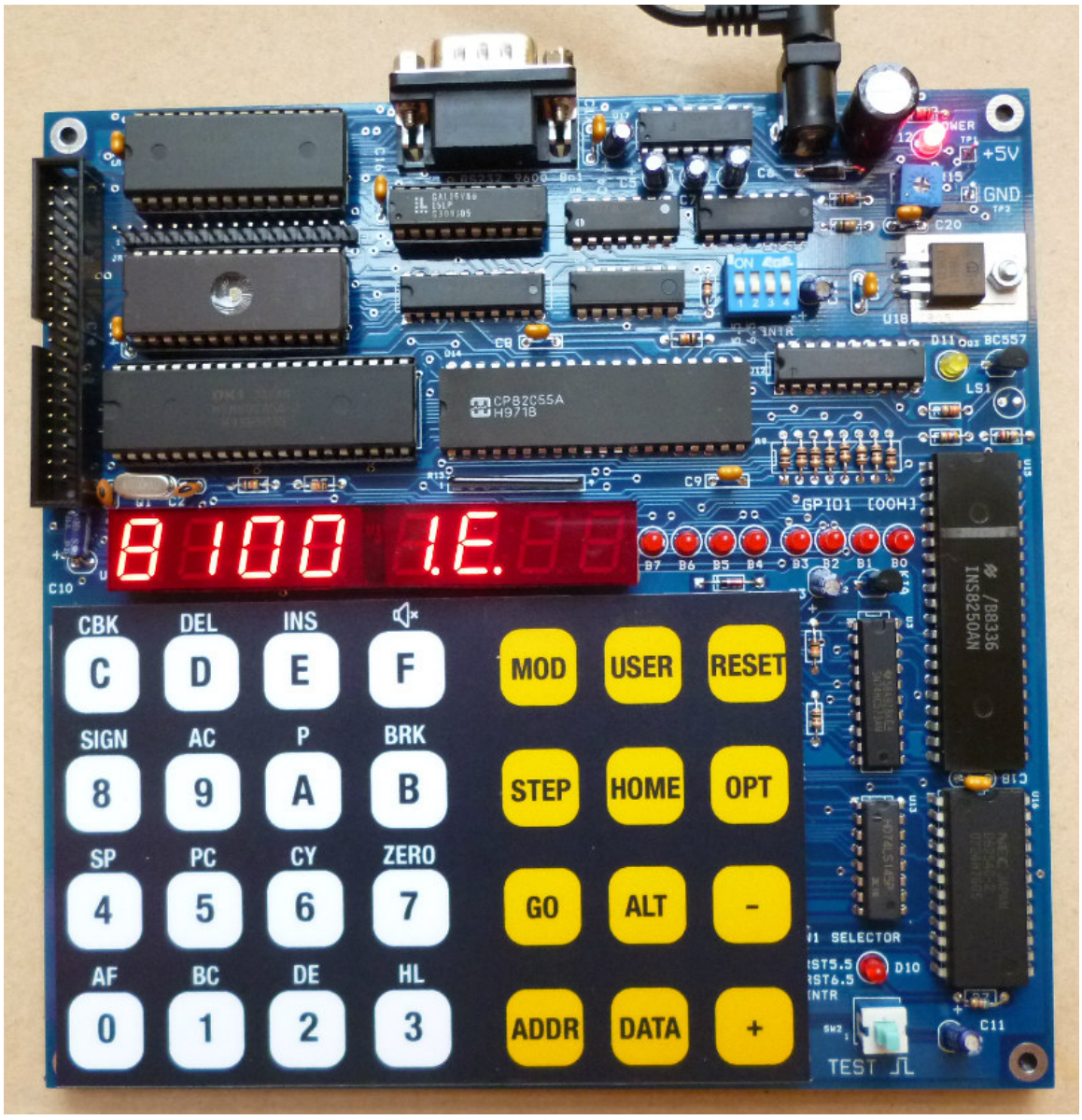
8085 Microprocessor Kit
User's Manual



Rev1.0, December, 2016

Contents

Overview……..…….3

Hardware Features – Software Features

Getting Started…..……5

AC Adapter – LED Display and Keypad – RESET – ADDR – DATA – INC – DEC – HOME – ALT – GO – STEP – MOD – Program 1 – Program 2 – Program 3

Connecting Terminal..…….……18

Command ‘A’ – Command ‘C’ – Command ‘D’ – Command ‘E’ – Command ‘F’ – Command ‘H’ – Command ‘I’ – Command ‘J’ – Command ‘K’ – Command ‘L’ – Command ‘M’ – Command ‘N’ – Command ‘Q’ – Command ‘R’ – Command ‘S’ – Command ‘W’ – Command ‘SPACE BAR’

Hardware…….27

CPU – Memory – GPIO – Programmable Port 8255 – Programmable Counter 8254 – Headers and Connectors – Interrupt Test Button – Technical Specifications – Monitor Call Number-NVRAM Bootable

Appendix A LCD Driver Routines
Appendix B SCAN Keyboard and Display Subroutines
Appendix C UART Driver Routines
Appendix D Using NVRAM Bootable
Appendix E Machine Code and 8085 Instructions
Appendix F Hardware Schematic
Appendix G Monitor source code listing

Overview

The 8085 Microprocessor kit is a low-cost single board computer designed for self-learning the popular 8085 Microprocessor. The kit enables studying from low level programming with direct machine code entering to high level programming with PC tools easily. A nice feature, single-step running, helps students learn the operation of microprocessor instructions quickly and clearly. The user registers provide simple means to verify the code execution. Using a PC as the terminal, the kit can receive the Intel hex file and disassemble the machine code into 8085 instructions.

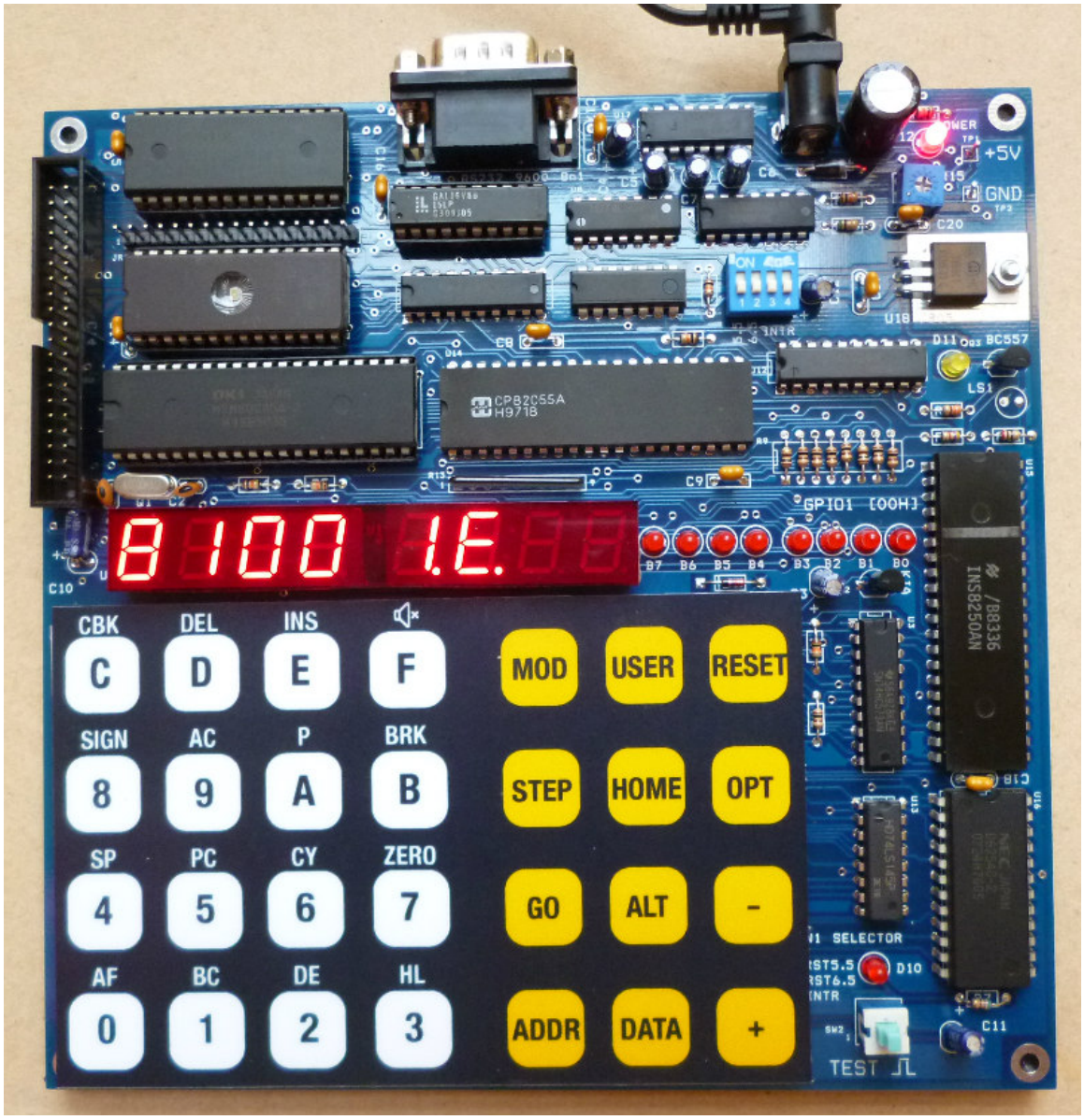


Figure 1: Components layout.

Hardware Features:

- CPU: Mitsubishi M5M80C85AP-2 @4MHz
- Memory: 32kB Monitor ROM and 32kB SRAM
- Simple I/O Port: 8-bit GPIO
- Programmable Ports: 8255 chips
- Programmable Counter: 8254
- UART: NS8250 UART chip

Onboard I/O devices:

- 6-digit seven segment super bright LED
- 28-keypad
- 8-bit dot LED indicates status of GPIO1
- Speaker
- Direct BUS interface text LCD
- Serial Interface: RS232C 9600bit/s 8-data bit no parity one stop bit
- +5V Power Supply: voltage regulator with input protection
- 40-pin header for CPU bus
- Counter timer 8254
- onboard logic probe power supply
- Test button for single pulse generation to the interrupt pins
- Brown-out Protection

Software Features:

- Enter the machine code in hexadecimal
- Single-step execution
- Examine and modify user registers
- Run user code with software break-point
- Insert and Delete byte
- Built-in LCD drivers
- Download Intel Hex file
- Disassemble machine code into 8085 instructions
- Display user registers and disassemble the instruction after single-stepping

Getting Started

AC Adapter

The kit requires DC power input to operate. The input voltage accepts from +7.5V to +12V. You may find any AC-to-DC adapter having DC jack with polarity as shown in Figure 2. The board has protection diode to prevent wrong polarity. If your adapter's jack has different polarity, when plug it to the board, no power will be supplied. The center pin is positive.

Figure 2: Polarity of DC jack.

When power up the board, the 8085 fetches
the instruction from the memory at location 0000H. The location from 0000H to 7FFFH or 32kB is ROM space. It contains the monitor program. The monitor program enables us to enter 8085 instruction using HEX digit into the RAM. We can let the 8085 RUN our program easily using monitor key GO.

When the board was powered up, the cold message running text 8085 will show on 7-segment LED and the onboard dot LED will turn on and the speaker will sound beep. The HOME location is pointed to RAM at address 8100H. The data LED will display the content at 8100H.

LED Display and Keypad

The kit has 6 digits 7-segment LED and 28 tact switches keypad.

Four digits is used for displaying the memory address and user registers contents. Two digits “DATA” is for displaying the 8-bit data byte at address shown in the left-hand. The dot indicator indicates the current mode of HEX digit entering. Figure 3 shows the memory location 8100 has an 8-bit data, 1E. The dot indicates the current mode is data entry. Typing Hex key will insert hex digit into data memory.

Figure 3: ADDRESS and DATA fields.

Keypad has two groups: the left-hand is 16-hex key 0-F and the right-hand is 10-function key. The hex key also has alternate functions when used with ALT key.

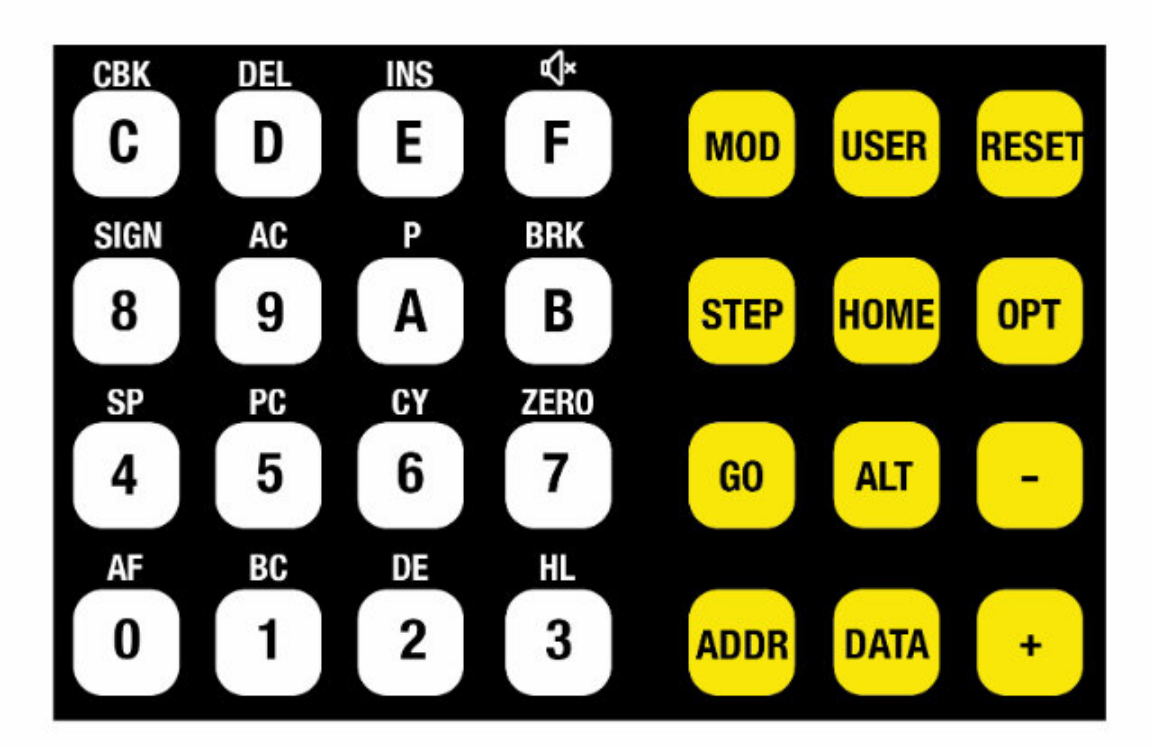


Figure 3: Keyboard layout, HEX and Function Keys.

The functions key are:

RESETis hardware reset. Press reset will force the CPU begins execution the ROM monitor at address 0000H. (The reset out signal which is active high also feed to reset pins of the UART and the 8255 PPI).

ADDR changes current mode to ADDRESS entry mode. The dot indicator will move to ADDRESS filed.

DATA changes current mode to DATA entry mode. The dot indicator will move to DATA filed.

Key + increments current address by one. The content of new address will show in data field LED.

Key - decrements current address by one. The content of new address will show in data field LED.

HOME brings home address back to current display. The home address is 8100H.

ALT enables alternate functions that used with HEX key. We can press ALT followed with HEX key. The Alternate functions are described below.

ALT 0 displays user register AF. The Accumulator and Flag registers. Contents of accumulator is high byte and Flag is low byte.

ALT 1 displays user register BC.

ALT 2 displays user register DE.

ALT 3 displays user register HL.

ALT 4 displays user register SP.

ALT 5 displays user register PC.

ALT 6 displays CARRY flag.

ALT 7 displays ZERO flag.

ALT 8 displays SIGN flag.

ALT 9 displays HALF CARRY flag.

ALT A displays PARITY flag.

ALT B sets break address.

ALT C clears break address.

ALT D deletes one of the current location and shifts the next byte UP.

ALT E inserts one byte and shifts the next byte DOWN.

ALT F Toggle beep ON/OFF.

GO forces CPU to jump from monitor program to user program at current address.

STEP executes one instruction at address shown in current display.

MOD modifies the user registers. It was used together with ALT 0-5.

User registers are memory spaces in RAM prepared for saving and loading to the CPU registers when the CPU jump from monitor program to user program and back to the monitor program. It is useful for program debugging. We will learn how to use them easily in the program testing section.

