8086 Monitor

For Use with the SCP 300 CPU Support Board
CONTENTS

Getting Started ............................................. 3
Parameters .................................................... 4
Commands .................................................... 4
    Boot ................................................... 4
    Dump ................................................. 5
    Enter ............................................... 6
    Fill ............................................... 6
    Go .................................................. 7
    Input ............................................. 7
    Move ............................................... 8
    Output ............................................ 8
    Register .......................................... 8
    Search ............................................. 9
    Trace .............................................. 10

8086 Monitor Assembly Listing ............................. 11
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 "a" 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 0FFFFH 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 specify 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.

SCP 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
>140
>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 FF FF FF FF FF FF FF FF FF FF
00410 DD FB DF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00420 BF FF BF FF BF BF BF 6F FF FF
>
>D445 463
00445 FF DF 7F-F9 FF 7E FF FE FF FF FF
00450 FF FF FF FF FF FF FF DF-FD D7 FF FF FF FF FF FF FF FF
00460 9F FF FA FF
>
>D80
00080 FF DF FF FF DF FF FF FF-F5 FF FF FF F5 FF FF FF FF FF FF FF FF FF
00090 CE FF FF FB FF FB FF FF-F7 FF FF FF FF FF FF FF FF FF FF FF FF FB
000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF DF DF
000B0 FF FF FB FF FF FF FF FE FF FF FA FF FA FF FF DF F7 FD
000C0 FF FF FF FF FF FF FF FF-F7 FF FF FF FF FF FF E6 FF FF
000D0 FF DF FF FF FF DC FB 7F-FE FF FF FF FF DF ED ED FF
000E0 FF FF FF FF FB FF FF FF-FF FF FF FF FF FF FF 5F FF
000F0 DF F7 FF DE FF FF BD-BF BF F9 FB DF FF DF DF

- 5 -
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 $..TestING........
 00510 60 01 76 00
>

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..Help
 00420 6C 70 0A 0D 48 65 6C 70-FF 7F FF FF FF FF FF FF FF F7 FF 1P..Help........W.
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 (OCCH) 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.

\[\text{AX}=0010 \quad \text{BX}=0000 \quad \text{CX}=0000 \quad \text{DX}=0000 \quad \text{SP}=0C00 \quad \text{BP}=0000 \quad \text{SI}=0000 \quad \text{DI}=0000 \]
\[\text{DS}=0040 \quad \text{ES}=0040 \quad \text{SS}=0040 \quad \text{CS}=0040 \quad \text{IP}=0010 \quad \text{NV UP EI PL NZ AC PO NC} \]

\[\text{AX}=0010 \quad \text{BX}=0000 \quad \text{CX}=0000 \quad \text{DX}=0000 \quad \text{SP}=0C00 \quad \text{BP}=0000 \quad \text{SI}=0000 \quad \text{DI}=0000 \]
\[\text{DS}=0040 \quad \text{ES}=0040 \quad \text{SS}=0040 \quad \text{CS}=0040 \quad \text{IP}=0000 \quad \text{NV UP EI PL NZ AC PO NC} \]

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 FF FE FF
00410 FF FF FE FF FF FF FF FF FE FF
00420 54 45 53 54 49 4E 47 FF-F7 FF FF F6 FF FE FF

> M404 40F 405
> D400 L10
00400 54 45 53 54 49 49 4E 47-F7 F7 FF FF F6 FF FE

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 except 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:

<table>
<thead>
<tr>
<th>FLAG</th>
<th>CLEAR</th>
<th>SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow</td>
<td>NV No Overflow</td>
<td>OV Overflow</td>
</tr>
<tr>
<td>Direction</td>
<td>UP Up (Incrementing)</td>
<td>DN Down (Decrementing)</td>
</tr>
<tr>
<td>Interrupt</td>
<td>DI Disabled Interrupts</td>
<td>EI Enabled Interrupts</td>
</tr>
<tr>
<td>Sign</td>
<td>PL Plus</td>
<td>NG Negative</td>
</tr>
<tr>
<td>Zero</td>
<td>NZ Not Zero</td>
<td>ZR Zero</td>
</tr>
<tr>
<td>Auxiliary Carry</td>
<td>NA No Auxiliary Carry</td>
<td>AC Auxiliary Carry</td>
</tr>
<tr>
<td>Parity</td>
<td>PO Parity Odd</td>
<td>PE Parity Even</td>
</tr>
<tr>
<td>Carry</td>
<td>NC No Carry</td>
<td>CY Carry</td>
</tr>
</tbody>
</table>
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
>R CX
CS 0040
:
>R DFS
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 occurrence the first address of the match is displayed.

```
>S 400 8000 'Help'
00400
00406
0040C
00412
00418
0041E
00424
>D 400 28
00400 48 65 6C 70 0A 0D 48 65-6C 70 0A 0D 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
00420 6C 70 0A 0D 48 65 6C 70 Help...Help
```
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

; Seattle Computer Products 8086 Monitor version 1.5 4/24/80
; by Tim Paterson
; This software is not copyrighted.

; To select a disk boot, set one of the following equates
to 1, the rest to 0.

CROMEMCO4FDGC: EQU 0 ; 1 for 4FDGC, 0 for others
NORTHSTARSDD: EQU 0 ; North Star single density?
TARBEll: EQU 1 ; Tarbell (single or double)?
OTHER: EQU 0 ; User-defined disk

PUTBASE: EQU 100H
LOAD: EQU 200H
ORG 7FH
PUT PUTBASE+7FH
JMP 0,OFF800H ; Power-on jump to monitor

; Baud Rate Table. The 9513 divides 2MHz by these values.
; They are for 9600, 1200, 300, 150, 110 baud

BAUD: DW 13,104,416,832,1144

ORG 100H ; RAM area base address

; System Equates
BASE: EQU 0FH ; CPU Support base port address
STAT: EQU BASE+7 ; UART status port
DATA: EQU BASE+6 ; UART data port
DAV: EQU 2 ; UART data available bit
TRMT: EQU 1 ; UART transmitter ready bit
BUFLEN: EQU 80 ; Maximum length of line input buffer
BPMAX: EQU 10 ; Maximum number of breakpoints
BPLLEN: EQU BPMAX+BPMAX ; Length of breakpoint table
REGTABLEN: EQU 14 ; Number of registers
SEGDIF: EQU 800H ; -OFF800H (ROM address)
PROMPT: EQU "Art
CAN: EQU "0"

; RAM area.
BRKCNT: DS 2 ; Number of breakpoints
TCOUNT: DS 2 ; Number of steps to trace
BPTAB: DS BUFLEN ; Breakpoint table
LINEBUF: DS BUFLEN+1 ; Line input buffer
ALIGN
DS 50 ; Working stack area
STACK:

