store
0401                ;Rotate DH left 4 bits to combine with DL and make a byte value
0401 51                    PUSH    CX
0402 B1 04                 MOV     CL,4
0404 D2 E6                 SHL     DH,CL
0406 59                    POP     CX
0407 0A D6                 OR      DL,DH           ;Hex is now converted to binary
0409 26                    SEG     ES
040A 88 15                 MOV     [DI],DL         ;Store new value
040C                NOSTO:
040C 47                    INC     DI              ;Prepare for next location
040D C3                    RET
040E                EOL:
040E E8 EB FF              CALL    STORE           ;Enter the new value
0411 E9 E0 FC              JMP     CRLF            ;CR/LF and terminate
0414                NEXT:
0414 E8 E5 FF              CALL    STORE           ;Enter new value
0417 41                    INC     CX              ;Leave a space plus two for
```

```
0418 41                         INC     CX              ;  each digit not entered
0419 E8 5B FD                   CALL    TAB
041C 8B C7                      MOV     AX,DI           ;Next memory address
041E 24 07                      AND     AL,7            ;Check for 8-byte boundary
0420 75 84                      JNZ     GETBYTE         ;Take 8 per line
0422                    NEWROW:
0422 E8 CF FC                   CALL    CRLF            ;Terminate line
0425 E9 78 FF                   JMP     GETROW          ;Print address on new line
0428                    PREV:
0428 E8 D1 FF                   CALL    STORE           ;Enter the new value
042B                    ;DI has been bumped to next byte. Drop it 2 to go to previous addr
042B 4F                         DEC     DI
042C 4F                         DEC     DI
042D EB F3                      JP      NEWROW          ;Terminate line after backing up
042F
042F                    ;Perform register dump if no parameters or set register if a
042F                    ;register designation is a parameter.
042F
042F                    REG:
042F E8 EA FC                   CALL    SCANP
0432 74 62                      JZ      DISPREG
0434 8A 15                      MOV     DL,[DI]
0436 47                         INC     DI
0437 8A 35                      MOV     DH,[DI]
0439 82 FE 0D                   CMP     DH,13
043C 74 76                      JZ      FLAG
043E 47                         INC     DI
043F E8 20 FF                   CALL    GETEOL
0442 82 FE 20                   CMP     DH," "
0445 74 6D                      JZ      FLAG
0447 BF D7 06                   MOV     DI,REGTAB
044A 92                         XCHG    AX,DX
044B 0E                         PUSH    CS
044C 07                         POP     ES
044D B9 0E 00                   MOV     CX,REGTABLEN
0450 F2                         REPNZ
0451 AF                         SCAW
0452 75 3C                      JNZ     BADREG
0454 0B C9                      OR      CX,CX
0456 75 06                      JNZ     NOTPC
0458 4F                         DEC     DI
0459 4F                         DEC     DI
045A 2E                         SEG     CS
045B 8B 45 FE                   MOV     AX,[DI-2]
045E                    NOTPC:
045E E8 07 FD                   CALL    OUT
0461 8A C4                      MOV     AL,AH
0463 E8 02 FD                   CALL    OUT
0466 E8 0A FD                   CALL    BLANK
0469 1E                         PUSH    DS
046A 07                         POP     ES
046B 8D 9D C3 FA                LEA     BX,[DI+REGDIF-2]
046F 8B 17                      MOV     DX,[BX]
0471 E8 D8 FC                   CALL    OUT16
0474 E8 7D FC                   CALL    CRLF
0477 B0 3A                      MOV     AL,":"
0479 E8 EC FC                   CALL    OUT
047C E8 42 FC                   CALL    INBUF
047F E8 A3 FC                   CALL    SCANB
0482 74 0B                      JZ      RET3
0484 B9 04 00                   MOV     CX,4
0487 E8 63 FE                   CALL    GETHEX1
048A E8 D5 FE                   CALL    GETEOL
048D 89 17                      MOV     [BX],DX
048F C3                 RET3:   RET
0490                    BADREG:
0490 B8 42 52                   MOV     AX,5200H+"B"    ;BR ERROR
0493 E9 96 00                   JMP     ERR
0496                    DISPREG:
0496 BE D7 06                   MOV     SI,REGTAB
```