Entering the program into RAM and Run it

Test Program 1

Let us learn how to use hex keypad to help enter the computer code to memory and test run it. Suppose we want to write the program that displays the content of the accumulator using onboard gpio LED. The kit has 8-bit dot LED tied to the 8-bit output port. Logic '1' presents at a given bit will make the LED ON. Logic '0' makes the LED OFF. We will write the small program that shows the accumulator content.

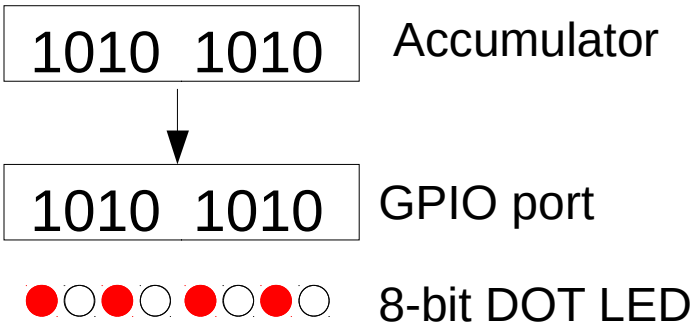


Figure 4: Writing the Accumulator content to gpio PORT at location 0.

Our program is,

```
main:  inr a ; increment accumulator
       out 0 ; write to port 00
       jmp main ; jump back to main
```

We see that the program has only three instructions, i.e., inr a, out 0 and jmp main.

The program was written using 8085 instructions. To test our program, we must translate above program into the 8085 hex code. This can be done easily with hand-code assembly. See Appendix E for machine code of the instructions.

Since we will write the machine code to the memory for testing, so the space must be RAM. We must know the memory allocation. Figure 5 shows the memory space allocation. We see that the board provides begin address for user program at 8100H. Some of the locations from 8000H to 803CH are reserved for interrupts vectors. The RAM locations from F000H to FFFFH are used by monitor program.

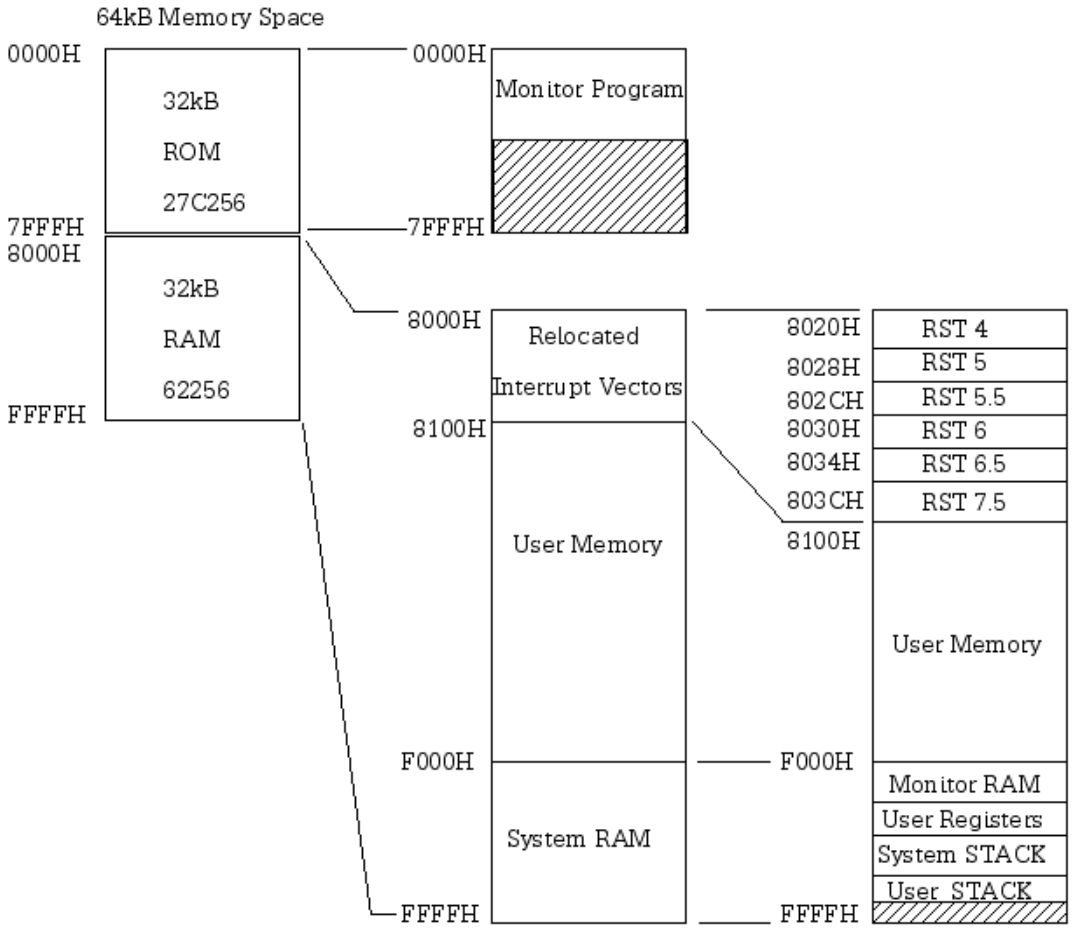


Figure 5: Memory space allocation.

Thus we can place our machine code started at location 8100. After translation we get the code for each instruction as shown below.

```
8100 3C          main:  inr a
8101 D300               out 0
8103 C30081             jmp main
```

The 1st instruction, inr a, increments the accumulator by one. It has one byte machine code 3C. This byte will be placed at location 8100.

The 2nd instruction, out 0, write accumulator content to the gpio port at location 00 has two bytes machine code, D3, 00. D3 is the instruction OUT and 00 is port location.

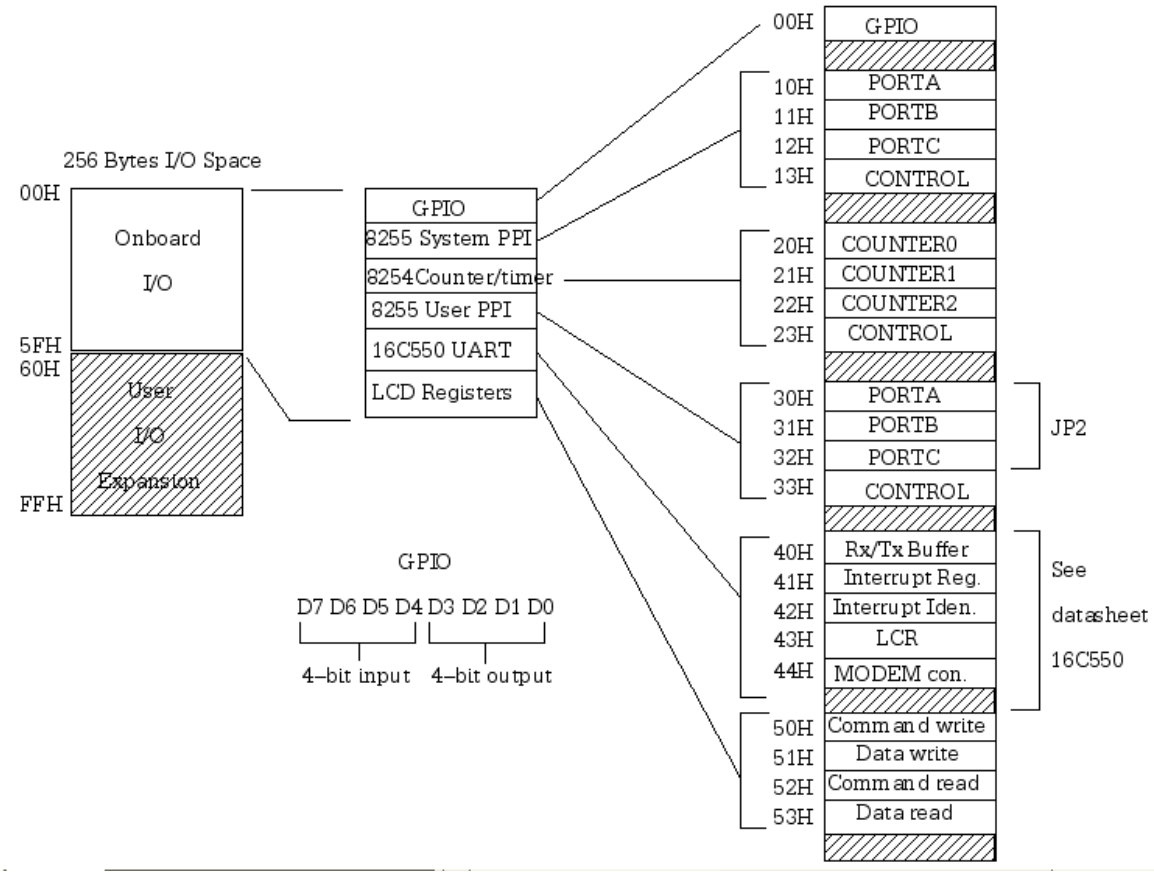


Figure 6: I/O space allocation, User PPI is not available for this version.

The 3rd instruction, jmp main, jump back to location 8100 has three bytes machine code, C3,00,81. C3 is the JMP, and 8100 is location to be jump (Intel places low byte to low address and high byte to high address).

Above program has only 6 bytes. We can enter such code into RAM easily using HEX key. Here is the byte sequence from address 8100 to 8105.

ADDRESS	DATA
8100	3C
8101	D3
8102	00
8103	C3
8104	00
8105	81

Now enter the code into memory address 8100.

Step 1 Press RESET, the address display will show 8100 and the data LED will show its contents.

The current mode will be data entry. We can swap entry mode for hex key between address and data by pressing key ADDR or DATA. The DOT indicator will swap between ADDR mode and DATA mode.

To enter a byte to this location, press HEX key 3 and key C.

The 3C byte will enter to address 8100.

Step 2 Press key + to increment address.

The address display will show 8101. Then repeat step1 until 81 byte was entered to address 8105.

You can use key + or key - to check the hex code, you can modify it easily in DATA entry mode.

We will begin set the value to user Accumulator beforehand. It will clear the user register A to zero.

Press key ALT, 0/AF, display will show the content of user Accumulator and Flag register.

Press key MOD, then key 0,0,0,0. The AF will be 0000.

Press key HOME, this brings current location to 8100.

Press key STEP, the display will show next instruction to be executed at address 8101. We can examine the content of AF by key ALT, 0/AF. We see that now Accumulator is 01.

Press key STEP again, the 01 will send to LED onboard GPIO. This is the content of the accumulator after increment instruction.

The next instruction, JMP 8100 will be executed.

We can keep press key STEP, we will see every time the instruction out 0 was executed, the value of accumulator will write to the GPIO LED.

It works! This demonstration how STEP key helps running the program single instruction.

Instead of execution one instruction using single step, we can run the program without stopping for each instruction. We will try with key GO.

Now press HOME to bring current location to 8100, press RUN.

What happen to the LED?

Did you see the LED counting?

Should it be counting up?

There are two methods of program running. First is to use single stepping. This kind executes only one instruction at a time when we press STEP key. We can learn the operation easily with user registers. The monitor program loads the contents of user registers to the CPU registers beforehand, after the instruction has been executed, the contents of CPU registers will then be saved back to the user registers. Thus we can examine the result after the instruction has been executed.

But above program, when we try with key GO, the CPU will jump from monitor program to user program and never get back to monitor program. Since the instruction JMP 8100 will jump back to 8100 forever. We see that the number incrementing in the accumulator will be very fast.

How can we make the speed of counting slower? We can just simply add the job that uses CPU time. See below program.

```
        org 8100h ; begin of code
main:    inr a ; increment accumulator
         out 0 ; write to port 00

; add the simple delay using register pair DE

         lxi d,1050h ; load 16-bit constant to DE
delay:   dcr e       ; decrement E
         jnz delay  ; jump to delay location if E != 0
         dcr d      ; decrement D
         jnz delay  ; jump to delay location if D != 0

         jmp main   ; done, jump back to main again
```

I suppose now you can translate the instruction into the machine code. The first mnemonic, ORG is not 8085 instruction. It is the assembler directive that tells the assembler program to place the hex code begins at location 8100. We will learn using assembler when using PC tools on later.

The portion of inserted code is bolded letters. We see that the method of time delay is just to let the CPU counts the value in register D and register E. Counting is done by instruction DCR E, decrement register by one for register E and D. The JNZ, jump to specified location when ZERO flag is not set. That means if the content of register E or D is not ZERO, it will jump back to decrement again. Until both are ZERO, the CPU will continue execute the next instruction.

Here is the translation from instructions to machine code.

```
                        org 8100h ; begin of code
8100 3C      main:      inr a ; increment accumulator
8101 D300               out 0 ; write to port 00

; add the simple delay using register pair DE

8103 115010      lxi d,1050h ; load 16-bit constant to DE
8106 1D      delay:  dcr e         ; decrement E
8107 C20681          jnz delay  ; jump to 81060 if E != 0
810A 15               dcr d       ; decrement D
810B C20681           jnz delay  ; jump to 8106 if D != 0

810E C30081           jmp main   ; jump back to main
                         again
```

This program has 17 bytes . We can enter the code into RAM from 8100 to 8110 easily.

ADDRESS	DATA
8100	3C
8101	D3
8102	00
8103	11
8104	50
8105	10
8106	1D
8107	C2
8108	06
8109	81
810A	15
810B	C2
810C	06
810D	81
810E	C3
810F	00
8110	81

After finished entering the code, press HOME to bring current RAM location to 8100. Then press key GO.

What happen to the onboard LED?

Can we change the speed of counting? How?

To stop running, press RESET key. You can modify the initial value of register DE, 1050 to whatever you want to speed up or slow down.

Test Program 2

This program shows how to use key GO to force CPU jump from monitor program to user program.

```
8100 1E02        main:    mvi e,2
8102 CF                   rst 1
8103 C30081              jmp main
```

This program has only 6 bytes i.e., 1E, 02, CF, C3, 00, 81. Enter the code, and press key HOME, GO.

We will see the cold message repeat running on the display. RST 1 having machine code CF is the method that used to call built-in monitor functions. Register E is monitor call number.

To stop program 2 running, we must press RESET key.

Test Program 3

We can test the program with software breakpoint. The instruction RTS 7 having machine code FF returns control back to monitor program and saves the contents of CPU registers to user registers. We can check the result in user registers easily.

Here is the program that adds two BCD numbers 19H and 02H. The result will be 21H.

```
8100 3E19      mvi a,19h ; load accumulator with 19h
8102 0602      mvi b,2 ; load register B with 02
8104 80        add b ; add register B to accumulator
8105 27        daa ; adjust result to BCD
```

```
8106 FF        rst 7 ; jump back to monitor
```

After enter the code, you can run it with key GO. Check the result in Accumulator with ALT 0.

For small program, we can place the RST 7 to the end of the program.

However for long program, sometime we may need to check at a given location, the board also provides tool that helps inserting the RST 7 instruction to the specified location. This tool is called set break point. Suppose we want to verify the result after add b instruction. We can set break point at location 8105 by setting the address to 8105 with key ADDR 8,1,0,5. Then press ALT B, the display will show this address was set breakpoint.

Press HOME and GO, check user AF with ALT 0, we see that after addition, the result in Accumulator is 1B. To clear this break address, press ALT C. The display will show current address 8105. The code 27 will be restored back to address 8105. We can continue execution, press GO, and check result in AF again, we will get 21. This the correct BCD number from the addition of 19+02=21.

Connecting Terminal

The kit provides RS232 port for connecting the terminal. The ROM monitor contains ASCII commands when using UART to connect a terminal. The UART drivers and serial commands are automatically configured when UART chip was inserted. Communication format is 9600 bit/sec, 8 data bit, no parity and one stop bit. We can use PC running VT100 terminal emulation. You may download free terminal program, teraterm from this URL,
http://ttssh2.sourceforge.jp/index.html.en

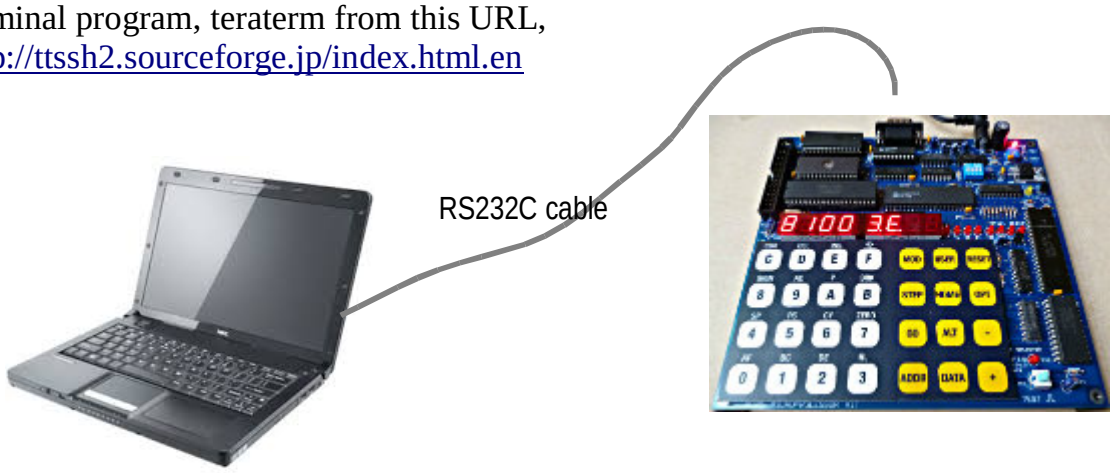


Figure 7: Connecting PC running teraterm and kit with RS232 cross cable.