; Register save area

AXSAVE: DS 2
BSAVE: DS 2
CSAVE: DS 2
DSAVE: DS 2
ESAVE: DS 2
SSSAVE: DS 2
SSSAVE: DS 2
CSSAVE: DS 2
IPSAVE: DS 2

- 11 -
FSAVE: DS 2

;Start of Monitor code

ORG 0
PUT PUTBASE

;One-time initialization

UP
XOR AX,AX
MOV SS,AX
MOV DS,AX
MOV ES,AX
MOV DI,AXSAVE
MOV CX,14
REP
STOW
OR B, [FSAVE+1], 2 ;Enable interrupts
MOV CL,4
MOV AL,40H
MOV DI,DSAVE
REP
STOW
MOV B [SPSAVE+1], 0CH ;Set user stack to 400H+0COOH
MOV SP,STACK
REP
9513
B 0 17
MOV AL,17H
OUT BASE+5 ;Select Master Mode register
B 0 F3
MOV AL,0E3H
OUT BASE+4 ;Low byte of Master Mode
E 0 05
MOV AX,584H
OUTW BASE+4 ;and 05H to BASE+5

;Prepare 9513
;Master Mode now set to 84F3H:
;Scaler set to BCD division
;Enable data pointer increment
;8-bit data bus
;FOUT=100Hz, dividing F5 by 4 (F5=4MHz/10000)
;Both alarm comparators disabled
;Time-of-day enabled
;Counter 5 selected

;Initialize loop. Ports BASE through BASE+7 are initialized
;from table. Each table entry has number of bytes followed by
;data.

BE 33 07
BA F0 00
3A
2E
3B AC
3C 8A C8
3E E3 05
40
40 2E
41 AC
42 EE
43 E2 FB
45
45 42
46 80 FA F8
49 75 EF
4B
4B
4B E8 19 00
4E

;Initialization complete except for determining baud rate.
;Both 8259As are ready to accept interrupts, the 9513 is
;providing 19.2k baud X 16 to the 8251A which is set for
;16X clock and one stop bit.

CALL CHECKB ;Check for correct baud rate
;CHECKB does not return if baud rate is correct
;Initial baud rate (19.2k) was wrong, so run auto-baud routine

INITBAUD:

    MOV    SI,BAUD

;First set up 9513 for slower baud rates (<9600).
;Counter 5 mode register has already been selected.
    MOV    AX,0E823H ;Output 23H to BASE+4
    OUTW   BASE+4 ;and 0E8H to BASE+5

;23H to BASE+4 sets lower half of Counter 5 mode register.
;Reload from Load, count down repetitively in binary,
;toggle output.
;0E8H to BASE+5 disables data pointer sequencing

    MOV    AL,ODH ;Select Counter 5 load reg.
    OUT     BASE+5

INITB:

    SEG    CS

    LODW   ;Get divisor
    OUT     BASE+4 ;Output low byte
    MOV    AL,AH
    OUT     BASE+4 ;Output high byte
    CALL    CHECKB ;Check if baud rate correct
    JP      INITB ;Try next rate if not

CHECKB:

    CALL    IN ;First byte could be messed up
    CALL    IN ;Get carriage return
    CMP    AL,13 ;Correct?
    JZ      MONITOR ;Don't return if correct
    RET ;Didn't get it yet

;Initialization complete, including baud rate.

MONITOR:

;Do auto boot if sense switch 0 is on.
    MOV    DI,LINENBUF
    MOV    B,[DI],13 ;No breakpoints after boot
    IN     BASE+0FH ;Sense switch port
    TEST   AL,1
    JZ      DOMON
    JMP     BOOT

DOMON:

    MOV    SI,HEADER
    CALL    PRINTMES

COMMAND:

;Re-establish initial conditions
    UP
    XOR    AX,AX
    MOV    DS,AX
    MOV    ES,AX
    MOV    SP,STACK
    MOV    [64H],INT ;Set UART interrupt vector
    MOV    [66H],CS
    MOV    AL,PROMPT
    CALL    OUT
    CALL    INBUF ;Get command line

;From now on and throughout command line processing, DI points
;to next character in command line to be processed.
    CALL    SCANB ;Scan off leading blanks
    JZ      COMMAND ;Null command?
    MOV    AL,[DI] ;AL=first non-blank character

;Prepare command letter for table lookup
    SUB    AL,\"B\" ;Low end range check
    JC      ERR1
    CMP    AL,\"T\"+1-\"B\" ;Upper end range check
    JNC    ERR1
    INC    DI
    SHL    AL
    CBW
    XCHG   BX,AX
    ;In BX we can address with it
SEG CS
CALL [BX+CONTAB] ;Execute command
JP COMMAND ;Get next command

;Get input line

INBUF:
MOV DI,LINEBUF ;Next empty buffer location
XOR CX,CX ;Character count
GETCH:
CALL IN ;Get input character
CMP AL,20H ;Check for control characters
JNZ CONTROL ;Rubout is a backspace
CMP AL,7FH
JZ BACKSP ;Echo character
CMP AL,CAN ;Cancel line?
JZ KILL
CMP CX,BUFLEN ;Buffer full?
JBE GETCH ;Drop in to backspace if full
BACKSP:
DEC DI ;Can't backspace over nothing
DEC CX ;Drop pointer
JNZ GETCH ;And character count
CALL BACKUP ;Send physical backspace
GETCH ;Get next char.
CONTROL:
CMP AL,8 ;Check for backspace
JZ BACKSP
CMP AL,13 ;Check for carriage return
GETCH
JNZ GETCH ;Ignore all other control char.
CALL CRLF ;Put the car. ret. in buffer
MOV DI,LINEBUF ;Set up DI for command processing
CRLF:
MOV AL,13
CALL OUT
MOV AL,10
JP OUT
IN:
DI ;Poll, don't interrupt
INB STAT
TEST AL,DAV
JZ IN ;Loop until ready
INB DATA
JZ IN
AND AL,7FH ;Only 7 bits
EIS ;Interrupts OK now
;Physical backspace - blank, backspace, blank
BACKUP:
MOV SI,BACKSP
;Print ASCII message. Last char has bit 7 set
PRINTMES:  SEG CS
       LODB
       CALL OUT
       SHL AL
       JNC PRINTMES
       RET

;Scan for parameters of a command

SCANP:
       CALL SCANB
       CMP B,[DI],"," ;Get first non-blank
       JNE EOLCHK
       INC DI ;One comma between params OK
       ;If not comma, we found param
       ;Skip over comma

;Scan command line for next non-blank character

SCAMB:
       MOV AL,"," ;Back up to first non-blank
       REPE
       SCAB
       DEC DI
       POP CX

EOLCHK:
       CMP B,[DI],13
       RET

;Print the 5-digit hex address of SI and DS

OUTSI:
       MOV DX,DS ;Put DS where we can work with it
       MOV AH,0 ;Will become high bits of DS
       CALL SHIFT4
       ADD DX,SI ;Shift DS four bits
       JP OUTADD ;Compute absolute address
       ;Finish below

;Print 5-digit hex address of DI and ES
;Same as OUTSI above

OUTDI:
       MOV DX,ES
       MOV AH,0
       CALL SHIFT4
       ADD DX,DI
       ;Finish OUTSI here too

OUTADD:
       ADC AH,0 ;Add in carry to high bits
       CALL HIDIG ;Output hex value in AH