```
0499 BB 9C 01                 MOV     BX,AXSAVE
049C B9 08 00                 MOV     CX,8
049F E8 65 00                 CALL    DISPREGLINE
04A2 E8 4F FC                 CALL    CRLF
04A5 B9 05 00                 MOV     CX,5
04A8 E8 5C 00                 CALL    DISPREGLINE
04AB E8 C5 FC                 CALL    BLANK
04AE E8 93 00                 CALL    DISPFLAGS
04B1 E9 40 FC                 JMP     CRLF
04B4                  FLAG:
04B4 82 FA 46                 CMP     DL,"F"
04B7 75 D7                    JNZ     BADREG
04B9 E8 88 00                 CALL    DISPFLAGS
04BC B0 2D                    MOV     AL,"-"
04BE E8 A7 FC                 CALL    OUT
04C1 E8 FD FB                 CALL    INBUF
04C4 E8 5E FC                 CALL    SCANB
04C7 33 DB                    XOR     BX,BX
04C9 8B 16 B6 01              MOV     DX,[FSAVE]
04CD                  GETFLG:
04CD 8B F7                    MOV     SI,DI
04CF AD                       LODW
04D0 3C 0D                    CMP     AL,13
04D2 74 66                    JZ      SAVCHG
04D4 82 FC 0D                 CMP     AH,13
04D7 74 66                    JZ      FLGERR
04D9 BF F3 06                 MOV     DI,FLAGTAB
04DC B9 20 00                 MOV     CX,32
04DF 0E                       PUSH    CS
04E0 07                       POP     ES
04E1 F2                       REPNE
04E2 AF                       SCAW
04E3 75 5A                    JNZ     FLGERR
04E5 8A E9                    MOV     CH,CL
04E7 80 E1 0F                 AND     CL,0FH
04EA B8 01 00                 MOV     AX,1
04ED D3 C0                    ROL     AX,CL
04EF 85 C3                    TEST    AX,BX
04F1 75 33                    JNZ     REPFLG
04F3 0B D8                    OR      BX,AX
04F5 0B D0                    OR      DX,AX
04F7 F6 C5 10                 TEST    CH,16
04FA 75 02                    JNZ     NEXFLG
04FC 33 D0                    XOR     DX,AX
04FE                  NEXFLG:
04FE 8B FE                    MOV     DI,SI
0500 1E                       PUSH    DS
0501 07                       POP     ES
0502 E8 17 FC                 CALL    SCANP
0505 EB C6                    JP      GETFLG
0507                  DISPREGLINE:
0507 2E                       SEG     CS
0508 AD                       LODW
0509 E8 5C FC                 CALL    OUT
050C 8A C4                    MOV     AL,AH
050E E8 57 FC                 CALL    OUT
0511 B0 3D                    MOV     AL,"="
0513 E8 52 FC                 CALL    OUT
0516 8B 17                    MOV     DX,[BX]
0518 43                       INC     BX
0519 43                       INC     BX
051A E8 2F FC                 CALL    OUT16
051D E8 53 FC                 CALL    BLANK
0520 E8 50 FC                 CALL    BLANK
0523 E2 E2                    LOOP    DISPREGLINE
0525 C3                       RET
0526                  REPFLG:
0526 B8 44 46                 MOV     AX,4600H+"D"      ;DF ERROR
0529                  FERR:
0529 E8 0E 00                 CALL    SAVCHG
```