There is no need to switch between standalone mode and terminal mode. Both commands using keypad or terminal commands are working concurrently.

When press reset the prompt appears on screen.

```
MTK-85 8085 MICROPROCESSOR TRAINING KIT (? HELP)

8100>
```

Type ? for help menu listing.

```
MTK-85 8085 MICROPROCESSOR TRAINING KIT (? HELP)

A - ASCII code
C - clear watch variables
D - disassemble
E - edit memory
F - fill constant
H - hex dump
I - i/o address map
J - jump to user program
K - display user STACK
L - load Intel hex file
M - monitor call number
N - new location pointer
Q - quick home location
R - user register display
S - set value to user register
W - watch variables
SPACE BAR - single step
? - help menu

8100>
```

Command ‘A’ prints the hexadecimal code for printable ASCII characters.

Command ‘C’ clears the 16-byte watch variables.

The monitor provides quick access to a16-byte RAM for program testing. The watch variables use RAM space from F000-F00F. Command 'W' prints such memory on screen.

```
8100>
F000 AD FD FC 15 8E 9C DB 4D 4F 19 5F FD EB 3E 8A F5
8100>
```

Command ‘D’ disassembles the machine code into 8085 instructions.

```
8100>disassemble...

8100 3E19       MVI     A,19
8102 0602       MVI     B,02
8104 80         ADD     B
```

```
8105 27          DAA
8106 FF          RST     7
8107 C20681      JNZ     8106
810A 15          DCR     D
810B C20681      JNZ     8106
810E C30081      JMP     8100
8111 62          MOV     H,D
8112 CDF862      CALL    62F8
8115 80          ADD     B
8116 DC1642      CC      4216
8119 A5          ANA     L
811A D3C1        OUT     C1
811C 68          MOV     L,B

811D>
```

Command 'E' examines and modify the data in memory. We can use this command to enter machine code. To view the content, uses Space key and to enter byte, press two digits. To quit just press ENTER.

```
8100>edit memory location = 8100
Enter to quit, SPACE key to view content

ADDR  DATA
8100  [3E]
8101  [19] 01
8102  [06]
8103  [02]
8104  [80] d3
8105  [27] 00
8106  [FF]
8107  [C2]
8108  [06]

8108>
```

Command 'F' fills 8-bit constant to memory. The example shows filling byte 00 to address 9010-9020.

```
9016>Begin address = 9010 End address = 9020 Data = 00
9016>
```

Command 'H' dumps memory. The content of memory from current pointer 9010 to 908F will display in hexadecimal. The ASCII code for each byte will be displayed also. The dot will be displayed for nonprintable ASCII code.

```
9010>

9010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
9020  4D C2 97 CB DA DF A0 BE 9E 73 1A 34 E3 A6 83 4E   M........s.4...N
9030  97 47 81 CE C1 99 98 CB 14 ED 45 DE 35 6A 7C F1   .G........E.5j|.
9040  F0 36 B2 69 CF 1D 90 90 70 F1 73 D8 C1 4F DF 56   .6.i....p.s..O.V
9050  A8 E2 30 84 76 AA C5 18 A7 84 C5 32 81 BF B9 03   ..0.v......2....
9060  8A 13 8C FD 4A 82 B9 99 4E 24 33 9E EB 16 A8 0D   ....J...N$3.....
9070  A9 31 CD B7 BB 4E 8D BE FF 5B 3C 8D EA 5E 4F 7F   .1...N...[<..^O
9080  41 00 89 F3 54 BF EC BF E0 9F 72 CB 7D E8 34 7A   A...T.....r.}.4z

9090>
```

Command 'I' displays onboard I/O address.

```
9090>
00H-0FH onboard 4-bit GPIO, D0-D3=output port
        D4-D7=input port

10H-13H 8255 system PPI, 10H=PORTA, 11H=PORTB, 12H=PORTC,
13H=CONTROL

20H-23H 8254 programmable counter, 20H=counter0, 21H= counter1
22H=counter2, 23H control register

30H-33H 8255 user PPI, 30H=PORTA, 31H=PORTB, 32H=PORTC,
33H=CONTROL

40H-47H C16550 UART registers
9090>
```

Command 'J' jumps from monitor program to user program. The example shows jump to address 9000. The user register displays results after running the code. The RST 7 returns control back to monitor program.

```
9090>jump to address [9006] = 9000


AF=5800 BC=19F4 DE=C256 HL=9504 SP=F098 PC=9007 S=0 Z=0 AC=0 P=0 CY=0
9090>
```

Command 'K' displays user STACK memory. The example below shows running instruction PUSH H.

We first check the user register with command ‘r’. We see that TOP of STACK is F098. After pressing SPACE BAR for single step, the SP is now F096. We can see the contents of STACK memory with command k. The contents of HL was saved in STACK.

```
9000>press r for user register display

AF=5800 BC=19F4 DE=C256 HL=9504 SP=F098 PC=9000 S=0 Z=0 AC=0 P=0 CY=0
9000>press SPACE bar for single step
        9000 E5         PUSH    H

AF=5800 BC=19F4 DE=C256 HL=9504 SP=F096 PC=9001 S=0 Z=0 AC=0 P=0 CY=0
9000>press k for STACK display

ADDR  DATA
F096  [04]
F097  [95]
F098  [C0]

9000>
```

Command ‘L’ loads Intel Hex file to memory. The Assembler and C compiler for 8085 CPU produce standard Intel Hex file. The hex file contains machine code represented by ASCII letters. The example below uses Teraterm to download the hex file. The hex file is ASCII text file. So with the teraterm, we can go to Send File. We can let it show only file with .hex extension by typing *.hex. Then double clicks at the hex file.

The onboard dot LED will indicate downloading is on going. When completed, the report will show number of byte received and print checksum error. If no error it will show 0 errors.

```
9080>load Intel hex file...000005 bytes loaded 0 errors

9080>
```

Command ‘M’ shows monitor call number. Some of common subroutines can be called through RST 1 with function number preloaded in register E.

```
9080>
see input parameters in user manual

1Enn MVI E,function_number
CF   RST 1

00 - demo
```

```
01 - delay
02 - cold_boot
03 - scan
04 - cin
05 - cout
06 - put_str
07 - init_lcd
08 - lcd_ready
09 - clear_lcd
0A - goto_xy
0B - put_str_lcd
0C - put_ch_lcd
0D - demo2

9080>
```

Command 'N' sets new location pointer at prompt. The example sets new pointer to E000 and press 'd' to disassemble.

```
9080>new location = e000
E000>disassemble...

E000 53         MOV     D,E
E001 32DE2A     STA     2ADE
E004 62         MOV     H,D
E005 25         DCR     H
E006 C9         RET
E007 1C         INR     E
E008 43         MOV     B,E
E009 CC4A05     CZ      054A
E00C 2655       MVI     H,55
E00E 67         MOV     H,A
E00F 04         INR     B
E010 F3         DI
E011 0E25       MVI     C,25
E013 54         MOV     D,H
E014 AF         XRA     A
E015 C3F0C3     JMP     C3F0

E018>
```

Command 'Q' sets location pointer at prompt to 9000 and sets user PC to 9000.

```
9000>

AF=5800 BC=19F4 DE=C256 HL=9504 SP=F096 PC=9000 S=0 Z=0 AC=0 P=0 CY=0
```

```
9000>
```

Command 'R' displays user registers contents.

```
9000>

AF=5800 BC=19F4 DE=C256 HL=9504 SP=F098 PC=9000 S=0 Z=0 AC=0 P=0 CY=0
9000>
```

Command 'S' sets value to user registers.

```
9013>set value to user register (enter A for AF) ?
AF=0404 ff00
9013>

AF=FF00 BC=19F4 DE=0434 HL=0534 SP=F096 PC=9006 S=0 Z=0 AC=0 P=0 CY=0
9013>
```

Command 'W' prints watch variables.

```
9013>
F000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Command 'SPACEBAR' executes the instruction at address in user PC. The instruction will show on screen with user registers result after execution.

Suppose we write a program as shown below.

```
      org 9000h
       xra  a

loop: out 0
      mov h,a
      inr h
      push h
      pop d
      mov a,d
      jmp loop

      end
```

Then translate it to machine code file using the Assembler program. Download hex file.

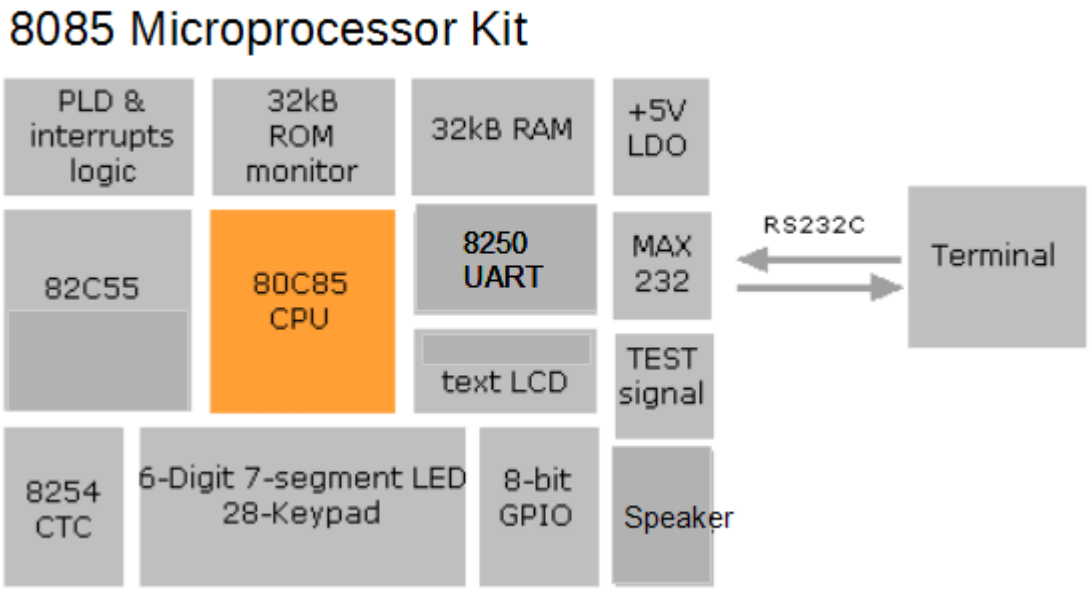
```
9000>load Intel hex file...000011 bytes loaded 0 errors
```

```
9000>disassemble...

9000 AF          XRA     A
9001 D300        OUT     00
9003 67          MOV     H,A
9004 24          INR     H
9005 E5          PUSH    H
9006 D1          POP     D
9007 7A          MOV     A,D
9008 C30190      JMP     9001
900B 00          NOP
900C 00          NOP
900D 00          NOP
900E 00          NOP
900F 00          NOP
9010 00          NOP
9011 00          NOP
9012 00          NOP

9013>print user register with command r

AF=5800 BC=19F4 DE=C256 HL=1234 SP=F098 PC=9000 S=0 Z=0 AC=0 P=0 CY=0
9013>press SPACE key to execute instruction at 9000, we see A=00

        9000 AF          XRA     A

AF=0044 BC=19F4 DE=C256 HL=1234 SP=F098 PC=9001 S=0 Z=1 AC=0 P=1 CY=0
9013>press SPACE key, the content of A will send to GPIO

        9001 D300        OUT     00

AF=0044 BC=19F4 DE=C256 HL=1234 SP=F098 PC=9003 S=0 Z=1 AC=0 P=1 CY=0
9013>press SPACE key, the content of A will copy to H
        9003 67          MOV     H,A

AF=0044 BC=19F4 DE=C256 HL=0034 SP=F098 PC=9004 S=0 Z=1 AC=0 P=1 CY=0
9013> press SPACE key, the content of H will increment by 1
        9004 24          INR     H

AF=0000 BC=19F4 DE=C256 HL=0134 SP=F098 PC=9005 S=0 Z=0 AC=0 P=0 CY=0
9013> press SPACE key, the content SP will decrement by 2
        9005 E5          PUSH    H

AF=0000 BC=19F4 DE=C256 HL=0134 SP=F096 PC=9006 S=0 Z=0 AC=0 P=0 CY=0
9013> press K, to see the content of STACK memory

ADDR  DATA
F096  [34]
F097  [01]
F098  [C0]

9013> press SPACE key, DE will be loaded with top of STACK
        9006 D1          POP     D

AF=0000 BC=19F4 DE=0134 HL=0134 SP=F098 PC=9007 S=0 Z=0 AC=0 P=0 CY=0
9013> press SPACE key, the content of D will copy to A
```

```
    9007 7A         MOV     A,D

AF=0100 BC=19F4 DE=0134 HL=0134 SP=F098 PC=9008 S=0 Z=0 AC=0 P=0 CY=0
9013> press SPACE key, PC will be loaded with 9001
        9008 C30190     JMP     9001

AF=0100 BC=19F4 DE=0134 HL=0134 SP=F098 PC=9001 S=0 Z=0 AC=0 P=0 CY=0
9013> press SPACE key, the content of A will send to GPIO, see LED!
        9001 D300       OUT     00