;Print out 16-bit value in DX in hex

OUT16:
       MOV AL,DH ;High-order byte first
       CALL HEX
       MOV AL,DL ;Then low-order byte

;Output byte in AL as two hex digits

HEX:
       MOV AH,AL ;Save for second digit
       ;Shift high digit into low 4 bits
       PUSH CX
       MOV CL,4
       SHR AL,CL
       POP CX
       CALL DIGIT ;Output first digit
HIDIG:  MOV  AL, AH  ;Now do digit saved in AH
DIGIT:  AND  AL, OFH  ;Mask to 4 bits
        ;Trick 6-byte hex conversion works on 8086 too.
        ADD  AL, 90H
        DAA
        ADC  AL, 40H
        DAA

;Console output of character in AL

OUT:    PUSH  AX  ;Character to output on stack
       INB    STAT
       AND    AL, TBMT
       JZ     OUT1  ;Wait until ready
       POP    AX
       OUTB   DATA
       RET

;Output one space

BLANK:  MOV    AL, " "
        JP     OUT

;Output the number of blanks in CX

TAB:    CALL    BLANK
        LOOP   TAB
        RET

;Command Table. Command letter indexes into table to get
;address of command. PERR prints error for no such command.

COMTAB: DW     BOOT  ;B
        DW     PERR  ;C
        DW     DUMP  ;D
        DW     ENTER  ;E
        DW     FILL  ;F
        DW     GO  ;G
        DW     PERR  ;H
        DW     INPUT  ;I
        DW     PERR  ;J
        DW     PERR  ;K
        DW     PERR  ;L
        DW     MOVE  ;M
        DW     PERR  ;N
        DW     OUTPUT  ;O
        DW     PERR  ;P
        DW     PERR  ;Q
        DW     REG  ;R
        DW     SEARCH  ;S
        DW     TRACE  ;T

;Given 20-bit address in AH:DX, breaks it down to a segment
;number in AX and a displacement in DX. Displacement is
;always zero except for least significant 4 bits.

GETSEG: MOV    AL, DL  ;AL has least significant 4 bits
        AND    AL, OFH
        CALL   SHIFT4
        MOV    DL, AL  ;Restore lowest 4 bits
        MOV    AL, DH
        MOV    AL, DH
        MOV    AL, DH
        MOV    AL, DH
0180 C3
0181
;Shift AH:DX left 4 bits
0181

SHIFT4:
0182 SHL DX
0183 RCL AH ;1
0184 SHL DX
0185 RCL AH ;2
0186 SHL DX
0187 RCL AH ;3
0188 SHL DX
0189 RCL AH ;4
RET

018A
018B
018C ;RET2:
018D RET

018E
018F
0190 ;RANGE – Looks for parameters defining an address range.
0191 ;The first parameter is a hex number of 5 or less digits
0192 ;which specifies the starting address. The second parameter
0193 ;may specify the ending address, or it may be preceded by
0194 ;"l" and specify a length (4 digits max), or it may be
0195 ;omitted and a length of 128 bytes is assumed. Returns with
0196 ;segment no. in AX and displacement (0–F) in DX.
0197
0198 RANGE:
0199
019A MOV CX,5 ;5 digits max
019B CALL GETHEX ;Get hex number
019C PUSH AX ;Save high 4 bits
019D PUSH DX ;Save low 16 bits
019E CALL SCANP ;Get to next parameter
019F CMP B, [DI], "L" ;Length indicator?
0200 JE GETLEN
0201 MOV DX, 128 ;Default length
0202 CALL HEXIN ;Second parameter present?
0203 JC RNGRET ;If not, use default
0204 MOV CX, 5 ;5 hex digits
0205 CALL GETHEX ;Get ending address
0206 MOV CX, DX ;Low 16 bits of ending addr.
0207 POP DX ;Low 16 bits of starting addr.
0208 POP BX ;BH=hi 4 bits of start addr.
0209 SUB CX, DX ;Compute range
020A SBB AH, BH ;Finish 20-bit subtract
020B JNZ RNGERR ;Range must be less than 64K
020C XCHG AX, BX ;AH=starting, BH=ending hi 4 bits
020D INC CX ;Range must include ending location
020E JP RNGCHK ;Finish range testing and return
020F
0210 GETLEN:
0211 INC DI ;Skip over "L" to length
0212 MOV CX, 4 ;Length may have 4 digits
0213 CALL GETHEX ;Get the range
0214
0215 RNGRET:
0216 MOV CX, DX ;Length
0217 POP DX ;Low 16 bits of starting addr.
0218 POP AX ;AH=hi 4 bits of starting addr.
0219
021A ;RNGCHK verifies that the range lies entirely within one segment.
021B ;CX=0 means count=10000H. Range is within one segment only if
021C ;adding the low 4 bits of the starting address to the count is
021D ;<10000H, because segments can start only on 16-byte boundaries.
021E
021F RNGCHK:
0220 MOV BX, DX ;Low 16 bits of start addr.
0221 AND BX, 0FH ;Low 4 bits of starting addr.
0222 JCCXZ MAXRNG ;If count=10000H then BX must be 0
0223 ADD BX, CX ;Must be <=10000H
0224 JNC GETSEG ;OK if strictly <
0225 MAXRNG:
0226 ;If here because of JCCXZ MAXRNG, we are testing if low 4 bits
0227 ;(in BX) are zero. If we dropped straight in, we are testing
0228 ;for BX+CX=10000H (=0). Either way, zero flag set means
0229 ;withing range.
Dump an area of memory in both hex and ASCII

DUMP:
CALL RANGE                        ;Get range to dump
PUSH AX                          ;Save segment
CALL GETEO             ;Check for errors
POP DS                      ;Set segment
MOV SI,DX            ;SI has displacement in segment

ROW:
CALL OUTSI               ;Print address at start of line
PUSH SI                  ;Save address for ASCII dump

BYTE:
CALL BLANK                ;Space between bytes

BYTE1:
LODB                    ;Get byte to dump
CALL HEX                ;and display it
POP DX                  ;DX has start addr. for ASCII dump
DEC CX                  ;Drop loop count
JNZ ASCII               ;If through do ASCII dump
MOV AX, CX             ;On 16-byte boundary?
TEST AL,OFH            ;Didn't need ASCII addr. yet
JZ ENDRW               ;On 8-byte boundary?