```
052C                        ERR:
052C E8 39 FC                        CALL     OUT
052F 8A C4                           MOV      AL,AH
0531 E8 34 FC                        CALL     OUT
0534 BE 6B 07                        MOV      SI,ERRMES
0537 E9 3B FE                        JMP      PRINT
053A                        SAVCHG:
053A 89 16 B6 01                     MOV      [FSAVE],DX
053E C3                              RET
053F                        FLGERR:
053F B8 42 46                        MOV      AX,4600H+"B"      ;BF ERROR
0542 EB E5                           JP       FERR
0544                        DISPFLAGS:
0544 BE F3 06                        MOV      SI,FLAGTAB
0547 B9 10 00                        MOV      CX,16
054A 8B 16 B6 01                     MOV      DX,[FSAVE]
054E                        DFLAGS:
054E 2E                              SEG      CS
054F AD                              LODW
0550 D1 E2                           SHL      DX
0552 72 04                           JC       FLAGSET
0554 2E                              SEG      CS
0555 8B 44 1E                        MOV      AX,[SI+30]
0558                        FLAGSET:
0558 0B C0                           OR       AX,AX
055A 74 0B                           JZ       NEXTFLG
055C E8 09 FC                        CALL     OUT
055F 8A C4                           MOV      AL,AH
0561 E8 04 FC                        CALL     OUT
0564 E8 0C FC                        CALL     BLANK
0567                        NEXTFLG:
0567 E2 E5                           LOOP     DFLAGS
0569 C3                              RET
056A
056A                        ;Trace 1 instruction or the number of instruction specified
056A                        ;by the parameter using 8086 trace mode. Registers are all
056A                        ;set according to values in save area
056A
056A                        TRACE:
056A E8 AF FB                        CALL     SCANP
056D E8 98 FD                        CALL     HEXIN
0570 BA 01 00                        MOV      DX,1
0573 72 06                           JC       STOCNT
0575 B9 04 00                        MOV      CX,4
0578 E8 6F FD                        CALL     GETHEX
057B                        STOCNT:
057B 89 16 02 01                     MOV      [TCOUNT],DX
057F E8 E0 FD                        CALL     GETEOL
0582                        STEP:
0582 C7 06 00 01 00 00               MOV      [BRKCNT],0
0588 80 0E B7 01 01                  OR       B,[FSAVE+1],1
058D                        EXIT:
058D C7 06 0C 00 D1 05               MOV      [12],BREAKFIX
0593 8C 0E 0E 00                     MOV      [14],CS
0597 C7 06 04 00 D8 05               MOV      [4],REENTER
059D 8C 0E 06 00                     MOV      [6],CS
05A1 FA                              DI
05A2 C7 06 64 00 D8 05               MOV      [64H],REENTER
05A8 8C 0E 66 00                     MOV      [66H],CS
05AC BC 9C 01                        MOV      SP,STACK
05AF 58                              POP      AX
05B0 5B                              POP      BX
05B1 59                              POP      CX
05B2 5A                              POP      DX
05B3 5D                              POP      BP
05B4 5D                              POP      BP
05B5 5E                              POP      SI
05B6 5F                              POP      DI
05B7 07                              POP      ES
05B8 07                              POP      ES
```

```
05B9 17                          POP     SS
05BA 8B 26 A4 01                 MOV     SP,[SPSAVE]
05BE FF 36 B6 01                 PUSH    [FSAVE]
05C2 FF 36 B2 01                 PUSH    [CSSAVE]
05C6 FF 36 B4 01                 PUSH    [IPSAVE]
05CA 8E 1E AC 01                 MOV     DS,[DSSAVE]
05CE CF                          IRET
05CF EB B1              STEP1:   JP      STEP
05D1
05D1                    ;Re-entry point from breakpoint. Need to decrement instruction
05D1                    ;pointer so it points to location where breakpoint actually
05D1                    ;occured.
05D1
05D1                    BREAKFIX:
05D1 87 EC                       XCHG    SP,BP
05D3 FF 4E 00                    DEC     [BP]
05D6 87 EC                       XCHG    SP,BP
05D8
05D8                    ;Re-entry point from trace mode or interrupt during
05D8                    ;execution. All registers are saved so they can be
05D8                    ;displayed or modified.
05D8
05D8                    REENTER:
05D8 2E                          SEG     CS
05D9 89 26 A4 09                 MOV     [SPSAVE+SEGDIF],SP
05DD 2E                          SEG     CS
05DE 8C 16 B0 09                 MOV     [SSSAVE+SEGDIF],SS
05E2 33 E4                       XOR     SP,SP
05E4 8E D4                       MOV     SS,SP
05E6 BC B0 01                    MOV     SP,RSTACK
05E9 06                          PUSH    ES
05EA 1E                          PUSH    DS
05EB 57                          PUSH    DI
05EC 56                          PUSH    SI
05ED 55                          PUSH    BP
05EE 4C                          DEC     SP
05EF 4C                          DEC     SP
05F0 52                          PUSH    DX
05F1 51                          PUSH    CX
05F2 53                          PUSH    BX
05F3 50                          PUSH    AX
05F4 16                          PUSH    SS
05F5 1F                          POP     DS
05F6 8B 26 A4 01                 MOV     SP,[SPSAVE]
05FA 8E 16 B0 01                 MOV     SS,[SSSAVE]
05FE 8F 06 B4 01                 POP     [IPSAVE]
0602 8F 06 B2 01                 POP     [CSSAVE]
0606 58                          POP     AX
0607 80 E4 FE                    AND     AH,OFEH
060A A3 B6 01                    MOV     [FSAVE],AX
060D 89 26 A4 01                 MOV     [SPSAVE],SP
0611 1E                          PUSH    DS
0612 17                          POP     SS
0613 1E                          PUSH    DS
0614 07                          POP     ES
0615 BC 9C 01                    MOV     SP,STACK
0618 C7 06 64 00 BB 06           MOV     [64H],INT
061E B0 20                       MOV     AL,20H
0620 E6 F2                       OUT     BASE+2
0622 FB                          EI
0623 FC                          UP
0624 E8 CD FA                    CALL    CRLF
0627 E8 6C FE                    CALL    DISPREG
062A FF 0E 02 01                 DEC     [TCOUNT]
062E 75 9F                       JNZ     STEP1
0630                    ENDGO:
0630 BE 04 01                    MOV     SI,BPTAB
0633 8B 0E 00 01                 MOV     CX,[BRKCNT]
0637 E3 10                       JCXZ    COMJMP
0639                    CLEARBP:
```