AF=0100 BC=19F4 DE=0134 HL=0134 SP=F098 PC=9003 S=0 Z=0 AC=0 P=0 CY=0
```

Hardware

A block diagram of the 8085 kit is shown below. For complete hardware schematic, see Appendix D.



CPU

The CPU is the 8-bit Microprocessor, 80C85. The XTAL frequency is 4MHz. The reset signal is generated by RC circuit. The CPU is reset by brownout circuit. In case of power supply is dipped caused by AC supply voltage dropped. The brownout circuit detects VCC, if it is below threshold level, it will reset the CPU.

The brownout condition can be tested by using a variable power supply. To test it, adjust the board VIN from 0-12V slowly and see the CPU can start operating properly.

Memory

The onboard has 64kB memory. The 32kB ROM monitor 27C256 is placed at address 0000-7FFFH. And the 32kB SRAM 62256, is placed at address 8000H-FFFFH.

Some of interrupt vectors are relocated to RAM, so user can write the jump instruction to the location of such interrupt service routine easily. Here is the list of location of interrupts.

8010H	RST 2
8018H	RST 3
8020H	RST 4
8028H	RST 5
802CH	RST 5.5
8030H	RST 6
8034H	RST 6.5
803CH	RST 7.5

Note:

1. RST 7 is used for software breakpoint.
2. RST 1 is used for monitor function call.
3. TRAP is used for hardware single-step.
4. RST 7.5 is tied to OUT0 of 8254 programmable counter.
5. Monitor program uses last page of RAM for data storage, STACK area, and monitor control functions. The space is from F000H to F098H.

GPIO1

GPIO1 provides 8-bit output port. The I/O address is 00. The output drives 8-dot LED. We can use it for program testing easily.

System Programmable Port 8255

The I/O addresses of system port, 8255 are PORTA=10H, PORTB=11H, PORTC=12H and Control Port = 13H. Buzzer control pin is PORT C bit 7. To enable buzzer, write 7FH to PORTC.

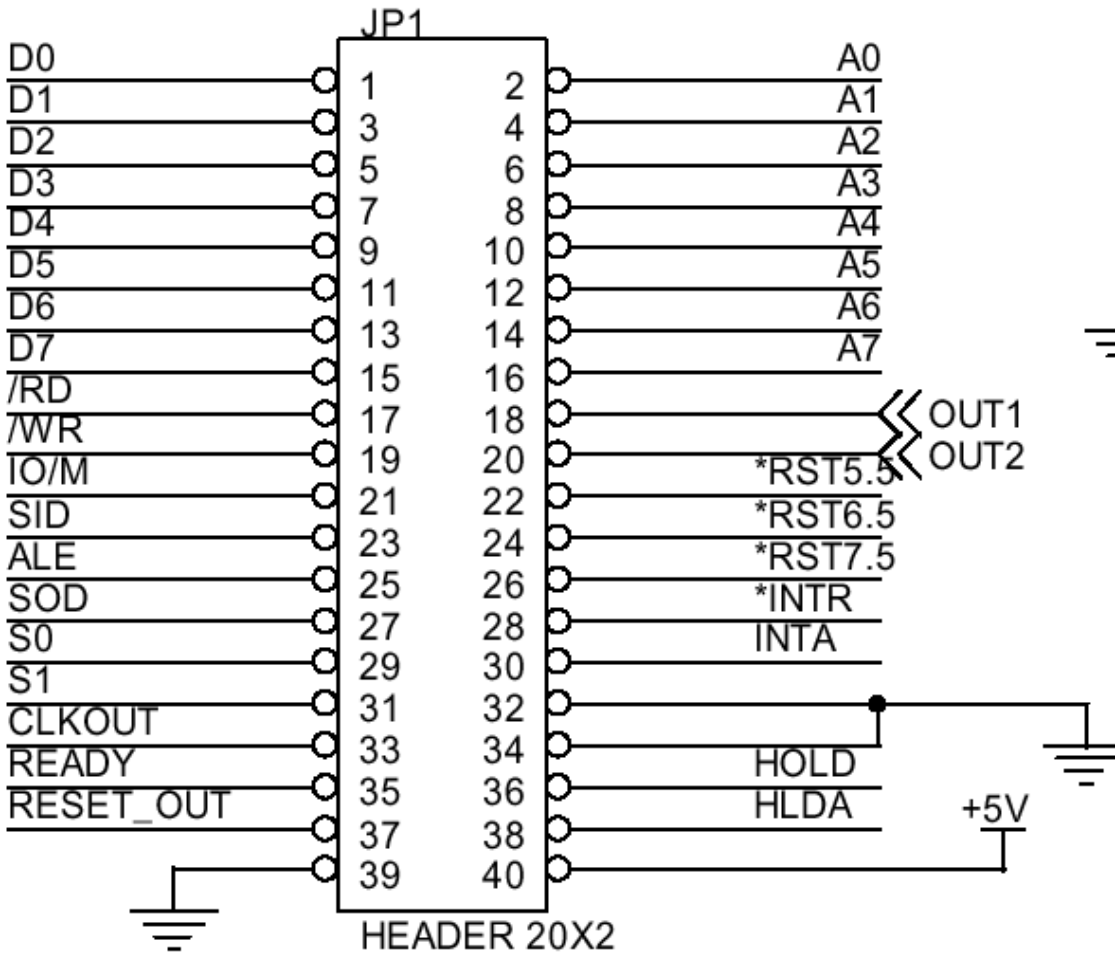
Programmable Counter 8254

The programmable counter, 8254 was supplied with clock signal from CLOCKOUT or 2MHz for counter0 and counter1. The internal registers of 8254 are mapped to I/O space from 20H to 23H.

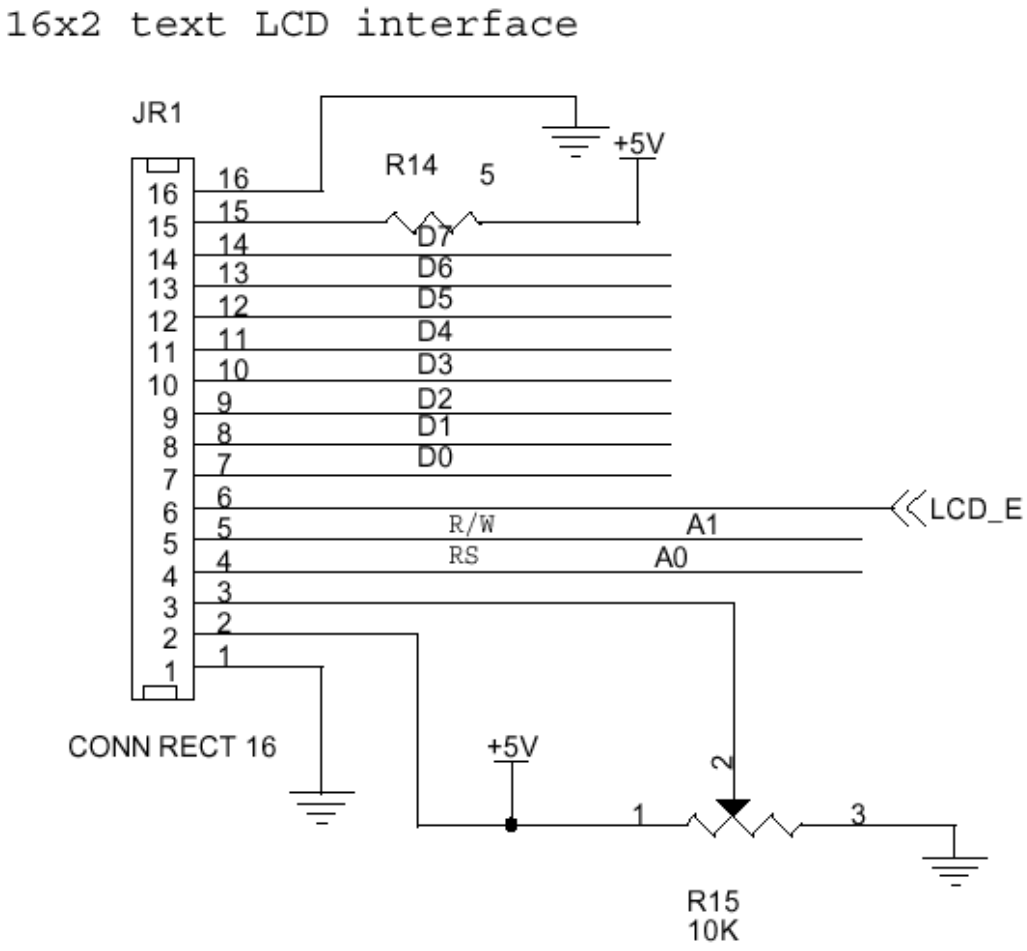
20H	COUNTER0
21H	COUNTER1
22H	COUNTER2
23H	CONTROL REGISTER

Headers and Connectors

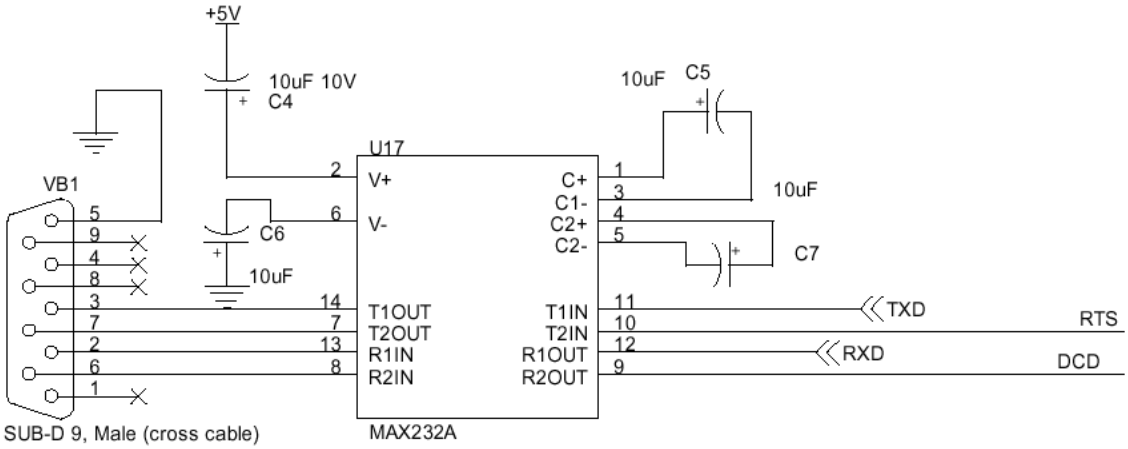
CPU Header JP1



Onboard LCD Header JR1



RS232C DB9 male connector VB1



Interrupts Test Button

The interrupt test button provides a single positive pulse that tied to CPU hardware interrupt pins, RST5.5, RST6.5 and INTR. User can select the pulse to be triggered for each pin by dip switch SW1. The onboard LED, D10 indicates the pulse is activated when press Test button.

Technical Specifications

CPU: CMOS 80C85 @4MHz
Memory: 64kB, 32kB 27C256, 32kB 62256
I/O port: Programmable Parallel port 8255, 8-bit GPIO
Counter: Programmable Counter 8254
UART: NS8250
Brownout Reset: KIA7042
Board Size: 170 x 170 mm
Weight: 320g (complete components except LCD)
DC Power Supply: AC-to-DC adapter +7.5V-12V 400mA
Power consumption: (350mA @12VDC)

Monitor Call Number

`00 - demo`
Scan 7-segment display with buffer display pointed by HL
Entry: HL
Exit: none

`01 - delay`
Delay subroutine using register pair DE, D is outer loop delay, E is inner loop.
Entry: DE
Exit: none

`02 - cold_boot`
Display cold-boot message on 7-segment LED.
Entry: none

`03 - scan`
Scan keyboard and display one cycle.
Entry: HL points the display buffer
Exit: key = scan code –1 no key pressed

`04 - cin`
Get character from console
Entry: none
Exit: A = character received

`05 - cout`
Send character to console
Entry: A = character to be sent
Exit: none

`06 - put_str`
Print string to console, string is terminated by 0.
Entry: HL
Exit: none

`07 - init_lcd`
Initialize LCD module
Entry: none
Exit: none

`08 - lcd_ready`

Wait until LCD module is ready.
Entry: none
Exit: none

`09 - clear_lcd`
Clear LCD display
Entry: none
Exit: none

`0A - goto_xy`
Set cursor position of LCD
Entry: HL, H = x, L = y
Exit: none

`0B - put_str_lcd`
Print string to LCD, string is terminated by 0
Entry: HL
Exit: none

`0C - put_ch_lcd`
Print character to LCD at current cursor position
Entry: A
Exit: none

`0D - demo2`
Running GPIO LED
Entry: none
Exit: none

NVRAM Bootable (available for special kit only)

User can replace U2, SRAM with a Nonvolatile RAM for program storage, when the board is powered off. A JMP instruction placed at 8000H will enable NVRAM bootable. The monitor program checks the location 8000H. If it has C3 (opcode of JMP instruction), it will jump to address 8000H. The feature allows application code to run easily. The monitor subroutines are available for the application program.

To get back to monitor mode, user can press USER key while press RESET. The byte C3 at location 8000H will be changed to 00, thus get back to normal RESET.

The sample code that demonstrates NVRAM Bootable is shown in Appendix D.

Appendix A Onboard LCD Driver Routines

```
;-------------------- onboard LCD registers -----------------
command_write  equ 50h
command_read   equ 52h
data_write     equ 51h
data_read      equ 53h
busy           equ 80h

;---------------------- LCD driver routines -----------------
lcd_ready:  push psw

lcd_ready1: in command_read
            ani busy
            jnz  lcd_ready1  ; wait until lcd ready
            pop psw

            ret

clear_lcd:  call lcd_ready
            mvi a,1
            out command_write
exit_clear: ret

init_lcd: call lcd_ready
          mvi a,38h
          out command_write
          call lcd_ready
          mvi a, 0ch
          out command_write
          call clear_lcd

          ret

; print ASCII text on LCD
; entry: HL pointer with 0 for end of string

put_str_lcd:  mov a,m    ; get A from [HL]
          cpi 0
          jnz put_str_lcd1
          ret

put_str_lcd1:

          call lcd_ready
```

```
          out data_write
          inx h
          jp put_str_lcd

; goto_xy set cursor location on lcd
; entry: HL: H = x, L = y

goto_xy:  call lcd_ready
          mov a,l
          cpi 0
          jnz goto_xy1
          mov a,h
          adi 80h
          out command_write
          ret

goto_xy1: cpi 1
          jnz goto_xy2
          mov a,h
          adi 0c0h
          out command_write
          ret

goto_xy2: cpi 2
          jnz goto_xy3
          mov a,h
          adi 094h
          out command_write
          ret

goto_xy3: cpi 3
          jnz goto_xy4
          mov a,h
          adi 0d4h
          out command_write
          ret

goto_xy4: ret

; put_ch_lcd put character to lcd
; entry: A

put_ch_lcd: call lcd_ready
            out data_write
            ret
```

Appendix B Subroutine Scan keyboard and Display

```
; subroutine scan keyboard and display
; entry: hl pointer to display buffer
; exit: key = scan code
;            -1 no key pressed
;

scan:    push h
         push b
         push d

         mvi c,6    ; for 6-digit LED
         mvi e,0    ; digit scan code appears at 4-to-10
                        decoder
         mvi d,0    ; key position
         mvi a,0ffh ; put -1 to key
         sta key    ; key = -1


scan1:   mov a,e
         ori 0f0h   ; high nibble must be 1111
         out system_port_c ; active digit first
         mov a,m        ; load a with [hl]
         out system_port_b ; then turn segment on

         mvi b,0    ; delay for electron transition process
wait1:   dcr b
         jnz wait1

         in  system_port_a  ; read input port

         mvi b,8            ; check all 8-row
shift_key: rar              ; rotate right through carry
         jc  next_key       ; if carry = 1 then no key
                        pressed

         push psw
         mov a,d
         sta key            ; save key position
         pop psw

next_key:
         inr d               ; next key position

         dcr b               ; until 8-bit was shifted
         jnz shift_key
```

```
        mvi a,0             ; clear a
        out system_port_b   ; turn off led

        inr e               ; next digit scan code
        inx h               ; next location

        dcr c               ; next column
        jnz scan1

        pop d
        pop b
        pop h
        ret


;---------- 8255 PPI system port I/O address ---------------
system_port_a:  equ 10h
system_port_b:  equ 11h
system_port_c:  equ 12h
system_port_control: equ 13h
```

Appendix C UART Driver Routines

```
;---------- 16C550 compatible UART I/O address ------------
; e.g. UM8250B, 16C450, 16C550

uart_buffer: equ 40h
uart_line_status: equ 45h
uart_fifo:        equ 42h
uart_lcr:         equ 43h
uart_divisor_lsb: equ 40h
uart_divisor_msb: equ 41h
uart_scr:         equ 47h

; initialize 16C550 uart to 9600 8n1 with 2MHz clock
; 2MHz/13 = 153846Hz

init_uart:

        mvi a,83h
        out uart_lcr     ; set DLAB bit to access divider

        mvi a,13
        out uart_divisor_lsb
        mvi a,0
        out uart_divisor_msb ; 2MHz/13 = 153846 Hz
                             ; 153846Hz/16 = 9615Hz
        mvi a,7
        out uart_fifo     ; init fifo and clear all buffers
        mvi a,03h
        out uart_lcr      ; clar DLAB

; check uart line status, if the byte is FF then no uart
;
;
        xra a
        out uart_scr      ; check if there is uart
        in uart_scr
        cpi 0
        jz found
        xra a
        sta uart_found
        ret

found   mvi a,1
        sta uart_found
        ret
```

```
cout:   mov b,a              ; save a

cout1:  in uart_line_status
        ani 20h               ; transmitter ready?
        jz cout1

        mov a,b              ; restore a
        out uart_buffer
        ret

cin:    in uart_line_status
        ani 1                  ; data available?
        jz  cin
        in uart_buffer
        ret

; print string terminated by 0
; input: HL

put_str:  mov a,m    ; get A from [HL]
          cpi 0
          jnz put_str1
          ret

put_str1: call cout
          inx h
          jp put_str
```

Appendix D Using NVRAM Bootable

```
; MTK-85 8085 Microprocessor Training Kit
; exp1.asm
;
; Using 8254 to produce 30.52Hz interrupt signal at RST7.5
;
; The 8254 counter0 was loaded with 0000 by system monitor.
; The input clock to the 8254 is 2MHz, the OUT0 then
                          produces
; 2MHz/65536 = 30.52Hz interrupt at RST7.5!
;
         CPU     "8085.TBL"    ;CPU TABLE
         HOF     "INT8"        ;HEX FORMAT
gpio     equ 0