ENDRW:
CALL ASCII              ;Show it in ASCII
JF ROW                 ;Loop until count is zero

ASCII:
PUSH CX                 ;Save byte count
MOV AX, SI             ;Current dump address
MOV SI, DX             ;ASCII dump address
SUB AX, DX             ;AX = length of ASCII dump

;Compute tab length. ASCII dump always appears on right side
;screen regardless of how many bytes were dumped. Figure 3
;characters for each byte dumped and subtract from 31, which
;allows a minimum of 3 blanks after the last byte dumped.
MOV BX, AX            ;Length times 2
SHL AX                ;Length times 3
ADD AX, BX          ;Amount to tab in CX
MOV CX, 51            ;ASCII dump length back in CX

ASCDMP:
LODB                    ;Get ASCII byte to dump
AND AL, 7FH          ;ASCII uses 7 bits
JZ NOPRT            ;Don't try to print RUBOUT
CMP AL, 7FH          ;Check for control characters
JNC PRIN

NOPRT:
MOV AL, ,"            ;If unprintable character

PRIN:
CALL OUT               ;Print ASCII character
LOOP ASCDMP          ;CX times
POP CX                ;Restore overall dump length
JMP CRLF             ;Print CR/LF and return

;Block move one area of memory to another. Overlapping moves
;are performed correctly, i.e., so that a source byte is not
;overwritten until after it has been moved.
026A
026A E8 55 FF
026D 51
0266 50
026F 8B F2
0271 B9 05 00
0274 E8 73 00
0277 E8 E8 00
027A E8 26 FF
027D 8B FA
027F 3B
0280 8E DB
0282 8E C0
0284 59
0285 3B FE
0287 1B C3
0289 72 07
028B
028B 49
028C 03 F1
028E 03 F9
0290 FD
0291 41
0292
0292 A4
0293 49
0294 F3
0295 A4
0296 C3
0297
0297 E8 28 FF
029A 51
029B 50
029C 52
029D E8 84 00
02A0 5F
02A1 07
02A2 59
02A3 3B D9
02A5 BE 18 01
02AA E8 03 02
02AC 73 E6
02AD
02AD 2B CB
02AE 87 D9
02B0 57
02B1 F3
02B2 A4
02B3 5E
02B4
02B4 8B CB
02B6 06
02B7 1F
02B8 EB D8
02BA
02BA
02BA
02BA
02BA
02BA

MOVE:
CALL RANGE ;Get range of source area
PUSH CX ;Save length
PUSH AX ;Save segment
MOV SI,DX ;Set source displacement
MOV CX,5 ;Allow 5 digits
CALL GETHEX ;In destination address
CALL GETSEL ;Check for errors
CALL GETSEG ;Convert dest. to seg_disp
MOV DI,DX ;Set disp. displacement
POP BX ;Source segment
MOV DS,BX ;Destination segment
MOV ES,AX ;Length
POP CX ;Check direction of move
CMP DI,SI ;Extend the CMP to 32 bits
JB COPYLIST ;Move forward into lower mem.
;Otherwise, move backward. Figure end of source and destination
;areas and flip direction flag.
ADD SI,CX ;End of source area
ADD DI,CX ;End of destination area
INC CX ;Reverse direction

COPYLIST:
MOV BX ;Do at least 1 - Range is l-10000H not 0-FFFFH
DEC CX
REF
MOVBX ;Block move
RET

;Fill an area of memory with a list values. If the list
;is bigger than the area, don't use the whole list. If the
;list is smaller, repeat it as many times as necessary.

FILL:
CALL RANGE ;Get range to fill
PUSH CX ;Save length
PUSH AX ;Save segment number
PUSH DX ;Save displacement
CALL LIST ;Get list of values to fill with
POP DI ;Displacement in segment
POP ES ;Segment
POP CX ;Length
CMP BX,CX ;BX is length of fill list
MOV SI,LINEBUF ;List is in line buffer
JNCZ BIGRNG
JAE COPYLIST ;If list is big, copy part of it

BIGRNG:
SUB CX,BX ;How much bigger is area than list?
XCHG CX,BX ;CX=length of list
MOV DI ;Save starting addr. of area
REP
MOVBX ;Move list into area
POP SI

;The list has been copied into the beginning of the
;specified area of memory. SI is the first address
;of that area, DI is the end of the copy of the list
;plus one, which is where the list will begin to repeat.
;All we need to do now is copy [SI] to [DI] until the
;end of the memory area is reached. This will cause the
;list to repeat as many times as necessary.
MOV CX,BX ;Length of area minus list
PUSH ES ;Different index register
POP DS
JP COPYLIST ;Do the block move

;Search a specified area of memory for given list of bytes.
;Print address of first byte of each match.
SEARCH:
CALL RANGE ;Get area to be searched
PUSH CX ;Save count
PUSH AX ;Save segment number
PUSH DX ;Save displacement
CALL LIST ;Get search list
DEC BX ;No. of bytes in list-1
POP DI ;Displacement within segment
POP ES ;Segment
POP CX ;Length to be searched
SUB CX,BX ; minus length of list
SCAN:
MOV SI,LINELIB ;List kept in line buffer
LODB ;Bring first byte into AL
DO SCAN:
SCAB
LOOP NE DOSCAN ;Search for first byte
JNZ RET ;Do at least once by using LOOP
POP BX ;Exit if not found
XCHG BX,CX ;Length of list minus 1
POP DI ;Will resume search here
REPE
CMPB ;Compare rest of string
MOV CX,BX ;Area length back in CX
POP DI ;Next search location
POP BX ;Restore list length
JNZ TEST ;Continue search if no match
DEC DI ;Match address
INC DI ;Print it
OUTDI ;Restore search address
CRLF
TEST:
JCXZ RET ;Look for next occurrence
JP SCAN ;CX is maximum number of digits the number may have.
GETHEX:
CALL SCANP ;Scan to next parameter
GETHEX1:
XOR DX,DX ;Initialize the number
MOV AH,DH
CALL HEXIN ;Get a hex digit
JNC ERROR ;Must be a valid digit
MOV DL,AL ;First 4 bits in position
GETLP:
INC DI ;Next char in buffer
DEC CX ;Digit count
CALL HEXIN ;Get another hex digit?
JC RET ;All done if no more digits
JCCXZ ERROR ;Too many digits?
CALL SHIFT4 ;Multiply by 16
OR DL,AL ;and combine new digit
JP GETLP ;Get more digits
;Check if next character in the input buffer is a hex digit
;and convert it to binary if it is. Carry set if not.
HEXIN:
MOV AL,[DI]
;Check if AL has a hex digit and convert it to binary if it
;is. Carry set if not.
HEXCHK:
SUB AL,"0" ;Kill ASCII numeric bias
JC RET
CMP AL,10
0310 F5         CMC      RET       ;OK if 0-9
0311 73 83      JNC      RET
0313 2C 07      SUB      AL,7      ;Kill A-F bias
0315 3C 0A      CHF      AL,10
0317 72 03      JNC      RET
0319 3C 10      JMP      AL,16
031B F5         CMC
031C C3         RET
031D
031D
031D
031D
031D E8 FC FD   CALL SCANP    ;Scan to parameter
0320 E8 E5 FF   CALL HEXIN    ;Is it in hex?
0323 72 0B      JNC STRINGCHK ;If not, could be a string
0325 B9 02 00   MOV CX,2      ;Only 2 hex digits for bytes
0326 E8 BF FF   CALL GETHEX   ;Get the byte value
0328 88 17      MOV [BX],DL   ;Add to list
032A 43         INC BX
032C F8         RET
032F C3         GRET: CLC
0330
0330 8A 05      MOV AL,[DI]    ;Parameter was OK
0332 3C 27      CMP AL,""
0334 74 06      JZ STRING     ;String?
0336 3C 22      CMP AL,""
0338 74 02      JZ STRING     ;Either quote is all right
033A F9         STC
033B C3         RET
033C
033C 8A E0      MOV AH,AL     ;Not string, not hex - bad
033E 47         INC DI
033F
033F 8A 05      MOV AL,[DI]    ;Next char of string
0341 47         INC DI
0342 3C 0D      CMP AL,13
0344 74 23      JZ ERROR      ;Must find a close quote
0346 3A C4      CMP AH,AL
0348 75 05      JNZ STOSTRG   ;Add new character to list
034A 3A 25      CMP AH,DI
034C 75 E0      JNZ GRET      ;Two quotes in a row?
034E 47         INC DI
034F
034F 88 07      MOV [BX],AL   ;Yes - skip second one
0351 43         INC BX
0352 EB EB        JP STRNGLP   ;Get more characters
0354
0354
0354
0354
0354
0354
0354
0354
035A BB 18 01
0357
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
0362
0362
0362
0362
0362
0362 E8 C0 FD
0365 75 02      GETEOL: CALL SCANB     ;Skip blanks
0367 C3         JNZ ERROR      ;Better be a RETURN
0368
;Command error. DI has been incremented beyond the
;command letter so it must decremented for the
;error pointer to work.