```
0639 8B 54 14                           MOV     DX,[SI+BPLEN]
063C AD                                 LODW
063D 50                                 PUSH    AX
063E E8 62 FB                           CALL    GETSEG
0641 8E C0                              MOV     ES,AX
0643 8B FA                              MOV     DI,DX
0645 58                                 POP     AX
0646 AA                                 STOB
0647 E2 F0                              LOOP    CLEARBP
0649 E9 3B FA                   COMJMP: JMP     COMMAND
064C
064C                            ;Input from the specified port and display result
064C
064C                            INPUT:
064C B9 04 00                           MOV     CX,4              ;Port may have 4 digits
064F E8 98 FC                           CALL    GETHEX            ;Get port number in DX
0652 EC                                 INB     DX                ;Variable port input
0653 E8 FD FA                           CALL    HEX               ;And display
0656 E9 9B FA                           JMP     CRLF
0659
0659                            ;Output a value to specified port.
0659
0659                            OUTPUT:
0659 B9 04 00                           MOV     CX,4              ;Port may have 4 digits
065C E8 8B FC                           CALL    GETHEX            ;Get port number
065F 52                                 PUSH    DX                ;Save while we get data
0660 B9 02 00                           MOV     CX,2              ;Byte output only
0663 E8 84 FC                           CALL    GETHEX            ;Get data to output
0666 92                                 XCHG    AX,DX             ;Output data in AL
0667 5A                                 POP     DX                ;Port in DX
0668 EE                                 OUTB    DX                ;Variable port output
0669 C3                                 RET
066A
066A                            ;Jump to program, setting up registers according to the
066A                            ;save area. Up to 10 breakpoint addresses may be specified.
066A
066A                            GO:
066A BB 18 01                           MOV     BX,LINEBUF
066D 33 F6                              XOR     SI,SI
066F                            GO1:
066F E8 AA FA                           CALL    SCANP
0672 74 19                              JZ      EXEC
0674 B9 05 00                           MOV     CX,5
0677 E8 70 FC                           CALL    GETHEX
067A 89 17                              MOV     [BX],DX
067C 88 67 ED                           MOV     [BX-BPLEN+1],AH
067F 43                                 INC     BX
0680 43                                 INC     BX
0681 46                                 INC     SI
0682 83 FE 0B                           CMP     SI,BPMAX+1
0685 75 E8                              JNZ     GO1
0687 B8 42 50                           MOV     AX,5000H+"B"      ;BP ERROR
068A E9 9F FE                           JMP     ERR
068D                            EXEC:
068D 89 36 00 01                        MOV     [BRKCNT],SI
0691 E8 CE FC                           CALL    GETEOL
0694 8B CE                              MOV     CX,SI
0696 E3 1A                              JCXZ    NOBP
0698 BE 04 01                           MOV     SI,BPTAB
069B                            SETBP:
069B 8B 54 14                           MOV     DX,[SI+BPLEN]
069E AD                                 LODW
069F E8 01 FB                           CALL    GETSEG
06A2 8E D8                              MOV     DS,AX
06A4 8B FA                              MOV     DI,DX
06A6 8A 05                              MOV     AL,[DI]
06A8 C6 05 CC                           MOV     B,[DI],0CCH
06AB 06                                 PUSH    ES
06AC 1F                                 POP     DS
06AD 88 44 FE                           MOV     [SI-2],AL
```