; enable NVRAM boot running

         org 8000h
         jmp start      ; put instruction JMP to boot from
                         RAM

         org 803ch      ; interrupt vector of RST7.5
                         (relocated from 003CH)
         jmp service_rst7.5

         org 8100h

start:   mvi a,11111011b ; enable rst7.5
         sim             ; set interrupt mask register
         ei              ; enable interrupt

         jmp $           ; jump here


service_rst7.5:

         lda count       ; increment count
         inr a
         sta count
         out gpio        ; write to onboard LED
         ei
         ret

         org 0e000h
```

```
count    dfs 1          ; use RAM one byte for count
                        variable
         end
```

Appendix E Machine code and 8085 Instructions

MOVE, LOAD and STORE

40	MOV	B,B
41	MOV	B,C
42	MOV	B,D
43	MOV	B,E
44	MOV	B,H
45	MOV	B,L
46	MOV	B,M
47	MOV	B,A
48	MOV	C,B
49	MOV	C,C
4A	MOV	C,D
4B	MOV	C,E
4C	MOV	C,H
4D	MOV	C,L
4E	MOV	C,M
4F	MOV	C,A
50	MOV	D,B
51	MOV	D,C
52	MOV	D,D
53	MOV	D,E
54	MOV	D,H
55	MOV	D,L
56	MOV	D,M
57	MOV	D,A
58	MOV	E,B
59	MOV	E,C
5A	MOV	E,D
5B	MOV	E,E
5C	MOV	E,H
5D	MOV	E,L
5E	MOV	E,M
5F	MOV	E,A
60	MOV	H,B
61	MOV	H,C
62	MOV	H,D
63	MOV	H,E
64	MOV	H,H
65	MOV	H,L
66	MOV	H,M
67	MOV	H,A
68	MOV	L,B
69	MOV	L,C
6A	MOV	L,D
6B	MOV	L,E
6C	MOV	L,H
6D	MOV	L,L
6E	MOV	L,M
6F	MOV	L,A
70	MOV	M,B
71	MOV	M,C
72	MOV	M,D
73	MOV	M,E
74	MOV	M,H
75	MOV	M,L
77	MOV	M,A
78	MOV	A,B
79	MOV	A,C
7A	MOV	A,D
7B	MOV	A,E
7C	MOV	A,H
7D	MOV	A,L
7E	MOV	A,M
7F	MOV	A,A

3E nn	MVI A,byte
06 nn	MVI B,byte
0E nn	MVI C,byte
16 nn	MVI D,byte
1E nn	MVI E,byte
26 nn	MVI H,byte
2E nn	MVI L,byte
36 nn	MVI M,byte

01 nnnn	LXI B,dble
11 nnnn	LXI D,dble
21 nnnn	LXI H,dble
31 nnnn	LXI SP,dble

02	STAX	B
12	STAX	D
0A	LDAX	B
1A	LDAX	D
32 nnnn	STA	adr
3A nnnn	LDA	adr
22 nnnn	SHLD	adr
2A nnnn	LHLD	adr
EB	XCHG	

STACK

C5	PUSH	B
D5	PUSH	D
E5	PUSH	H
F5	PUSH	PSW

C1	POP	B
D1	POP	D
E1	POP	H
F1	POP	PSW
E3	XTHL	
F9	SPHL	
33	INX	SP
3B	DCX	SP

ARITHEMATICS

C6 nn	ADI	byte
CE nn	ACI	byte
80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
87	ADD	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
8F	ADC	A
D6 nn	SUI	byte
DE nn	SBI	byte
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
97	SUB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H
9D	SBB	L
9E	SBB	M
9F	SBB	A

09	DAD	B
19	DAD	D
29	DAD	H
39	DAD	SP

LOGICAL

E6 nn	ANI	byte
EE nn	XRI	byte
F6 nn	ORI	byte
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
A7	ANA	A
A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
AF	XRA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
B7	ORA	A

COMPARE

FE nn	CPI	byte
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
BF	CMP	A

ROTATE

```
07          RLC
17          RAL
0F          RRC
1F          RAR
```

JUMP

```
C3 nnnn     JMP    adr
DA nnnn     JC     adr
D2 nnnn     JNC    adr
CA nnnn     JZ     adr
C2 nnnn     JNZ    adr
F2 nnnn     JP     adr
FA nnnn     JM     adr
EA nnnn     JPE    adr
E2 nnnn     JPO    adr
E9          PCHL
```

CALL

```
CD nnnn     CALL   adr
DC nnnn     CC     adr
D4 nnnn     CNC    adr
CC nnnn     CZ     adr
C4 nnnn     CNZ    adr
F4 nnnn     CP     adr
FC nnnn     CM     adr
EC nnnn     CPE    adr
E4 nnnn     CPO    adr
```

RETURN

```
C9          RET
D8          RC
D0          RNC
C8          RZ
C0          RNZ
F0          RP
F8          RM
E8          RPE
E0          RPO
```

RESTART

```
C7          RST    0
CF          RST    1
D7          RST    2
DF          RST    3
E7          RST    4
EF          RST    5
F7          RST    6
FF          RST    7
```

INPUT/OUTPUT

```
DB nn       IN     byte
D3 nn       OUT    byte
```

INCREMENT/DECREMENT

```
04          INR    B
0C          INR    C
14          INR    D
1C          INR    E
24          INR    H
2C          INR    L
34          INR    M
3C          INR    A
03          INX    B
13          INX    D
23          INX    H
05          DCR    B
0D          DCR    C
15          DCR    D
1D          DCR    E
25          DCR    H
2D          DCR    L
35          DCR    M
3D          DCR    A
0B          DCX    B
1B          DCX    D
2B          DCX    H
```

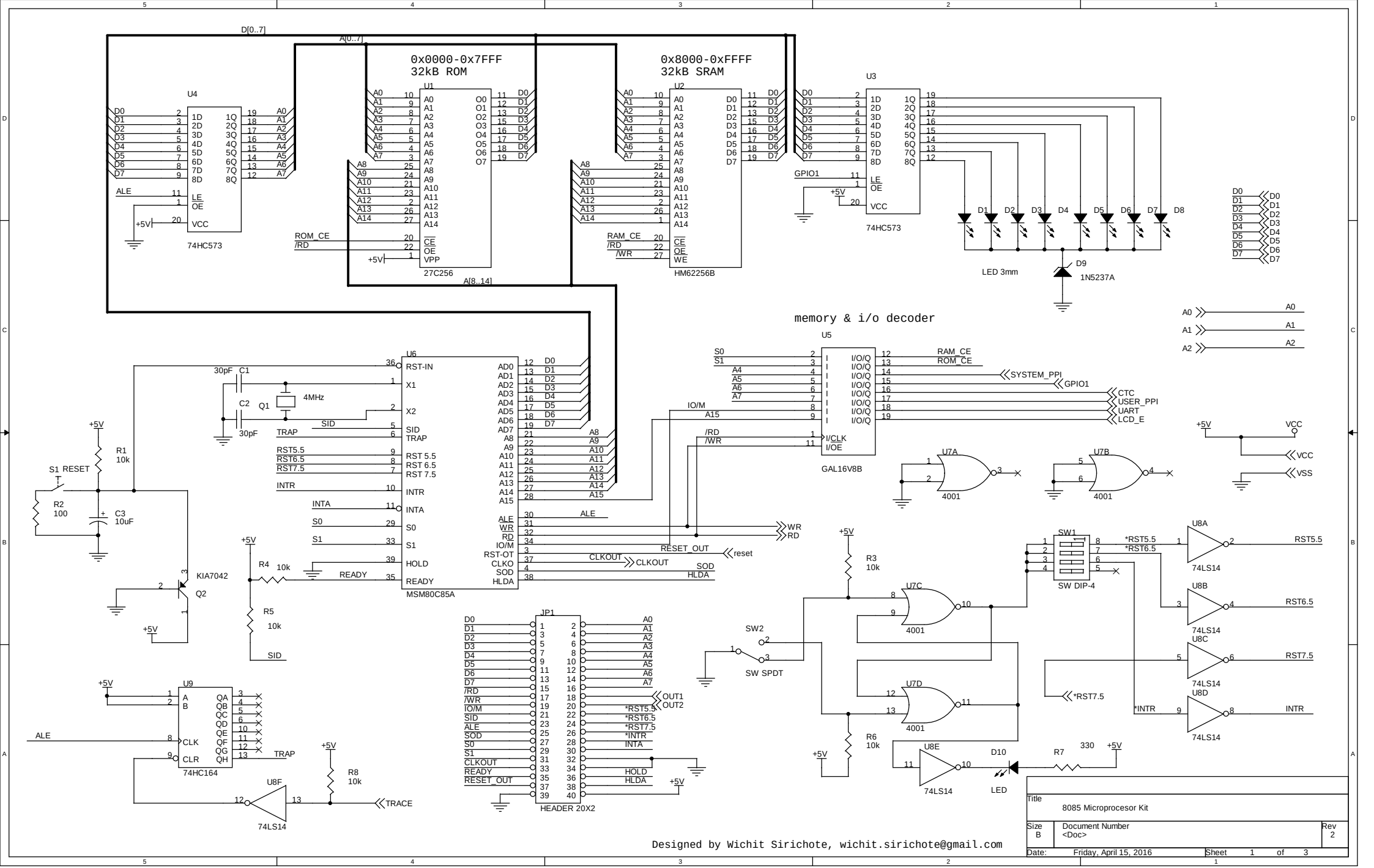
SPECIALS

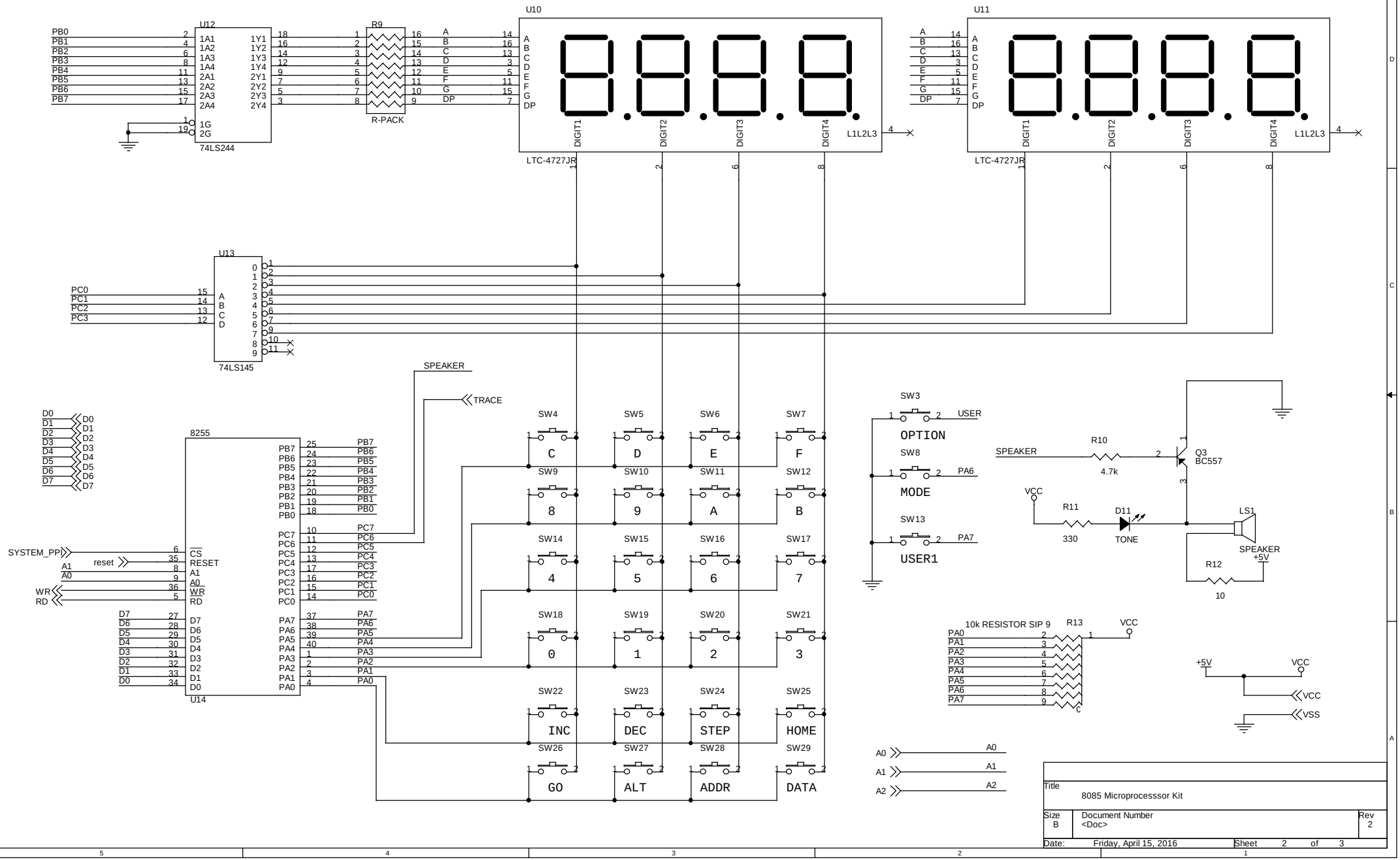
```
2F          CMA
37          STC
3F          CMC
27          DAA
```

CONTROL

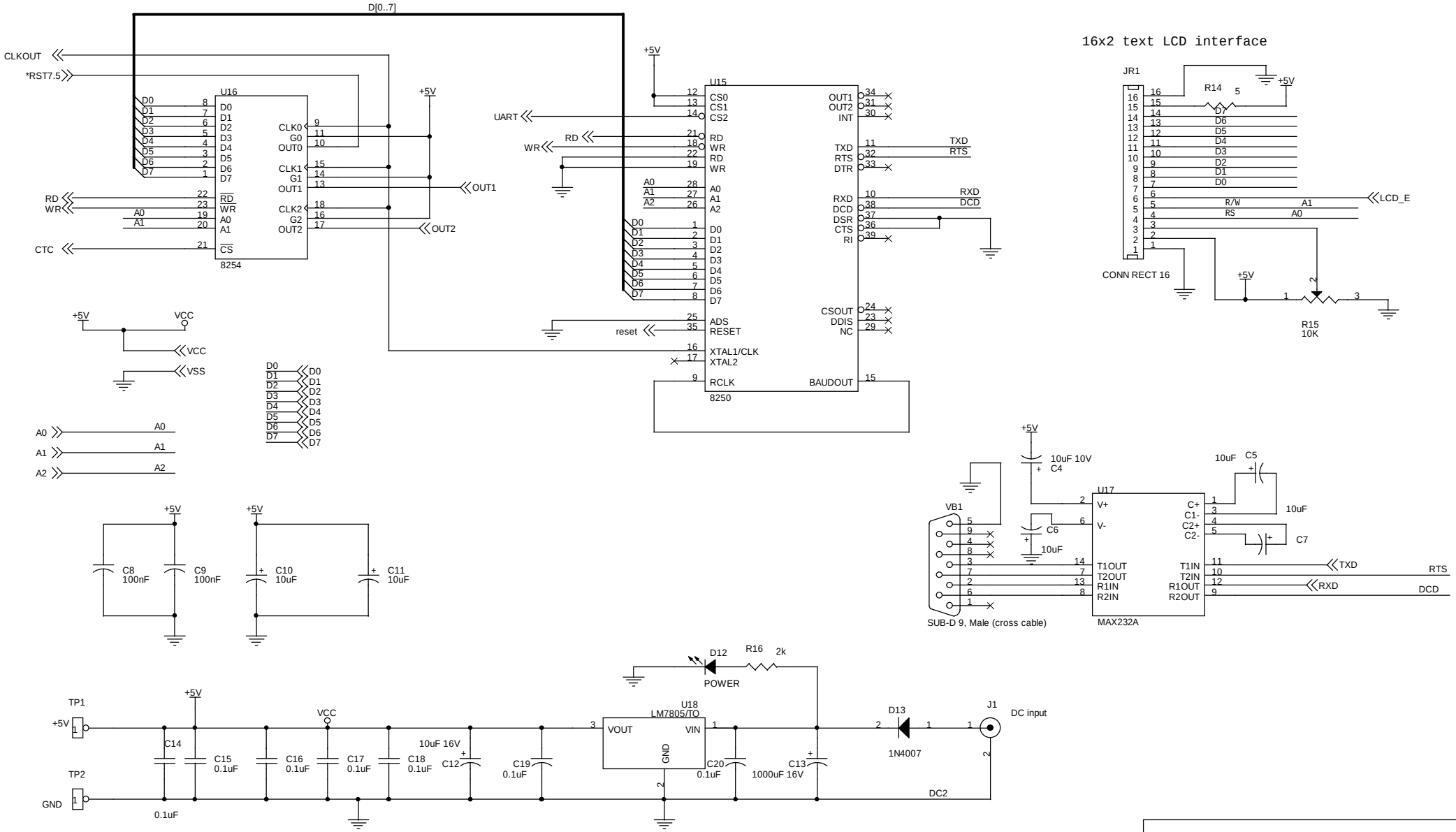
```
00          NOP
F3          DI
FB          EI
76          HLT
20          RIM
30          SIM
```

Appendix F Hardware schematic, Parts list


0x0000-0x7FFF
32kB ROM
0x8000-0xFFFF
32kB SRAM
memory & i/o decoder
U1
27C256
U2
HM62256B
U3
74HC573
U4
74HC573
U5
GAL16V8B
U6
MSM80C85A
U7A
4001
U7B
4001
U7C
4001
U7D
4001
U8A
74LS14
U8B
74LS14
U8C
74LS14
U8D
74LS14
U8E
74LS14
U8F
74LS14
U9
74HC164
JP1
HEADER 20X2
SW1
SW DIP-4
SW2
SW SPDT
LED 3mm
D9
1N5237A
D10
LED
Q1
4MHz
Q2
KIA7042
S1 RESET
R1
10k
R2
100
C1
30pF
C2
30pF
C3
10uF
R3
10k
R4
10k
R5
10k
R6
10k
R7
330
R8
10k
Designed by Wichit Sirichote, wichit.sirichote@gmail.com
Title
8085 Microprocesor Kit
Size
B
Document Number
<Doc>
Rev
2
Date:
Friday, April 15, 2016
Sheet
1
of
3



Title	8085 Microprocesssor Kit	
Size B	Document Number <Doc>	Rev 2
Date: Friday, April 15, 2016	Sheet 2 of 3	



Title	8085 Microprocessor Kit	
Size B	Document Number <Doc>	Rev 2
Date: Friday, April 15, 2016	Sheet 3 of 3	

PARTS LIST

Semiconductors

U1 27C256, 32kM EPROM
U2 HM62256B, 32kB SRAM
U4,U3 74HC573, 8-bit LATCH
U5 GAL16V8B, PLD
U6 MSM80C85A, Microprocessor
U7 4001, Nor gate
U8 74LS14, hex inverter
U9 74HC164, shift register
U11,U10 LTC-4727JR, 7-segment LED
U12 74LS244, tri-state driver
U13 74LS145, BCD to decimal
U14 8255, PPI
U15 8250, UART
U16 8254, CTC
U17 MAX232A
U18 LM7805/TO, voltage regulator
D1,D2,D4,D5,D6,D7,D8,D10 LED
D3 LED 3mm
D9 1N5227A
D11 TONE LED
D12 POWER LED
D13 1N4007
Q2 KIA7042
Q3 BC557

Capacitors

C1,C2 30pF, ceramic cap
C3,C5,C6,C7,C10,C11 10uF
C4 10uF 10V
C8,C9 100nF
C12 10uF 16V, electrolytic
C13 1000uF 16V, electrolytic

C14,C15,C16,C17,C18 0.1uF
C19,C20 0.1uF

Resistors (all resistors are 1/8W +/-5%)

R1,R3,R4,R5,R6,R8,R15 10K
R2 100 Ohms
R11,R7 330 Ohms
R9 100x8
R10 4.7k
R12 10
R13 10k RESISTOR SIP 9
R14 5 Ohms
R16 1k

Additional parts

JP1 HEADER 20X2
JR1 CONN RECT 16
J1 DC input
LS1 SPEAKER
Q1 4MHz

SW1 SW DIP-4, DIP switch
SW2 SW SPDT
SW3,SW4,SW5,SW6,SW7,SW8,SW PUSHBUTTON-SPST,SW9,SW10,SW11,SW12,SW13, SW14,SW15,SW16,SW17,SW18, SW19,SW20,SW21,SW22,SW23, SW24,SW25,SW26,SW27,SW28, SW29
S1 RESET
TP1 +5V
TP2 GND

VB1 SUB-D 9, Male (cross cable)
PCB double side plate through hole
LED cover Clear RED color 0.8mm acrylic plastic
Keyboard sticker printable SVG file

Appendix G Monitor source code listing