PERR:
  DEC DI

;Syntax error. DI points to character in the input buffer
;which caused error. By subtracting from start of buffer,
;we will know how far to tab over to appear directly below
;it on the terminal. Then print "Error".

ERROR:
  SUB DI,LIN BUF-1 ;How many char processed so far?
  MOV CX,DI ;Parameter for TAB in CX
  CALL TAB ;Directly below bad char
  MOV SI,SY NERR ;Error message

;Print error message and abort to command level

PRINT:
  CALL PRINTM ES
  JMP COMMAND

;Short form of ENTER command. A list of values from the
;command line are put into memory without using normal
;ENTER mode.

GETLIST:
  CALL LIST
  POP DI ;Get the bytes to enter
  POP ES ;Displacement within segment
  POP CX,DX ;Segment to enter into
  MOV SI,LIN BUF ;List of bytes is in line buffer
  MOV CX,BX ;Count of bytes
  REP MOVB ;Enter that byte list
  RET

;Enter values into memory at a specified address. If the
;line contains nothing but the address we go into "enter
;mode", where the address and its current value are printed
;and the user may change it if desired. To change, type in
;new value in hex. Backspace works to correct errors. If
;an illegal hex digit or too many digits are typed, the
;bell is sounded but it is otherwise ignored. To go to the
;next byte (with or without change), hit space bar. To
;back up to a previous address, type "-". On
;every 8-byte boundary a newline is started and the address
;is printed. To terminate command, type carriage return.
;Alternatively, the list of bytes to be entered may be
;included on the original command line immediately following
;the address. This is in regular LIST format so any number
;of hex values or strings in quotes may be entered.

ENTER:
  MOV CX,5 ;5 digits in address
  CALL GETHEX ;Get ENTER address
  CALL GETSEG ;Convert to seg/disp format

;Adjust segment and displacement so we are in the middle
;of the segment instead of the very bottom. This allows
;backing up a long way.
  SUB AH,8 ;Adjust segment 32K down
  ADD DH,80H ;and displacement 32K up
  PUSH AX ;Save for later
  PUSH DX
  CALL SCANB ;Any more parameters?
  JNZ GET LIST ;If not end-of-line get list
  POP DI ;Displacement of ENTER
  POP ES ;Segment

GETROW:
CALL OUTDI ;Print address of entry
CALL BLANK ;Leave a space
GETBYTE:
SEG ES
MOV AL,[DI]
CALL HEX ;And display it
MOV AL,"\n"
CALL OUT ;Prompt for new value
MOV CX,2 ;Max of 2 digits in new value
MOV DX,0 ;Initial new value
GETDIG:
CALL IN ;Get digit from user
MOV AH,AL ;Save
CALL HEXCHK ;Hex digit?
XCHG AH,AL ;Need original for echo
JC NOHEX ;If not, try special command
CALL OUT ;Echo to console
MOV DH,DL ;Rotate new value
MOV DL,AH ;And include new digit
LOOP GETDIG ;At most 2 digits

;We have two digits, so all we will accept now is a command.
WAIT:
CALL IN ;Get command character
NOHEX:
CMP AL,8 ;Backspace
JZ BS ;RUBOUT
CMP AL,7FH
JZ BS
CMP AL,"\n"
JZ PREV
CMP AL,13
JZ EOL
CMP AL,"
JZ NEXT

;If we got here, character was invalid. Sound bell.
MOV AL,7
CALL OUT
JCXZ WAIT ;CX=0 means no more digits
JP GETDIG ;Don't have 2 digits yet

BS:
CMP CL,2
JZ GETDIG ;CX=2 means nothing typed yet
INC CL ;Can't back up over nothing
INC CL ;Accept one more character
MOV DL,DL
MOV DH,CH ;Rotate out last digit
MOV DH,CH ;Zero this digit
CALL BACKUP ;Physical backspace
JP GETDIG ;Get more digits

;If new value has been entered, convert it to binary and
;put into memory. Always bump pointer to next location
STORE:
CMP CL,2
JZ NOSTO ;CX=2 means nothing typed yet
MOV CL,4
INC DI ;So no new value to store
;Rotate DH left 4 bits to combine with DL and make a byte value
PUSH CX
MOV CL,4
SHL DH,CL
POP CX
OR DL,DH
SEG ES
MOV [DI],DL ;Store new value
NOSTO:
INC DI ;Prepare for next location
RET

EOL:
CALL STORE ;Enter the new value
JMP CRLF ;CR/LF and terminate

NEXT:
CALL STORE ;Enter new value
INC CX
INC CX
; Leave a space plus two for
INC CX
; each digit not entered
CALL TAB
; Next memory address
MOV AX,DI
; Check for 8-byte boundary
AND AL,7
GETBYTE
; Take 8 per line
NEWROW:
CALL CRLF
; Terminate line
JMP GETROW
; Print address on new line
PREV:
CALL STORE
; Enter the new value
DEC DI
DEC DI
; DI has been bumped to next byte. Drop it 2 to go to previous addr
DEC DI
JP NEWROW
; Terminate line after backing up

; Perform register dump if no parameters or set register if a
; register designation is a parameter.