```
06B0 E2 E9                      LOOP    SETBP
06B2                    NOBP:
06B2 C7 06 02 01 01 00          MOV     [TCOUNT],1
06B8 E9 D2 FE                   JMP     EXIT
06BB
06BB                    ;Console input interrupt handler. Used to interrupt commands
06BB                    ;or programs under execution (if they have interrupts
06BB                    ;enabled). Control-S causes a loop which waits for any other
06BB                    ;character to be typed. Control-C causes abort to command
06BB                    ;mode. All other characters are ignored.
06BB
06BB                    INT:
06BB 50                         PUSH    AX              ;Don't destroy accumulator
06BC                    ;Output End-of-Interrupt commands to slave 8259A. This
06BC                    ;wouldn't be necessary if Automatic End of Interrupt mode
06BC                    ;worked like it was supposed to!
06BC B0 20                      MOV     AL,20H
06BE E6 F2                      OUT     BASE+2
06C0 E4 F6                      IN      DATA            ;Get interrupting character
06C2 24 7F                      AND     AL,7FH          ;ASCII has only 7 bits
06C4 3C 13                      CMP     AL,"S"-"@"      ;Check for Control-S
06C6 75 03                      JNZ     NOSTOP
06C8 E8 37 FA                   CALL    IN              ;Wait for continue character
06CB                    NOSTOP:
06CB 3C 03                      CMP     AL,"C"-"@"      ;Check for Control-C
06CD 74 02                      JZ      BREAK
06CF                    ;Just ignore interrupt - restore AX and return
06CF 58                         POP     AX
06D0 CF                         IRET
06D1                    BREAK:
06D1 E8 20 FA                   CALL    CRLF
06D4 E9 B0 F9                   JMP     COMMAND
06D7                    REGTAB:
06D7 41 58 42 58 43 58          DB      "AXBXCXDXSPBPSIDIDSESSSCSIPPC"
     44 58 53 50 42 50
     53 49 44 49 44 53
     45 53 53 53 43 53
     49 50 50 43
06F3                    REGDIF: EQU     AXSAVE-REGTAB
06F3
06F3                    ;Flags are ordered to correspond with the bits of the flag
06F3                    ;register, most significant bit first, zero if bit is not
06F3                    ;a flag. First 16 entries are for bit set, second 16 for
06F3                    ;bit reset.
06F3
06F3                    FLAGTAB:
06F3 00 00                      DW      0
06F5 00 00                      DW      0
06F7 00 00                      DW      0
06F9 00 00                      DW      0
06FB 4F 56                      DB      "OV"
06FD 44 4E                      DB      "DN"
06FF 45 49                      DB      "EI"
0701 00 00                      DW      0
0703 4E 47                      DB      "NG"
0705 5A 52                      DB      "ZR"
0707 00 00                      DW      0
0709 41 43                      DB      "AC"
070B 00 00                      DW      0
070D 50 45                      DB      "PE"
070F 00 00                      DW      0
0711 43 59                      DB      "CY"
0713 00 00                      DW      0
0715 00 00                      DW      0
0717 00 00                      DW      0
0719 00 00                      DW      0
071B 4E 56                      DB      "NV"
071D 55 50                      DB      "UP"
071F 44 49                      DB      "DI"
0721 00 00                      DW      0
```