```
                ;---------------------------------------------------------------
                ; B8085.ASM
                ; monitor program for MTK-85 8085 MICROPROCESSOT TRAINING KIT
                ; COPYTIGHT (C) 2007-2015 BY WICHIT SIRICHOTE, kswichit@kmitl.ac.th
                ;
                ; source file was assembled with C32 Cross Assembler V3.0
                ;
                ; 18 May 2007 add insert byte, ALT E
                ;                delete byte, ALT D
                ;                click sound when key pressed
                ;
                ; 8 March 2015 remove repeat key
                ;              modified address and data entry mode
                ; 3 April 2016 replace buzzer with small speaker for tone experiment
                ;              add beep/no beep with ALT F press
                ; 16 April 2016 add delay after no beep
                ;
                ;---------------------------------------------------------------

0000                    CPU     "8085.TBL"    ;CPU TABLE
0000                    HOF     "INT8"        ;HEX FORMAT


                ; ---------- onboard GPIO ---------------------------------------

0000 =          gpio    equ 0 ; D0-D3 is 4-bit output port, D4-D7 is 4-bit input port


                ;---------- 8255 PPI system port I/O address ---------------------

0010 =          system_port_a:  equ 10h
0011 =          system_port_b:  equ 11h
0012 =          system_port_c:  equ 12h
0013 =          system_port_control: equ 13h

                ;-------------------- 8254 counter/timer -------------------------

0020 =          counter0_8254  equ 20h
0021 =          counter1_8254  equ 21h
0022 =          counter2_8254  equ 22h
0023 =          control_8254 equ 23h

0034 =          control_word_8254 equ 00110100B  ; mode 0, counter0

                ;---------- 8255 PPI user port I/O address -----------------------

0030 =          user_port_a:  equ 30h
0031 =          user_port_b:  equ 31h
0032 =          user_port_c:  equ 32h
0033 =          user_port_control: equ 33h


                ;---------- 16C550 compatible UART I/O address -------------------
                ; e.g., UM8250B, 16C450, 16C550

0040 =          uart_buffer: equ 40h
0045 =          uart_line_status: equ 45h
0042 =          uart_fifo:        equ 42h
0043 =          uart_lcr:         equ 43h
0040 =          uart_divisor_lsb: equ 40h
0041 =          uart_divisor_msb: equ 41h
0047 =          uart_scr:         equ 47h


                ;------------------- onboard LCD registers ----------------------

0050 =          command_write  equ 50h
0052 =          command_read   equ 52h
0051 =          data_write     equ 51h
0053 =          data_read      equ 53h
0080 =          busy           equ 80h

0009 =          TAB    EQU     9             ; ASCII TAB
0000 =          RS     EQU     0               ; terminator

000D =          cr: equ 0dh
```

```
 77   000A =         lf: equ 0ah
 78   0020 =         sp: equ 20h
 79
 80
 81   F000 =         system_ram equ 0f000h
 82
 83                  ;system_stack equ 0ffffh
 84
 85   8100 =         home_address  equ 8100h
 86
 87   0000 =         rom      equ  0 ;8000h           ; change to 8000 for testing under RAM
 88                                                  ; change to 0000 for rom programming
 89
 90   8000 =         my_rom   equ  8000h
 91
 92   0000                    ORG rom
 93   0000 C30001             JMP START           ; reset vector
 94
 95   0008                    ORG rom+8           ; RST 1  opcode is CF
 96   0008 C36102             jmp monitor_call
 97
 98   0010                    ORG rom+10h         ; RST 2  used for testing RST 7
 99                         ;  jmp service_rst2
100
101   0010 C31080            jmp my_rom+10h
102
103   0018                    ORG rom+18h         ; DF RST 3 for testing monitor call functior
104   0018 C31880             jmp my_rom+18h
105
106                          ; jmp monitor_call
107
108   0020                    ORG rom+20h         ; RST 4
109   0020 C32080             jmp my_rom+20h
110
111   0024                    ORG rom+24h
112                          ; jmp my_rom+24h
113   0024 C3DD02             jmp service_trap    ; sing step running service routine
114
115   0028                    ORG rom+28h         ; RST 5
116   0028 C32880             jmp my_rom+28h
117
118   002C                    ORG rom+2ch         ; relocate RST5.5 to external ram
119   002C C32C80             jmp my_rom+2ch
120
121   0030                    ORG rom+30h         ; relocate RST 6
122   0030 C33080             jmp my_rom+30h
123
124   0034                    ORG rom+34h         ; relocate RST6.5 to external ram
125   0034 C33480             jmp my_rom+34h
126
127   0038                    ORG ROM+38H
128   0038 C39302             jmp service_rst7    ; RST 7 service jump back to monitor
129
130   003C                    ORG rom+3ch         ; relocate RST7.5 to external ram
131   003C C33C80             jmp my_rom+3ch
132
133
134   0100                    ORG rom+100h
135
136
137   0100 F3        START    di
138   0101 3179F0             lxi sp,system_stack+32  ; point to top of system stack
139   0104 2199F0             lxi h,user_stack+32     ; point to top of user stack
140   0107 2234F0             shld user_SP
141
142
143   010A 3E90                MVI A,90H
144   010C D313                OUT system_port_control
145
146   010E 3EF0                mvi a,0f0h      ; disable trap
147   0110 D312                out system_port_c
148
149   0112 CD0612              call init_uart
150
151   0115 CD0802              call init_lcd
152   0118 21A41D              lxi h,prompt2
```

```
011B CD1A02             call put_str_lcd
011E 210100             lxi h,01
0121 CD2A02             call goto_xy
0124 21B91D             lxi h,text3
0127 CD1A02             call put_str_lcd

012A CD8F01             call init_8254

                ; NVRAM booting
                ; if location 8000H has C3 opcode then jump to 8000H
                ;      if user press USER1 with RESET put 00 to 8000H
                ;      ans skip booting

012D DB10               in system_port_a
012F E680               ani 80h
0131 CA4101             jz skip_boot

0134 3A0080             lda 8000H
0137 FEC3               cpi 0c3h
0139 C24101             jnz skip_boot
013C 210080             lxi h,8000h
013F E5                 push h
0140 C9                 ret      ; jump to NVRAM


0141 AF         skip_boot: xra a        ; write 00 to 8000H
0142 320080             sta 8000H
0145 3227F0       sta counter1   ; clear counter1

0148 3A29F0             lda warm_code
014B FE24               cpi "$"
014D CA6401             jz  skip_cold_boot
0150 3E24               mvi a,"$"
0152 3229F0             sta warm_code

0155 CD2D06             call test_buzzer

0158 3EFF               mvi a,0ffh
015A D300               out gpio       ; make GPIO LED on

015C CD760B             call cold_boot

015F 3E00         mvi a, 0
0161 3224F0       sta beep_flag


0164            skip_cold_boot:


0164 210081             lxi h,home_address
0167 222AF0             shld user_PC
016A 223CF0             shld pointer

016D 3A25F0             lda uart_found
0170 FE00               cpi 0
0172 CA7B01             jz skip_send_prompt

0175 CDB312             call send_prompt3

0178 CD3A0D             call send_prompt


017B            skip_send_prompt:

017B AF                 xra a
017C D300               out gpio         ; turn LED off

017E 3E00               mvi a,0
0180 3226F0             sta entry_mode         ; set data entry mode
0183 CDBE0A             call read_memory

0186 CD7E0C     main:   call scan_key
0189 CD8105             call key_execute
018C F28601             jp main
```

```
                ;---------------- initialize counter0 for RST7.5 interrupt -------------
                
018F 3E34       init_8254:  mvi a, control_word_8254
0191 D323                   out control_8254
0193 AF                     xra a
0194 D320                   out counter0_8254
0196 D320                   out counter0_8254
0198 C9                     ret

                ; convert 8-bit unsigned in A to ASCII string in line_buffer
                ; entry: A

0199            bin2ascii:
0199 1E00                   mvi e,0

019B FE64       bin1        cpi 100
019D DAA601                 jc  bin2
01A0 D664                   sui 100
01A2 1C                     inr e
01A3 C39B01                 jmp bin1

01A6 57         bin2:       mov d,a
01A7 7B                     mov a,e
01A8 C630                   adi "0"
01AA 3247F0                 sta line_buffer
01AD 7A                     mov a,d
01AE 1E00                   mvi e,0

01B0 FE0A       bin3:       cpi 10
01B2 DABB01                 jc  bin4
01B5 D60A                   sui 10
01B7 1C                     inr e
01B8 C3B001                 jmp bin3

01BB 57         bin4:       mov d,a
01BC 7B                     mov a,e
01BD C630                   adi "0"
01BF 3248F0                 sta line_buffer+1
01C2 7A                     mov a,d
01C3 C630                   adi "0"
01C5 3249F0                 sta line_buffer+2
01C8 C9                     ret


                ; print 8-bit unsigned decimal to terminal
                ; entry: A

01C9 CD9901     pint8u:     call bin2ascii
01CC 3A47F0                 lda line_buffer
01CF FE30                   cpi "0"
01D1 CAE401                 jz  pint1
01D4 CD2F12                 call cout
01D7 3A48F0                 lda line_buffer+1
01DA CD2F12                 call cout
01DD 3A49F0                 lda line_buffer+2
01E0 CD2F12                 call cout
01E3 C9                     ret

01E4 3A48F0     pint1:      lda line_buffer+1
01E7 FE30                   cpi "0"
01E9 CAEF01                 jz  pint2
01EC CD2F12                 call cout

01EF 3A49F0     pint2:      lda line_buffer+2
01F2 CD2F12                 call cout
01F5 C9                     ret


                ; convert 16-bit unsigned integer to ASCII code stored in line_buffer
                ; entry: HL
```

```
305
306
307
308
309                 ;---------------------- LCD driver routines -----------------------------
310
311   01F6 F5       lcd_ready:  push psw
312
313   01F7 DB52     lcd_ready1: in command_read
314   01F9 E680                 ani 80h
315   01FB C2F701               jnz  lcd_ready1  ; wait until lcd ready
316   01FE F1                   pop psw
317
318   01FF C9                   ret
319
320   0200 CDF601   clear_lcd:  call lcd_ready
321   0203 3E01                 mvi a,1
322   0205 D350                 out command_write
323   0207 C9       exit_clear: ret
324
325   0208 CDF601   init_lcd: call lcd_ready
326   020B 3E38               mvi a,38h
327   020D D350               out command_write
328   020F CDF601             call lcd_ready
329   0212 3E0C               mvi a, 0ch
330   0214 D350               out command_write
331   0216 CD0002             call clear_lcd
332
333   0219 C9                 ret
334
335                 ; print ASCII text on LCD
336                 ; entry: HL pointer with 0 for end of string
337
338   021A 7E       put_str_lcd:  mov a,m    ; get A from [HL]
339   021B FE00               cpi 0
340   021D C22102             jnz put_str_lcd1
341   0220 C9                 ret
342
343   0221          put_str_lcd1:
344
345   0221 CDF601             call lcd_ready
346   0224 D351               out data_write
347   0226 23                 inx h
348   0227 F21A02             jp put_str_lcd
349
350                 ; goto_xy set cursor location on lcd
351                 ; entry: HL: H = x, L = y
352
353   022A CDF601   goto_xy:  call lcd_ready
354   022D 7D                 mov a,l
355   022E FE00               cpi 0
356   0230 C23902             jnz goto_xy1
357   0233 7C                 mov a,h
358   0234 C680               adi 80h
359   0236 D350               out command_write
360   0238 C9                 ret
361
362   0239 FE01     goto_xy1: cpi 1
363   023B C24402             jnz goto_xy2
364   023E 7C                 mov a,h
365   023F C6C0               adi 0c0h
366   0241 D350               out command_write
367   0243 C9                 ret
368
369   0244 FE02     goto_xy2: cpi 2
370   0246 C24F02             jnz goto_xy3
371   0249 7C                 mov a,h
372   024A C694               adi 094h
373   024C D350               out command_write
374   024E C9                 ret
375
376   024F FE03     goto_xy3: cpi 3
377   0251 C25A02             jnz goto_xy4
378   0254 7C                 mov a,h
379   0255 C6D4               adi 0d4h
380   0257 D350               out command_write
```

```
0259 C9                 ret

025A C9         goto_xy4: ret

                ; put_ch_lcd put character to lcd
                ; entry: A

025B CDF601     put_ch_lcd: call lcd_ready
025E D351                   out data_write
0260 C9                     ret

                ;---------------------------------------------------------------------

                ; monitor call entry
                ; entry: E = monitor call number 0-255
                ; calling monitor function is made with RST 1 command after loading the
                ; regsiter E with call number
                ; destroy: BC user must save it in stack memory