REG:
CALL SCANP
JZ DISPREG
MOV DL,[DI]
INC DI
MOV DH,[DI]
CMP DH,13
JZ FLAG
INC DI
CALL GTEOEL
CMP DH,""
JZ FLAG
MOV DI,REGETABL
PUSH CS
POP ES
MOV CX,REGETABLEN
REPNZ SCAW
JNZ BADREG
OR CX,CX
JNZ NOTPC
DEC DI
DEC DI
SEG CS
MOV AX,[DI-2]
NOTPC:
CALL OUT
MOV AL,AH
CALL OUT
CALL BLANK
PUSH DS
POP ES
LEA BX,[DI+REGDIF-2]
MOV DX,[BX]
CALL OUT16
CALL CRLF
MOV AL,"":"
CALL OUT
CALL INBUF
CALL SCANB
JZ RET3
MOV CX,4
CALL GETHX1
CALL GTEOEL
MOV [BX],DX
MOV AX,5200H+"B"
; BR ERROR
RET3: RET
BADREG:
MOV AX,5200H+"B"
; BR ERROR
JMP ERR
0496 BE D7 06
0499 BB 9C 01
049C B9 08 00
049F EB 65 00
04A2 EB 4F FC
04A5 B9 05 00
04A8 EB 5C 00
04AB EB C5 FC
04AE EB 93 00
04B1 E9 40 FC
04B4 82 FA 46
04B7 75 D7
04B9 EB 88 00
04BC B0 2D
04BE EB A7 FC
04C1 EB FD FB
04C4 EB 5E FC
04C7 33 D8
04C9 8B 16 B6 01
04CD BB F7
04CE AD
04D0 3C 0D
04D2 74 66
04D4 82 FC 0D
04D7 74 66
04D9 BF F3 06
04DC B9 20 00
04DF 0E
04E0 07
04E1 F2
04E2 AD
04E3 75 5A
04E5 8A E9
04E7 80 E1 0F
04EA BB 01 00
04ED D3 C0
04EF 85 C3
04F1 75 33
04F3 OB D8
04F5 OB D0
04F7 F6 C5 10
04FA 75 02
04FC 33 D0
04FE 8B FE
0500 1E
0501 07
0502 EB 17 FC
0505 EB C6
0507
0507 2E
0508 AD
0509 EB 5C FC
050C 8A C4
050E EB 57 FC
0511 B0 3D
0513 EB 52 FC
0516 8B 17
0518 43
0519 43
051A EB 2F FC
051D EB 53 FC
0520 EB 50 FC
0523 E2 E2
0525 C3
0526
0526 8B 44 46
0529

0496 BE D7 06
0499 BB 9C 01
049C B9 08 00
049F EB 65 00
04A2 EB 4F FC
04A5 B9 05 00
04A8 EB 5C 00
04AB EB C5 FC
04AE EB 93 00
04B1 E9 40 FC
04B4 82 FA 46
04B7 75 D7
04B9 EB 88 00
04BC B0 2D
04BE EB A7 FC
04C1 EB FD FB
04C4 EB 5E FC
04C7 33 D8
04C9 8B 16 B6 01
04CD BB F7
04CE AD
04D0 3C 0D
04D2 74 66
04D4 82 FC 0D
04D7 74 66
04D9 BF F3 06
04DC B9 20 00
04DF 0E
04E0 07
04E1 F2
04E2 AD
04E3 75 5A
04E5 8A E9
04E7 80 E1 0F
04EA BB 01 00
04ED D3 C0
04EF 85 C3
04F1 75 33
04F3 OB D8
04F5 OB D0
04F7 F6 C5 10
04FA 75 02
04FC 33 D0
04FE 8B FE
0500 1E
0501 07
0502 EB 17 FC
0505 EB C6
0507
0507 2E
0508 AD
0509 EB 5C FC
050C 8A C4
050E EB 57 FC
0511 B0 3D
0513 EB 52 FC
0516 8B 17
0518 43
0519 43
051A EB 2F FC
051D EB 53 FC
0520 EB 50 FC
0523 E2 E2
0525 C3
0526
0526 8B 44 46
0529

MOV SI,RECTAB
MOV BX,AXSAVE
MOV CX,8
CALL DISPREGLINE
CALL CRLF
MOV CX,5
CALL DISPREGLINE
CALL BLANK
CALL DISPFLAGS
JMP CR LF

FLAG:
CMP DL,"F"
JNZ BADREG
CALL DISPFLAGS
MOV AL,"_"
CALL OUT
CALL INBUF
CALL SCANL
XOR BX,BX
MOV DX,[FSAVE]

GETFLG:
MOV SI,DI
LODW
CMP AL,13
JZ SAVCHG
CMP AH,13
JZ FLGERR
MOV DI,FLAGTAB
MOV CX,32
PUSH CS
POP ES
REPNE SCANL
JNZ FLGERR
MOV CH,CL
AND CL,OFH
MOV AX,1
ROL AX,CL
TEST AX,BX
JNZ REPFLG
OR BX,AX
OR BX,AX
TEST CH,16
JNZ NEXFLG
XOR DX,AX

NEXFLG:
MOV DI,SI
PUSH DS
POP ES
CALL SCANL
JP GETFLG

DISPREGLINE:
SEG CS
LODW
CALL OUT
MOV AL,4AH
CALL OUT
MOV AL,"_"
CALL OUT
MOV DX,[BX]
INC BX
INC BX
CALL OUT16
CALL BLANK
CALL BLANK
LOOP DISPREGLINE
RET

REPIFLG:
MOV AX,4600H+"D" ;DF ERROR

FERR:
ERR: CALL SAVCHG
052E CALL OUT
052F C4 MOV AL, AH
0530 CALL OUT
0531 BE 07 MOV SI, ERRMES
0532 3B FE JMP PRINT
0533 SAR
0535 MOV [FSAVE], DX
0536 C3 RET
0538 FLGERR:
053F B8 42 MOV AX, 4600H+"B" ; BF ERROR
0540 EB E5 JP FERR
0544 DISPFLAGS:
0545 BE E3 MOV SI, FLAGTAB
0546 B9 00 MOV CX, 16
0547 B6 01 MOV DX, [FSAVE]
0548 DFLAGS:
0549 2E SEG CS
054A AD LODW
0550 8E D1 MOV AX, [SI+30]
0551 E2 JP DX
0552 04 JC FLAGSET
0553 2E SEG CS
0554 44 MOV AX, [SI+30]
0555 08 OR AX, AX
0556 7A 0B CALL NEXTFLG
0557 EB 09 CALL OUT
0558 C4 MOV AL, AH
0559 E8 04 CALL OUT
055A EB 0C CALL BLANK
055B E2 5D LOOP DFLAGS
055D C3 RET
0560 ; Trace l instruction or the number of instruction specified
0561 ; by the parameter using 8086 trace mode. Registers are all
0562 ; set according to values in save area
0563 TRACE:
0564 8E AF CALL SCAP
0565 98 EB CALL HEXIN
0566 01 00 MOV DX, 1
0567 EB 02 JC STOCNT
0568 04 MOV CX, 4
0569 00 CALL GETHEX
056A 89 B9 CALL GETEOL
056B 02 STEP:
056C C7 06 00 MOV [BRKCNT], 0
0570 E8 00 01 CALL GETEOL
0571 E7 00 00 OR B, [FSAVE+1], 1
0572 E7 00 00 EXIT:
0573 C7 06 MOV [12], BREAKFIX
0574 0C 0D MOV [14], CS
0575 0E 0D MOV [4], REENTER
0576 0E 0D MOV [6], CS
0577 0C 0E DF D1 05 MOV [64H], REENTER
0578 0E 6D 66 MOV [66H], CS
0579 0C 0E 6D MOV SP, STACK
057A 58 CALL AX
057B 5B POP AX
057C 59 POP BX
057D 5A POP CX
057E 5D POP DX
057F 5D POP BP
0580 5E POP SI
0581 5F POP DI
0582 07 POP ES
05B8 07          POP  ES
05B9 17          POP  SS
05BA 8B 26 A4 01  MOV  SP,[SSAVE]
05BE FF 36 B6 01  PUSH  [FSAVE]
05C2 FF 36 B2 01  PUSH  [CSSAVE]
05C6 FF 36 B4 01  PUSH  [IPSSAVE]
05CA BE 1E AC 01  MOV  DS,[DSSAVE]
05CE CF          IRET

STEP1: JP STEP

;Re-entry point from breakpoint. Need to decrement instruction
;pointer so it points to location where breakpoint actually
;occurred.