```
0723 50 4C                   DB      "PL"
0725 4E 5A                   DB      "NZ"
0727 00 00                   DW      0
0729 4E 41                   DB      "NA"
072B 00 00                   DW      0
072D 50 4F                   DB      "PO"
072F 00 00                   DW      0
0731 4E 43                   DB      "NC"
0733
0733                 ;Initialization table. First byte of each entry is no.
0733                 ;of bytes to output to the corresponding port. That
0733                 ;many initialization bytes follow.
0733
0733                 INITTABLE:
0733                 ;Port BASE+0 - Master 8259A. Intialization Command Word (ICW)
0733                 ;One sets level-triggered mode, multiple 8259As, require
0733                 ;ICW4.
0733 01                      DB      1
0734 19                      DB      19H
0735                 ;Port BASE+1 - Master 8259A. ICW2 sets vector base to 10H
0735                 ;ICW3 sets a slave on interrupt input 1; ICW4 sets buffered
0735                 ;mode, as a master, with Automatic End of Interrupt, 8086
0735                 ;vector; Operation Command Word (OCW) One sets interrupt
0735                 ;mask to enable line 1 (slave 8259A) only.
0735 04                      DB      4
0736 10 02 0F FD             DB      10H,2,0FH,0FDH
073A                 ;Port BASE+2 - Slave 8259A. ICW1 sets level-triggered mode,
073A                 ;multiple 8259As, require ICW4.
073A 01                      DB      1
073B 19                      DB      19H
073C                 ;Port BASE+3 - Slave 8259A. ICW2 sets vector base to 18H
073C                 ;ICW3 sets slave address as 1; ICW4 sets buffered mode,
073C                 ;as slave, with Automatic End of Interrupt (which doesn't
073C                 ;work in slaves), 8086 vector; OCW1 sets interrupt mask
073C                 ;to enable line 1 (serial receive) only.
073C 04                      DB      4
073D 18 01 0B FD             DB      18H,1,0BH,0FDH
0741                 ;Port Base+4 - 9513 Data. 9513 has previously been set
0741                 ;up for Counter 5 mode register with auto increment. Thus
0741                 ;mode is set to 0B63H, which is no gating, count source is
0741                 ;F1 (4 MHz), reload from load or hold, count down repetitively
0741                 ;in binary, with output toggle. Load register is set to
0741                 ;0007H, and Hold register is set to 0006H. Thus we
0741                 ;alternately divide by 7 and 6, which is divided by 2 by
0741                 ;the output toggle, thus providing a square wave of
0741                 ;4 MHz/13 = 307.7 kHz, which divided by 16 in the 8251A
0741                 ;provides 19,230 baud (0.16% high).
0741 06                      DB      6
0742 63 0B 07 00 06 00       DB      63H,0BH,7,0,6,0
0748                 ;Port BASE+5 - 9513 Control. Load and arm counter 5,
0748                 ;enabling baud rate generation. Then select counter
0748                 ;5 mode register, in case baud rate wasn't right.
0748 02                      DB      2
0749 70 05                   DB      70H,5
074B                 ;Port BASE+6 - 8251A Data. No initialization to this port.
074B 00                      DB      0
074C                 ;Port BASE+7 - 8251A Control. Since it is not possible to
074C                 ;know whether the 8251A next expects a Mode Instruction or
074C                 ;a Command Instruction, a dummy byte is sent which could
074C                 ;safely be interpreted as either but guarantees it is now
074C                 ;expecting a Command. The command sent is Internal Reset
074C                 ;which causes it to start expecting a mode. The mode sent
074C                 ;is for 2 stop bits, no parity, 8 data bits, 16X clock.
074C                 ;This is followed by the command to error reset, enable
074C                 ;transmitter and receiver, set RTS and DTR to +12V.
074C 04                      DB      4
074D B7 77 CE 37             DB      0B7H,77H,0CEH,37H
0751 0D 0A 0A 53 43 50 HEADER: DM    13,10,10,"SCP 8086 Monitor 1.4",13,10
     20 38 30 38 36 20
     4D 6F 6E 69 74 6F
     72 20 31 2E 34 0D
     8A
076A 5E                 SYNERR: DB      '^'
076B 20 45 72 72 6F 72  ERRMES: DM      " Error",13,10
     0D 8A
0773 08 20 88           BACMES: DM      8,32,8
0776
0776                    ;Disk boot.
0776
0776                    BOOT:
0776 57                         PUSH    DI
0777
0777                    ;************************************************
0777
0777                    ;Boot for Cromemco 4FDC disk controller with either
0777                    ;large or small disks. Loads track 0, sector 1 into LOAD.
0777
0777                            IF      CROMEMCO4FDC
0777 B0 01                      MOV     AL,1
0779 E6 02                      OUT     2               ;Reset 4FDC serial I/O
077B B0 84                      MOV     AL,84H
077D E6 00                      OUT     0               ;and set for 300 baud
077F B0 7F                      MOV     AL,7FH
0781 E6 04                      OUT     4
0783 B2 21                      MOV     DL,21H
0785                    RETRY:
0785 B0 D0                      MOV     AL,0D0H
0787 E6 30                      OUTB    30H
0789                    READY:
0789 E4 30                      INB     30H
078B D0 C8                      ROR     AL
078D 72 FA                      JC      READY
078F 80 F2 10                   XOR     DL,10H
0792 8A C2                      MOV     AL,DL
0794 E6 34                      OUTB    34H
0796 BF 00 02                   MOV     DI,LOAD
0799 B0 0C                      MOV     AL,12
079B E6 30                      OUTB    30H
079D                    HOME:
079D E4 34                      INB     34H
079F D0 C8                      ROR     AL
07A1 73 FA                      JNC     HOME
07A3 E4 30                      INB     30H
07A5 24 98                      AND     AL,98H
07A7 75 DC                      JNZ     RETRY
07A9 B0 01                      MOV     AL,1
07AB E6 32                      OUTB    32H
07AD B9 80 00                   MOV     CX,80H
07B0 8A C2                      MOV     AL,DL
07B2 0C 80                      OR      AL,80H
07B4 E6 34                      OUTB    34H
07B6 B0 8C                      MOV     AL,8CH
07B8 E6 30                      OUTB    30H
07BA                    READ:
07BA E4 34                      INB     34H
07BC D0 C8                      ROR     AL
07BE 72 0B                      JC      DONE
07C0 E4 33                      INB     33H
07C2 AA                         STOB
07C3 E2 F5                      LOOP    READ
07C5                    WSTAT:
07C5 E4 34                      INB     34H
07C7 D0 C8                      ROR     AL
07C9 73 FA                      JNC     WSTAT
07CB                    DONE:
07CB E4 30                      INB     30H
07CD 24 9C                      AND     AL,9CH
07CF 75 B4                      JNZ     RETRY
07D1                            ENDIF
07D1
```