0261 E5         monitor_call:  push h
0262 F5                        push psw
0263 D5                        push d

0264 7B                        mov a,e    ; get call number
0265 07                        rlc        ; x2
0266 5F                        mov e,a    ; put it back

0267 217502                    lxi h,vector_table
026A 1600                      mvi d,0
026C 19                        dad d      ; get location in jump table
026D 4E                        mov c,m
026E 23                        inx h
026F 46                        mov b,m

0270 D1                        pop d
0271 F1                        pop psw
0272 E1                        pop h

0273 C5                        push b     ; push address into top of stack

0274 C9                        ret      ; jump to monitor call function

0275            vector_table:
0275 A00B                      dwl demo  ; #0 running LED with HL pointer
0277 0006                      dwl delay ; #1 simple delay routine
0279 760B                      dwl cold_boot ; #2 show 8085 running
027B AE0C                      dwl scan  ; #3 scan display one cycle
027D 3B12                      dwl cin   ; #4 get byte from console
027F 2F12                      dwl cout  ; #5 print byte to console
0281 5812                      dwl put_str ; #6 print string with 0 terminator to console
0283 0802                      dwl init_lcd ; #7 initialize lcd
0285 F601                      dwl lcd_ready ; #8 wait until lcd is ready
0287 0002                      dwl clear_lcd ; #9 clear lcd display
0289 2A02                      dwl goto_xy   ; #10 set lcd cursor position
028B 1A02                      dwl put_str_lcd ; #11 print ASCII string on lcd
028D 5B02                      dwl put_ch_lcd  ; #12 print ASCII letter on lcd
028F F205                      dwl test_led    ; #13 run LED onboard
0291 C901                      dwl pint8u     ; #14 print 8-bit unsigned to terminal


                ; save CPU registers to stack and write them to user registers
                ;

0293            service_rst7:
0293 F5                     push psw
0294 C5                     push b
0295 D5                     push d

0296 E1                     pop h
0297 2230F0                 shld user_DE
029A E1                     pop h
029B 222EF0                 shld user_BC
029E E1                     pop h
029F 222CF0                 shld user_AF
02A2 E1                     pop h
```

```
457
458   02A3 222AF0               shld user_PC   ; store next PC
459
460   02A6 210000               lxi h,0000h
461   02A9 39                   dad sp        ; get SP
462   02AA 2234F0               shld user_SP  ; save user SP
463
464   02AD CDBE0A               call read_memory
465   02B0 CD540F               call register_display1
466
467   02B3 2A57F0               lhld save_stack
468
469   02B6 F9                   sphl            ; restore system stack
470
471
472   02B7 C9                   ret
473
474
475                 ; test diplay register after break
476                 ; RST 2 opcode is D7
477                 ; later will be changed to RST 7
478
479   02B8          service_rst2:
480   02B8 F5                   push psw
481   02B9 C5                   push b
482   02BA D5                   push d
483
484   02BB E1                   pop h
485   02BC 2230F0               shld user_DE
486   02BF E1                   pop h
487   02C0 222EF0               shld user_BC
488   02C3 E1                   pop h
489   02C4 222CF0               shld user_AF
490   02C7 E1                   pop h
491
492   02C8 222AF0               shld user_PC   ; store next PC
493
494   02CB 210000               lxi h,0000h
495   02CE 39                   dad sp        ; get content of SP
496
497   02CF 2234F0               shld user_SP   ; save user SP
498
499   02D2 CDBE0A               call read_memory
500   02D5 CD540F               call register_display1
501
502   02D8 2A57F0               lhld save_stack
503
504   02DB F9                   sphl            ; restore system stack
505
506
507   02DC C9                   ret
508
509
510                 ; service trap for single step running
511                 ; disable trap input by setting system port c.6
512                 ; save CPU registers to user registers
513
514   02DD          service_trap:
515   02DD F5                   push psw        ; save A and Flag
516
517   02DE 3EFF                 mvi a,0ffh
518   02E0 D312                 out system_port_c ; turn trap off by clearing shift register
519
520   02E2 C5                   push b
521   02E3 D5                   push d
522   02E4 E5                   push h
523
524   02E5 E1                   pop h
525   02E6 2232F0               shld user_HL   ; save HL
526   02E9 E1                   pop h
527   02EA 2230F0               shld user_DE
528   02ED E1                   pop h
529   02EE 222EF0               shld user_BC
530   02F1 E1                   pop h
531   02F2 222CF0               shld user_AF
532
```

```
533   02F5 E1                   pop h           ; store next PC
534   02F6 222AF0               shld user_PC
535
536   02F9 210000               lxi h,0
537   02FC 39                   dad sp
538   02FD 2234F0               shld user_SP   ; save user SP
539
540   0300 CDBE0A               call read_memory
541
542   0303 3A25F0               lda uart_found
543   0306 FE00                 cpi 0
544   0308 CA0E03               jz skip1
545   030B CD540F               call register_display1
546
547   030E            skip1:
548   030E 2A57F0               lhld save_stack
549
550   0311 F9                   sphl            ; restore system stack
551
552   0312 C9                   ret             ; jump back to main body
553
554
555                   ; disassemble machine code into mnemonic
556
557   0313            disassemble1:
558   0313 3A22F0                   lda command
559   0316 FE64                     cpi "d"
560   0318 C23903                   jnz exit_disassemble
561
562   031B 21A11E                   lxi h,disassemble_text
563   031E CD5812                   call put_str
564
565   0321 CDDA12                   call new_line
566   0324 0E10                     mvi c,16       ; 16 lines
567
568
569   0326 C5         dis2:         push b
570
571   0327 CDDA12                   call new_line
572
573   032A CDB003                   call d_disassemble
574
575   032D C1                       pop b
576   032E 0D                       dcr c
577   032F C22603                   jnz dis2
578
579   0332 CDDA12                   call new_Line
580   0335 CD3A0D                   call send_prompt
581   0338 C9                       ret
582
583
584   0339 C9         exit_disassemble: ret
585
586
587                   ; disassemble opcode to mnemonic
588                   ; entry: user_PC
589                   ; exit: user_PC = next address
590
591   033A 2A2AF0     disassemble:  lhld user_PC
592
593   033D E5                       push h
594
595   033E 7C                       mov a,h
596   033F CDCB12                   call out2x
597   0342 7D                       mov a,l
598   0343 CDCB12                   call out2x
599   0346 CDE512                   call space
600
601   0349 7E                       mov a,m     ; get opcode
602   034A CD2604                   call get_number_of_byte
603   034D 4F                       mov c,a
604
605   034E 7E         disassem3:    mov a,m
606   034F CDCB12                   call out2x
607   0352 23                       inx h
608   0353 0D                       dcr c
```

```
609   0354 C24E03                jnz disassem3
610
611   0357 E1                    pop h
612
613   0358 7E                    mov a,m
614   0359 CD2604                call get_number_of_byte
615   035C FE01                  cpi 1
616   035E C26603                jnz one_tab
617
618   0361 3E09                  mvi a,tab         ; print two tabs for one byte opcode
619   0363 CD2F12                call cout
620
621   0366 3E09      one_tab:    mvi a,tab         ; else only one tab
622   0368 CD2F12                call cout
623
624   036B E5                    push h
625
626   036C 7E                    mov a,m           ; get opcode
627
628   036D F5                    push psw
629
630   036E 210000                lxi h,0000h       ; clear HL
631   0371 6F                    mov l,a
632
633   0372 29                    dad h            ; HL = HLx2
634
635   0373 5D                    mov e,l
636   0374 54                    mov d,h
637
638   0375 210B15                lxi h,ins_table
639   0378 19                    dad d            ; ADD HL,DE
640   0379 5E                    mov e,m
641   037A 23                    inx h
642   037B 56                    mov d,m
643
644   037C 6B                    mov l,e
645   037D 62                    mov h,d
646
647   037E CD5812                call put_str
648
649   0381 F1                    pop psw
650   0382 E1                    pop h
651
652   0383 CD2604                call get_number_of_byte
653   0386 FE01                  cpi 1
654   0388 C29003                jnz disassem1
655   038B 23                    inx h
656   038C 222AF0                shld user_PC
657   038F C9                    ret
658
659   0390 FE02      disassem1:  cpi 2
660   0392 C29F03                jnz disassem2
661   0395 23                    inx h
662   0396 7E                    mov a,m
663   0397 CDCB12                call out2x
664   039A 23                    inx h
665   039B 222AF0                shld user_PC
666   039E C9                    ret
667
668   039F 23        disassem2:  inx h
669   03A0 23                    inx h
670   03A1 7E                    mov a,m
671   03A2 CDCB12                call out2x
672   03A5 2B                    dcx h
673   03A6 7E                    mov a,m
674   03A7 CDCB12                call out2x
675   03AA 23                    inx h
676   03AB 23                    inx h
677   03AC 222AF0                shld user_PC
678   03AF C9                    ret
679
680                  ; disassemble opcode to mnemonic with command 'd'
681                  ; entry: pointer
682                  ; exit: pointer = next address
683
684   03B0 2A3CF0    d_disassemble:  lhld pointer
```

```
685
686   03B3 E5                     push h
687
688   03B4 7C                     mov a,h
689   03B5 CDCB12                 call out2x
690   03B8 7D                     mov a,l
691   03B9 CDCB12                 call out2x
692   03BC CDE512                 call space
693
694   03BF 7E                     mov a,m     ; get opcode
695   03C0 CD2604                 call get_number_of_byte
696   03C3 4F                     mov c,a
697
698   03C4 7E       d_disassem3:    mov a,m
699   03C5 CDCB12                 call out2x
700   03C8 23                     inx h
701   03C9 0D                     dcr c
702   03CA C2C403                 jnz d_disassem3
703
704   03CD E1                     pop h
705
706   03CE 7E                     mov a,m
707   03CF CD2604                 call get_number_of_byte
708   03D2 FE01                   cpi 1
709   03D4 C2DC03                 jnz d_one_tab
710
711   03D7 3E09                   mvi a,tab         ; print two tabs for one byte opcode
712   03D9 CD2F12                 call cout
713
714   03DC 3E09     d_one_tab:      mvi a,tab         ; else only one tab
715   03DE CD2F12                 call cout
716
717   03E1 E5                     push h
718
719   03E2 7E                     mov a,m           ; get opcode
720
721   03E3 F5                     push psw
722
723   03E4 210000                 lxi h,0000h       ; clear HL
724   03E7 6F                     mov l,a
725
726   03E8 29                     dad h            ; HL = HLx2
727
728   03E9 5D                     mov e,l
729   03EA 54                     mov d,h
730
731   03EB 210B15                 lxi h,ins_table
732   03EE 19                     dad d            ; ADD HL,DE
733   03EF 5E                     mov e,m
734   03F0 23                     inx h
735   03F1 56                     mov d,m
736
737   03F2 6B                     mov l,e
738   03F3 62                     mov h,d
739
740   03F4 CD5812                 call put_str
741
742   03F7 F1                     pop psw
743   03F8 E1                     pop h
744
745   03F9 CD2604                 call get_number_of_byte
746   03FC FE01                   cpi 1
747   03FE C20604                 jnz d_disassem1
748   0401 23                     inx h
749   0402 223CF0                 shld pointer
750   0405 C9                     ret
751
752   0406 FE02     d_disassem1:    cpi 2
753   0408 C21504                 jnz d_disassem2
754   040B 23                     inx h
755   040C 7E                     mov a,m
756   040D CDCB12                 call out2x
757   0410 23                     inx h
758   0411 223CF0                 shld pointer
759   0414 C9                     ret
760
```

```
761   0415 23        d_disassem2:  inx h
762   0416 23                      inx h
763   0417 7E                      mov a,m
764   0418 CDCB12                  call out2x
765   041B 2B                      dcx h
766   041C 7E                      mov a,m
767   041D CDCB12                  call out2x
768   0420 23                      inx h
769   0421 23                      inx h
770   0422 223CF0                  shld pointer
771   0425 C9                      ret
772
773
774                  ; get number of byte
775                  ; entry: A = OPCODE
776                  ; exit: A = number of byte, 1,2,3
777                  ;           0 = undefined opcode
778
779   0426           get_number_of_byte:
780
781   0426 FE01                    cpi 01
782   0428 C22E04                  jnz number1
783   042B 3E03                    mvi a,3
784   042D C9                      ret
785
786   042E FE06      number1:      cpi 6
787   0430 C23604                  jnz number2
788   0433 3E02                    mvi a,2
789   0435 C9                      ret
790
791   0436 FE0E      number2:      cpi 0eh
792   0438 C23E04                  jnz number3
793   043B 3E02                    mvi a,2
794   043D C9                      ret
795
796   043E FE11      number3:      cpi 11h
797   0440 C24604                  jnz number4
798   0443 3E03                    mvi a,3
799   0445 C9                      ret
800
801   0446 FE16      number4:      cpi 16h
802   0448 C24E04                  jnz number5
803   044B 3E02                    mvi a,2
804   044D C9                      ret
805
806   044E FE1E      number5:      cpi 1eh
807   0450 C25604                  jnz number6
808   0453 3E02                    mvi a,2
809   0455 C9                      ret
810
811   0456 FE21      number6:      cpi 21h
812   0458 C25E04                  jnz number7
813   045B 3E03                    mvi a,3
814   045D C9                      ret
815
816   045E FE22      number7:      cpi 22h
817   0460 C26604                  jnz number8
818   0463 3E03                    mvi a,3
819   0465 C9                      ret
820
821   0466 FE26      number8:      cpi 26h
822   0468 C26E04                  jnz number9
823   046B 3E02                    mvi a,2
824   046D C9                      ret
825
826   046E FE2A      number9:      cpi 2ah
827   0470 C27604                  jnz number10
828   0473 3E03                    mvi a,3
829   0475 C9                      ret
830
831   0476 FE2E      number10:     cpi 2eh
832   0478 C27E04                  jnz number11
833   047B 3E02                    mvi a,2
834   047D C9                      ret
835
836   047E FE31      number11:     cpi 31h
```

```
0480 C28604                jnz number12
0483 3E03                  mvi a,3
0485 C9                    ret

0486 FE32      number12:   cpi 32h
0488 C28E04                jnz number13
048B 3E03                  mvi a,3
048D C9                    ret

048E FE36      number13:   cpi 36h
0490 C29604                jnz number14
0493 3E03                  mvi a,3
0495 C9                    ret

0496 FE3A      number14:   cpi 3ah
0498 C29E04                jnz number15
049B 3E03                  mvi a,3
049D C9                    ret

049E FE3E      number15:   cpi 3eh
04A0 C2A604                jnz number16
04A3 3E02                  mvi a,2
04A5 C9                    ret

04A6 FEC2      number16:   cpi 0c2h
04A8 C2AE04                jnz number17
04AB 3E03                  mvi a,3
04AD C9                    ret

04AE FEC3      number17:   cpi 0c3h
04B0 C2B604                jnz number18
04B3 3E03                  mvi a,3
04B5 C9                    ret

04B6 FEC4      number18:   cpi 0c4h
04B8 C2BE04                jnz number19
04BB 3E03                  mvi a,3
04BD C9                    ret

04BE FEC6      number19:   cpi 0c6h
04C0 C2C604                jnz number20
04C3 3E02                  mvi a,2
04C5 C9                    ret

04C6 FECA      number20:   cpi 0cah
04C8 C2CE04                jnz number21
04CB 3E03                  mvi a,3
04CD C9                    ret

04CE FECC      number21:   cpi 0cch
04D0 C2D604                jnz number22
04D3 3E03                  mvi a,3
04D5 C9                    ret

04D6 FECD      number22:   cpi 0cdh
04D8 C2DE04                jnz number23
04DB 3E03                  mvi a,3
04DD C9                    ret

04DE FECE      number23:   cpi 0ceh
04E0 C2E604                jnz number24
04E3 3E02                  mvi a,2
04E5 C9                    ret

04E6 FED2      number24:   cpi 0d2h
04E8 C2EE04                jnz number25
04EB 3E03                  mvi a,3
04ED C9                    ret

04EE FED3      number25:   cpi 0d3h
04F0 C2F604                jnz number26
04F3 3E02                  mvi a,2
04F5 C9                    ret