BREAKFIX:
XCHG  SP, BP
DEC  [BP]
XCHG  SP, BP

;Re-entry point from trace mode or interrupt during
;execution. All registers are saved so they can be
;displayed or modified.

REENTER:
SEG  CS
MOV  [SSAVE+SEGDIF], SP
SEG  CS
MOV  [SSSAVE+SEGDIF], SS
XOR  SP, SP
MOV  SS, SP
MOV  SP, RSTACK
PUSH  ES
PUSH  DS
PUSH  SI
PUSH  BP
DEC  SP
DEC  SP
PUSH  DX
PUSH  CX
PUSH  BX
PUSH  AX
PUSH  SS
POP  DS
MOV  SP, [SSAVE]
MOV  SS, [SSSAVE]
POP  [IPSSAVE]
POP  [CSSAVE]
POP  AX
AND  AH, OFEH
MOV  [FSAVE], AX
MOV  [SSAVE], SP
PUSH  DS
POP  ES
PUSH  DS
POP  SS
MOV  SP, STACK
MOV  [64H], INT
MOV  AL, 2OH
OUT  BASE+2
EI
UP
CALL  CRLF
CALL  DISPRG
DEC  [TCOUNT]
JNZ  STEP1

ENDGO:
MOV  SI, BPTAB
MOV  CX, [BRKCNT]
JCXZ  COMMP
CLEARBP:
MOV DX,[SI+BPLEN]
LODW
PUSH AX
CALL GETSEG
MOV ES,AX
MOV DI,DX
POP AX
STOB
LOOP CLEARBP

COMJMP: JMP COMMAND

; Input from the specified port and display result

INPUT:
MOV CX,4 ; Port may have 4 digits
CALL GETHEX ; Get port number in DX
INB DX ; Variable port input
CALL HEX ; And display
JMP CRLF

; Output a value to specified port.

OUTPUT:
MOV CX,4 ; Port may have 4 digits
CALL GETHEX
PUSH DX ; Save while we get data
MOV CX,2 ; Byte output only
CALL GETHEX
XCHG AX,DX ; Output data in AL
POP DX ; Port in DX
OUTB DX ; Variable port output
RET

; Jump to program, setting up registers according to the
; save area. Up to 10 breakpoint addresses may be specified.

GO:
MOV BX,LINBUF
XOR SI,SI

GO1:
CALL SCANP
JZ EXEC
MOV CX,5
CALL GETHEX
MOV [BX],DX
INC BX
INC BX
INC SI
CMP SI,BPMA+1
JNZ GO1
MOV AX,5000H"b" ; BP ERROR
JMP ERR

EXEC:
MOV [BRKCNT],SI
CALL GETEOI
MOV CX,SI
MOV CXZ NOBP
MOV SI,BPAB

SETBP:
MOV DX,[SI+BPLEN]
LODW
CALL GETSEG
MOV DS,AX
MOV DI,DX
MOV AL,[DI]
MOV B,[DI],OCCH
PUSH ES
POP DS
MOV [SI-2], AL
LOOPE SETBP
NOBP:
MOV [TCOUNT], 1
JMP EXIT

; Console input interrupt handler. Used to interrupt commands
; for programs under execution (if they have interrupts
; enabled). Control-S causes a loop which waits for any other
; character to be typed. Control-C causes abort to command
; mode. All other characters are ignored.

INT:
PUSH AX ; Don't destroy accumulator

; Output End-of-Interrupt commands to slave 8259A. This
; wouldn't be necessary if Automatic End of Interrupt mode
; worked like it was supposed to!

; MOV AL, 20H
OUT BASE + 2

; Get interrupting character
IN DATA

; ASCII has only 7 bits
AND AL, 7FH

; Check for Control-S
CMP AL, "S" "~@
JNZ NOSTOP

; Wait for continue character
CALL IN

; Control-C
CMP AL, "C" "~@
JZ BREAK

; Just ignore interrupt - restore AX and return
POP AX
IRET

BREAK:
CALL CRLF
JMP Command

REGTAB:
DB "AXBXCXDSFSPSIDIDSESSSISIPPC"

44 58 53 42 50
33 49 44 49 44 53
45 53 53 43 53
49 50 50 43

REGDIF: EQU AXSAVE-REGTAB

; Flags are ordered to correspond with the bits of the flag
; register, most significant bit first, zero if bit is not
; a flag. First 16 entries are for bit set, second 16 for
; bit reset.

FLAGTAB:
DW 0
DW 0
DW 0
DW 0
DB "OY"
DB "DN"
DB "ET"
DB "NC"
DB "ZR"
DB "AC"
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0
DB 0

- 29 -
;Initialization table. First byte of each entry is no. of bytes to output to the corresponding port. That many initialization bytes follow.

INITTABLE:
;Port BASE+0 - Master 8259A. Initialization Command Word (ICW)
;One sets level-triggered mode, multiple 8259As, require ICW4.
    DB 1
    DB 19H

;Port BASE+1 - Master 8259A. ICW2 sets vector base to 10H
;ICW3 sets a slave on interrupt input I; ICW4 sets buffered
;mode, as a master, with Automatic End of Interrupt, 8086
;vector; Operation Command Word (OCW) One sets interrupt
;mask to enable line 1 (slave 8259A) only.
    DB 4
    DB 10H,2,0FH,0FDH

;Port BASE+2 - Slave 8259A. ICW1 sets level-triggered mode,
multiple 8259As, require ICW4.
    DB 1
    DB 19H

;Port BASE+3 - Slave 8259A. ICW2 sets vector base to 18H
;ICW3 sets slave address as I; ICW4 sets buffered mode,
as slave, with Automatic End of Interrupt (which doesn’t
;work in slaves), 8086 vector; OCW1 sets interrupt mask
to enable line 1 (serial receive) only.
    DB 4
    DB 18H,1,0BH,0FDH