```
07D1                           ;Successful read
07D1 C7 06 B2 01 00 00                 MOV     [CSSAVE],0
07D7 C7 06 B4 01 00 02                 MOV     [IPSAVE],LOAD
07DD 5F                                POP     DI
07DE E9 89 FE                          JMP     GO
07E1

Error Count =     0

0777                           ;*****************************************************
0777
0777                           ;Boot for North Star disk, single density.
0777                           ;Loads track 0, sector 0 into address LOAD
0777
0777                                   IF      NORTHSTARSD
0777
0777                           ;Disk command equates
0777
0777                           SEL:    EQU     1
0777                           STP1:   EQU     9
0777                           STP2:   EQU     8
0777                           NOP:    EQU     10H
0777                           SEC:    EQU     14H
0777                           STPOUT: EQU     1CH
0777                           RD:     EQU     40H
0777                           BST:    EQU     20H
0777
0777 1E                                PUSH    DS
0778 B8 B8 FE                          MOV     AX,0FEB8H
077B 8E D8                             MOV     DS,AX
077D A0 01 00                          MOV     AL,[SEL]
0780 B9 14 00                          MOV     CX,20
0783                           MOTOR:
0783 E8 19 00                          CALL    SECTOR
0786 E2 FB                             LOOP    MOTOR
0788                           CHKTRK:
0788 F6 06 1C 00 01                    TEST    B,[STPOUT],1
078D 75 1B                             JNZ     ONTRACK
078F A0 09 00                          MOV     AL,[STP1]
0792 D4 0A                             AAM
0794 A0 08 00                          MOV     AL,[STP2]
0797 E8 05 00                          CALL    SECTOR
079A E8 02 00                          CALL    SECTOR
079D EB E9                             JP      CHKTRK
079F                           SECTOR:
079F A0 14 00                          MOV     AL,[SEC]
07A2                           SECLP:
07A2 A0 30 00                          MOV     AL,[BST+NOP]
07A5 A8 80                             TEST    AL,80H
07A7 74 F9                             JZ      SECLP
07A9 C3                                RET
07AA                           ONTRACK:
07AA BF 00 02                          MOV     DI,LOAD
07AD B9 18 01                          MOV     CX,280
07B0 BB 50 00                          MOV     BX,RD+NOP
07B3                           GETSEC:
07B3 E8 E9 FF                          CALL    SECTOR
07B6 24 0F                             AND     AL,0FH
07B8 75 F9                             JNZ     GETSEC
07BA                           GETSYNC:
07BA F6 06 10 00 04                    TEST    B,[NOP],4
07BF E1 F9                             LOOPZ   GETSYNC
07C1 74 E7                             JZ      ONTRACK
07C3 B9 00 01                          MOV     CX,100H
07C6 32 D2                             XOR     DL,DL
07C8 D5 0A                             AAD
07CA                           READ:
07CA 8A 07                             MOV     AL,[BX]
07CC AA                                STOB                            ;Uses ES
```

```
07CD 32 D0                      XOR     DL,AL
07CF D0 C2                      ROL     DL
07D1 D5 0A                      AAD
07D3 E2 F5                      LOOP    READ
07D5 8A 07                      MOV     AL,[BX]
07D7 3A C2                      CMP     AL,DL
07D9 75 CF                      JNZ     ONTRACK
07DB 1F                         POP     DS
07DC                            ENDIF
07DC                    ;Successful read
07DC C7 06 B2 01 00 00          MOV     [CSSAVE],0
07E2 C7 06 B4 01 00 02          MOV     [IPSAVE],LOAD
07E8 5F                         POP     DI
07E9 E9 7E FE                   JMP     GO
07EC

Error Count =    0


0777                    ;************************************************
0777
0777                    ;Boot for Tarbell disk controllers. Load track 0,
0777                    ;sector 1 into LOAD.
0777
0777                            IF      TARBELL
0777
0777                    DISK:   EQU     78H
0777
0777                    RETRY:
0777 B0 D0                      MOV     AL,0D0H
0779 E6 78                      OUTB    DISK
077B                    READY:
077B E4 78                      INB     DISK
077D D0 C8                      ROR     AL
077F 72 FA                      JC      READY
0781 BF 00 02                   MOV     DI,LOAD
0784 B0 0E                      MOV     AL,0EH  ;Home command @ 10ms/track
0786 E6 78                      OUTB    DISK
0788 E4 7C                      INB     DISK+4
078A E4 78                      INB     DISK
078C 24 98                      AND     AL,98H
078E 75 E7                      JNZ     RETRY
0790 B0 01                      MOV     AL,1
0792 E6 7A                      OUTB    DISK+2
0794 B9 80 00                   MOV     CX,80H
0797 B0 8C                      MOV     AL,8CH
0799 E6 78                      OUTB    DISK
079B                    READ:
079B E4 7C                      INB     DISK+4
079D D0 C0                      ROL     AL
079F 73 0B                      JNC     DONE
07A1 E4 7B                      INB     DISK+3
07A3 AA                         STOB
07A4 E2 F5                      LOOP    READ
07A6                    WSTAT:
07A6 E4 7C                      INB     DISK+4
07A8 D0 C0                      ROL     AL
07AA 72 FA                      JC      WSTAT
07AC                    DONE:
07AC E4 78                      INB     DISK
07AE 24 9C                      AND     AL,9CH
07B0 75 C5                      JNZ     RETRY
07B2                            ENDIF
07B2                    ;Successful read
07B2 C7 06 B2 01 00 00          MOV     [CSSAVE],0
07B8 C7 06 B4 01 00 02          MOV     [IPSAVE],LOAD
07BE 5F                         POP     DI
07BF E9 A8 FE                   JMP     GO
07C2

Error Count =    0
```

```
0777                     ;****************************************************
0777
0777                             IF      OTHER
0777
0777                     ;User may insert customized disk boot here. All
0777                     ;registers are available, stack pointer is valid
0777                     ;and interrupts are enabled. Stack should be at
0777                     ;same level on fall-through to code below. Last
0777                     ;address available is 07DF hex.
0777
0777                             ORG     7E0H    ;Simulate boot of maximum length
07E0
07E0                             ENDIF
07E0                     ;Successful read
07E0 C7 06 B2 01 00 00           MOV     [CSSAVE],0
07E6 C7 06 B4 01 00 02           MOV     [IPSAVE],LOAD
07EC 5F                          POP     DI
07ED E9 7A FE                    JMP     GO
07F0

Error Count =    0
```