;Port Base+4 - 9513 Data. 9513 has previously been set
;up for Counter 5 mode register with auto increment. Thus
;mode is set to 0B3H, which is no gating, count down repetitively
;in binary, with output toggle. Load register is set to
;0007H, and Hold register is set to 0006H. Thus we
;alternately divide by 7 and 6, which is divided by 2 by
;the output toggle, thus providing a square wave of
;4 MHz/13 = 307.7 kHz, which divided by 16 in the 8251A
;provides 19,230 baud (0.16% high).
    DB 6
    DB 63H,0BH,7,0,6,0

;Port BASE+5 - 9513 Control. Load and arm counter 5,
;enabling baud rate generation. Then select counter
;5 mode register, in case baud rate wasn’t right.
    DB 2
    DB 70H,5

;Port BASE+6 - 8251A Data. No initialization to this port.
    DB 0

;Port BASE+7 - 8251A Control. Since it is not possible to
;know whether the 8251A next expects a Mode Instruction or
;a Command Instruction, a dummy byte is sent which could
;safely be interpreted as either but guarantees it is now
;expecting a Command. The command sent is Internal Reset
;which causes it to start expecting a mode. The mode sent
;is for 2 stop bits, no parity, 8 data bits, 16K clock.
;This is followed by the command to error reset, enable
;transmitter and receiver, set RTS and DTR to +12V.
    DB 4
    DB 0BH,77H,0CEH,37H

HEADER: DM 13,10,10,"SCP 8086 Monitor 1.5",13,10
SYNERR: DB ""'
ERRMES: DM "Error",13,10'
BAMES: DM 8,32,8

; Disk boot. Select one of the following routines by
; setting the equates at the start of this program.

;*****************************************************************************

; Boot for Cromemo 4FD disk controller with either
; large or small disks. Loads track 0, sector 1 into LOAD.

IF CROMEMCO4FDC

DISK: EQU 30H

MOV AL,1
OUT 2 ; Reset 4FD serial I/O
MOV AL,84H ; and set for 300 baud
OUT 0
MOV AL,7FH
OUT 4
MOV DL,21H

RETRY:
MOV AL,0D0H
OUTB DISK

READY:
INB DISK
ROR AL
JC READY
XOR DL,10H
MOV AL,DL
OUTB DISK+4
MOV DI,LOAD
MOV AL,12
OUTB DISK

HOME:
INB DISK+4
ROR AL
JNC HOME
INB DISK
AND AL,98H
JNZ RETRY
MOV AL,1
OUTB DISK+2
MOV CX,80H
MOV AL,DL
OR AL,80H
OUTB DISK+4
MOV AL,8CH
OUTB DISK

READ:
INB DISK+4
ROR AL
JC DONE
INB DISK+3
STOB
LOOP READ

WSTAT:
INB DISK+4
ROR AL
JNC WSTAT

- 31 -
07CB
07CB E4 30
07CD 24 9C
07CF 75 B4
07D0 INB
07D1 DISK
07D1 AND
07D1 AL,9CH
07D1 JNZ
07D1 RETRY
07D1 ENDF
07D1
07D0 ENDIF
07D0 POP
07D0 DS
07D0 ;Successful read
07D1 MOV
07D2 [CSSAVE],0
07D3 MOV
07D4 [IPSAVE],LOAD
07D5 POP
07D6 DI
07D7 MOV
07D8 [DSAVE],0
07D9 POP
07DA DS
07DB DI
07DE CALL
07E0 GO
07E1
0777
0777 Error Count = 0
0777
0777 ;Boot for North Star disk, single density.
0777 ;Loads track 0, sector 0 into address LOAD
0777 ;Bug in North Star boot fixed 5-26-81.
0777
0777 IF NORTHSTARS
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
0777 PUSH DS
0778 88 B8 FE MOV AX,0FE8B
0779 B8 D8 MOV DS,AX
077A 01 00 MOV AL,[SEL]
077B 14 00 MOV CX,20
077C 78 E8 19 00 CALL SECTOR
077D 78 E2 F8 LOOP MOTOR
077E 78 CHKTRK:
077F 86 F6 06 1C 00 01 TEST B,[STPOUT],1
0780 73 1B JNZ ONTRACK
0781 90 A0 09 00 MOV AL,[STP1]
0782 92 D4 0A AAM
0783 94 A0 08 00 MOV AL,[STP2]
0784 97 E8 05 00 CALL SECTOR
0785 9A E8 02 00 CALL SECTOR
0786 99 EB 00 00 MOV AL,[SEC] ; Reset sector flag.
0787 7A2 SECLP:
0788 86 F6 06 10 00 80 TEST B,[NOP],80H ; Wait for sector flag.
0789 74 F9 JZ SECLP
078A 9 C3 RET
078A ONTRACK:
078B A0 00 02 MOV DL,LOAD
078C B9 18 01 MOV CX,280
078D B5 50 00 MOV BX,RD+NOP
078E GETSEC:
078F 88 E9 FF CALL SECTOR
0790 86 F6 06 30 00 0F TEST B,[BST+NOP],OFH ; Test for sector zero.
0791 B7 75 6F JNZ GETSEC
0792 B F6 06 10 00 04 GETSYNC:
0793 81 E1 F9 TEST B,[NOP],4
0794 LOOPZ GETSYNC
0795
- 32 -
07C4 74 B4
07C5 B9 00 01
07C9 32 D2
07CB D5 0A
07CD READ:
07CD 8A 07
07CF AA
07D0 32 D0
07D2 D0 C2
07D4 D5 0A
07D6 E2 F5
07D8 8A 07
07DA 3A C2
07DC 75 CC
07DE 1F
07DF ENDIF
07DF C7 06 B2 01 00 00
07E5 C7 06 B4 01 00 02
07EB 5F
07EC E9 7B FE
07EF

Error Count = 0

;************************************************************************

; Boot for Tarbell disk controllers. Load track 0, sector 1 into LOAD.

IF TARBELL

DISK: EQU 78H

DCOM:
JP RETRY

OUTB DISK
MOV AL,50

HOLD:
DEC AL
JNZ HOLD
RET

RETRY:
MOV AL,O0H
CALL DCOM

READY:
INB DISK
ROR AL
JC READY
MOV DI,LOAD
MOV AL,0EH ;Home command @ 10ms/track
CALL DCOM
INB DISK+4
INB DISK
AND AL,98H
JNZ RETRY
MOV AL,1
OUTB DISK+2
MOV CX,80H
MOV AL,8CH
CALL DCOM

READ:
INB DISK+4
ROL AL
JNC DONE
INB DISK+3
STOB LOOP READ
WSTAT:
INB DISK+4
ROL AL
JCB WSTAT
DONE:
INB DISK
AND AL,9CH
JNZ RETRY
;Successful read
MOV [CSSAVE],0
MOV [IPSAVE],LOAD
POP DI
JMP GO

;******************************************************************************

IF OTHER

;User may insert customized disk boot here. All
;registers are available, stack pointer is valid
;and interrupts are enabled. Stqck should be at
;same level on fall-through to code below. Last
;address available is 07DF hex.

ORG 7E0H ;Simulate boot of maximum length

ENDIF

;******************************************************************************

;Successful read
MOV [CSSAVE],0
MOV [IPSAVE],LOAD
POP DI
JMP GO

Error Count = 0