04F6 FED4      number26:   cpi 0d4h
04F8 C2FE04                jnz number27
```

```
913   04FB 3E03                 mvi a,3
914   04FD C9                   ret
915
916   04FE FED6      number27:  cpi 0d6h
917   0500 C20605               jnz number28
918   0503 3E02                 mvi a,2
919   0505 C9                   ret
920
921   0506 FEDA      number28:  cpi 0dah
922   0508 C20E05               jnz number29
923   050B 3E03                 mvi a,3
924   050D C9                   ret
925
926   050E FEDB      number29:  cpi 0dbh
927   0510 C21605               jnz number30
928   0513 3E02                 mvi a,2
929   0515 C9                   ret
930
931   0516 FEDC      number30:  cpi 0dch
932   0518 C21E05               jnz number31
933   051B 3E03                 mvi a,3
934   051D C9                   ret
935
936   051E FEE2      number31:  cpi 0e2h
937   0520 C22605               jnz number32
938   0523 3E03                 mvi a,3
939   0525 C9                   ret
940
941   0526 FEE4      number32:  cpi 0e4h
942   0528 C22E05               jnz number33
943   052B 3E03                 mvi a,3
944   052D C9                   ret
945
946   052E FEE6      number33:  cpi 0e6h
947   0530 C23605               jnz number34
948   0533 3E02                 mvi a,2
949   0535 C9                   ret
950
951   0536 FEEA      number34:  cpi 0eah
952   0538 C23E05               jnz number35
953   053B 3E03                 mvi a,3
954   053D C9                   ret
955
956   053E FEEC      number35:  cpi 0ech
957   0540 C24605               jnz number36
958   0543 3E03                 mvi a,3
959   0545 C9                   ret
960
961   0546 FEEE      number36:  cpi 0eeh
962   0548 C24E05               jnz number37
963   054B 3E02                 mvi a,2
964   054D C9                   ret
965
966   054E FEF2      number37:  cpi 0f2h
967   0550 C25605               jnz number38
968   0553 3E03                 mvi a,3
969   0555 C9                   ret
970
971   0556 FEF4      number38:  cpi 0f4h
972   0558 C25E05               jnz number39
973   055B 3E03                 mvi a,3
974   055D C9                   ret
975
976   055E FEF6      number39:  cpi 0f6h
977   0560 C26605               jnz number40
978   0563 3E02                 mvi a,2
979   0565 C9                   ret
980
981   0566 FEFA      number40:  cpi 0fah
982   0568 C26E05               jnz number41
983   056B 3E03                 mvi a,3
984   056D C9                   ret
985
986   056E FEFC      number41:  cpi 0fch
987   0570 C27605               jnz number42
988   0573 3E03                 mvi a,3
```

```
 989   0575 C9                    ret
 990
 991   0576 FEFE      number42:   cpi 0feh
 992   0578 C27E05                jnz number43
 993   057B 3E02                  mvi a,2
 994   057D C9                    ret
 995
 996   057E 3E01      number43:   mvi a,1
 997   0580 C9                    ret
 998
 999
1000
1001
1002
1003
1004
1005
1006                 ; execute key 0-F or 10H-19H
1007
1008   0581 FE10      key_execute: cpi 10h
1009   0583 D28D05                jnc  function_key   ; 0-9 jump to data key
1010   0586 57                    mov d,a
1011   0587 CD3106                call buzzer
1012   058A F20906                jp   data_key
1013
1014   058D FE12      function_key: cpi 12h
1015   058F C29805                 jnz function1
1016   0592 CD3106                 call buzzer
1017   0595 F2650A                 jp  increment
1018
1019   0598 FE15      function1:    cpi 15h
1020   059A C2A305                  jnz function2
1021   059D CD3106                  call buzzer
1022   05A0 F2780A                  jp  decrement
1023
1024   05A3 FE10      function2:    cpi 10h
1025   05A5 C2AE05                  jnz function3
1026   05A8 CD3106                  call buzzer
1027   05AB F2010A                  jp  address_mode
1028
1029   05AE FE11      function3:    cpi 11h
1030   05B0 C2B905                  jnz function4
1031   05B3 CD3106                  call buzzer
1032   05B6 F20E0A                  jp  data_mode
1033
1034   05B9 FE13      function4:    cpi 13h
1035   05BB C2C505                  jnz function5
1036   05BE CD3106                  call buzzer
1037   05C1 F29B09                  jp  go
1038   05C4 C9                      ret
1039
1040   05C5 FE14      function5:   cpi 14h
1041   05C7 C2D005                 jnz function6
1042   05CA CD3106                 call buzzer
1043   05CD F2190A                 jp function_2nd
1044
1045
1046   05D0 FE16      function6:   cpi 16h
1047   05D2 C2DB05                 jnz function7
1048   05D5 CD3106                 call buzzer
1049   05D8 F2BF09                 jp single_step
1050
1051   05DB FE17      function7:    cpi 17h
1052   05DD C2E605                  jnz function8
1053   05E0 CD3106                  call buzzer
1054   05E3 F28109                  jp home
1055
1056   05E6 FE18      function8:    cpi 18h
1057   05E8 C2F105                  jnz function9
1058   05EB CD3106                  call buzzer
1059   05EE F23D0A                  jp modify_register
1060
1061   05F1 C9        function9:    ret
1062
1063                 ; test running onboard led
1064
```

```
1065   05F2 3E01      test_led:  mvi a,1
1066
1067   05F4 D300      test_led1: out gpio
1068   05F6 115010               lxi d,1050h
1069   05F9 CD0006               call delay
1070   05FC 07                   rlc
1071   05FD C3F405               jmp  test_led1
1072
1073
1074                  ; delay subroutine
1075                  ; entry: D= outer loop E=inner loop (should be 0 for long delay)
1076                  ; exit: none
1077
1078   0600 1D        delay:     dcr e
1079   0601 C20006               jnz delay
1080   0604 15                   dcr d
1081   0605 C20006               jnz delay
1082   0608 C9                   ret
1083
1084
1085   0609 3A26F0    data_key:     lda entry_mode
1086   060C FE00                    cpi 0
1087   060E C21406                  jnz data_key1
1088   0611 F2CF06                  jp  enter_data
1089
1090   0614 FE01      data_key1:    cpi 1
1091   0616 C21C06                  jnz data_key2
1092   0619 F20907                  jp  enter_address
1093
1094   061C FE02      data_key2:    cpi 2
1095   061E C22406                  jnz data_key3
1096   0621 F24307                  jp select_register
1097
1098   0624 FE03      data_key3:    cpi 3
1099   0626 C22C06                  jnz data_key4
1100   0629 F26F06                  jp enter_register
1101
1102   062C           data_key4:
1103
1104   062C C9                      ret
1105
1106
1107   062D CD4006    test_buzzer:  call beep_on
1108   0630 C9              ret
1109
1110                        ;mvi a,7fh
1111                              ;out system_port_c
1112                              ;lxi d,1000h
1113                              ;call delay
1114                              ;mvi a,0ffh
1115                              ;out system_port_c
1116                              ;ret
1117
1118
1119                  ; produce beep output at system port c.7
1120                  ; click when key pressed
1121
1122   0631 3A24F0    buzzer:       lda beep_flag
1123   0634 E601                    ani 1
1124   0636 CA4006            jz  beep_on
1125
1126   0639 0600              mvi b,0
1127   063B 05        delay_nobeep: dcr b
1128   063C C23B06                  jnz delay_nobeep
1129   063F C9                      ret
1130   0640           beep_on:
1131   0640 0E20              mvi c,20h
1132
1133   0642 3E7F      buzzer1:      mvi a,7fh
1134   0644 D312                    out system_port_c  ;nop              ;out system_port_c
1135   0646 CD5906                  call delay_us
1136   0649 3EFF                    mvi a,0ffh
1137   064B D312                    out system_port_c  ;nop              ;out system_port_c
1138   064D CD5906            call delay_us
1139
1140   0650 0D                      dcr c
```

```
0651 C24206                 jnz buzzer1

0654 3EFF                   mvi a,0ffh
0656 D312                   out system_port_c

0658 C9                     ret

0659 0660      delay_us:    mvi b,60h
065B 05        delay_us1:   dcr b
065C C25B06                 jnz delay_us1
065F C9                     ret


               ;-----  turn display off while key has been pressing ------------------
               ;       useful for no function accepted

0660 219E1D    display_off:  lxi h,off_display
0663 CDAE0C    off_display1:  call scan
0666 3A21F0                 lda key
0669 FEFF                   cpi 0ffh
066B C26306                 jnz  off_display1  ; loop if key still pressed
066E C9                     ret

               ;************* modify current displayed register **********************
               ; entry: current user register displayed
               ;


066F           enter_register:

066F 2A38F0                 lhld current_register

0672 4E                     mov c,m
0673 23                     inx h
0674 46                     mov b,m
0675 210000                 lxi h,0

0678 09                     dad b          ; MOV HL,BC

0679 5A                     mov e,d        ; save key code to E

067A           shift_register:
067A 7D                      mov a,l
067B 07                      rlc
067C 6F                      mov l,a
067D 7C                      mov a,h
067E 17                      ral
067F 67                      mov h,a

0680 7D                      mov a,l
0681 07                      rlc
0682 6F                      mov l,a
0683 7C                      mov a,h
0684 17                      ral
0685 67                      mov h,a

0686 7D                      mov a,l
0687 07                      rlc
0688 6F                      mov l,a
0689 7C                      mov a,h
068A 17                      ral
068B 67                      mov h,a

068C 7D                      mov a,l
068D 07                      rlc
068E 6F                      mov l,a
068F 7C                      mov a,h
0690 17                      ral
0691 67                      mov h,a

0692 7D                      mov a,l

0693 E6F0                    ani 0f0h
0695 82                      add d
0696 6F                      mov l,a
```

```
1217   0697 44                      mov b,h
1218   0698 4D                      mov c,l
1219
1220   0699 2A38F0                  lhld current_register
1221   069C 71                      mov m,c
1222   069D 23                      inx h
1223   069E 70                      mov m,b
1224
1225   069F 2A38F0                 lhld current_register
1226
1227   06A2 4E                     mov c,m
1228   06A3 23                     inx h
1229   06A4 46                     mov b,m
1230   06A5 210000                 lxi h,0
1231
1232   06A8 09                     dad b          ; MOV HL,BC
1233
1234   06A9 CD930A                 call read_register
1235
1236   06AC F5                    push psw
1237
1238   06AD 3A13F0          lda buffer+3
1239   06B0 F680                   ori 80h
1240   06B2 3213F0                 sta buffer+3
1241
1242   06B5 3A12F0          lda buffer+2
1243   06B8 F680                   ori 80h
1244   06BA 3212F0                 sta buffer+2
1245
1246   06BD 3A11F0          lda buffer+1
1247   06C0 F680                   ori 80h
1248   06C2 3211F0                 sta buffer+1
1249
1250   06C5 3A10F0          lda buffer
1251   06C8 F680                   ori 80h
1252   06CA 3210F0                 sta buffer
1253
1254
1255
1256
1257
1258   06CD F1              pop psw
1259
1260   06CE C9                      ret
1261
1262                  ; enter nibble into current location
1263
1264   06CF 2A2AF0    enter_data:   lhld user_PC
1265
1266   06D2 5A                      mov e,d        ; save key code to E
1267
1268   06D3 3A27F0                  lda counter1
1269   06D6 FE00                    cpi 0
1270   06D8 C2E206                  jnz shift_data
1271   06DB 3C                      inr a
1272   06DC 3227F0                  sta counter1
1273   06DF 3E00                    mvi a,0
1274   06E1 77                      mov m,a
1275
1276   06E2 7E        shift_data:   mov a,m
1277
1278   06E3 07                      rlc
1279   06E4 07                      rlc
1280   06E5 07                      rlc
1281   06E6 07                      rlc
1282   06E7 E6F0                    ani 0f0h       ; make low nibble to 0 before insert
1283   06E9 83                      add e          ; insert low nibble to A
1284   06EA 77                      mov m,a
1285   06EB 57                      mov d,a
1286   06EC 7E                      mov a,m        ; check if the space is ram or rom
1287   06ED BA                      cmp d
1288   06EE CA0507                  jz  it_is_ram
1289
1290                  ; if it was rom them turn of led while key has been pressed
1291
1292   06F1 219E1D                  lxi h,off_display
```

```
06F4 CDAE0C    enter_data1:  call scan
06F7 3A21F0                  lda key
06FA FEFF                    cpi 0ffh
06FC C2F406                  jnz  enter_data1  ; loop if key still pressed

06FF CDA70C                  call debounce     ; debounce after key was released

0702 2110F0                  lxi h,buffer      ; back to show display again

0705           it_is_ram:
0705 CDBE0A                  call read_memory
0708 C9                      ret

                 ; enter nibble into current pointer

0709 2A2AF0    enter_address: lhld user_PC
070C 5A                       mov e,d        ; save key code to E

070D 3A28F0                  lda counter2
0710 FE00                    cpi 0
0712 C21F07                  jnz shift_address
0715 3C                      inr a
0716 3228F0                  sta counter2
0719 210000                  lxi h,0
071C 222AF0                  shld user_PC

071F           shift_address:
071F 7D                       mov a,l
0720 07                       rlc
0721 6F                       mov l,a
0722 7C                       mov a,h
0723 17                       ral
0724 67                       mov h,a

0725 7D                       mov a,l
0726 07                       rlc
0727 6F                       mov l,a
0728 7C                       mov a,h
0729 17                       ral
072A 67                       mov h,a

072B 7D                       mov a,l
072C 07                       rlc
072D 6F                       mov l,a
072E 7C                       mov a,h
072F 17                       ral
0730 67                       mov h,a

0731 7D                       mov a,l
0732 07                       rlc
0733 6F                       mov l,a
0734 7C                       mov a,h
0735 17                       ral
0736 67                       mov h,a

0737 7D                       mov a,l

0738 E6F0                     ani 0f0h
073A 82                       add d
073B 6F                       mov l,a

073C 222AF0                   shld user_PC   ; store new pointer

073F CDBE0A                   call read_memory

0742 C9                       ret


                 ;****************** ALT register display ****************************

0743 5A        select_register: mov e,d  ; save key for selecting user register

0744 7A                      mov a,d
0745 FE00                    cpi 0
0747 C26107                  jnz register1

```

```
1369   074A 3E77                    mvi a,77h       ; AF register pair
1370   074C 3214F0                  sta buffer+4
1371   074F 3E71                    mvi a,71h
1372   0751 3215F0                  sta buffer+5
1373
1374   0754 212CF0                  lxi h,user_AF
1375   0757 2238F0                  shld current_register
1376
1377
1378   075A 2A2CF0                  lhld user_AF
1379   075D CD930A                  call read_register
1380   0760 C9                      ret
1381
1382   0761           register1:
1383   0761 FE01                    cpi 1
1384   0763 C27D07                  jnz register2
1385
1386   0766 3E7C                    mvi a,7ch       ; BC register pair
1387   0768 3214F0                  sta buffer+4
1388   076B 3E39                    mvi a,39h
1389   076D 3215F0                  sta buffer+5
1390
1391   0770 212EF0                  lxi h,user_BC
1392   0773 2238F0                  shld current_register
1393
1394   0776 2A2EF0                  lhld user_BC
1395   0779 CD930A                  call read_register
1396
1397   077C C9                      ret
1398
1399   077D           register2:
1400   077D FE02                    cpi 2
1401   077F C29907                  jnz register3
1402
1403   0782 3E5E                    mvi a,5eh       ; DE register pair
1404   0784 3214F0                  sta buffer+4
1405   0787 3E79                    mvi a,79h
1406   0789 3215F0                  sta buffer+5
1407
1408   078C 2130F0                  lxi h,user_DE
1409   078F 2238F0                  shld current_register
1410
1411   0792 2A30F0                  lhld user_DE
1412   0795 CD930A                  call read_register
1413
1414   0798 C9                      ret
1415
1416   0799           register3:
1417   0799 FE03                    cpi 3
1418   079B C2B507                  jnz register4
1419
1420   079E 3E76                    mvi a,76h       ; HL register pair
1421   07A0 3214F0                  sta buffer+4
1422   07A3 3E38                    mvi a,38h
1423   07A5 3215F0                  sta buffer+5
1424
1425   07A8 2132F0                  lxi h,user_HL
1426   07AB 2238F0                  shld current_register
1427
1428   07AE 2A32F0                  lhld user_HL
1429   07B1 CD930A                  call read_register
1430
1431   07B4 C9                      ret
1432   07B5           register4:
1433   07B5 FE04                    cpi 4
1434   07B7 C2D107                  jnz register5
1435
1436   07BA 3E6D                    mvi a,6dh        ; user SP
1437   07BC 3214F0                  sta buffer+4
1438   07BF 3E73                    mvi a,73h
1439   07C1 3215F0                  sta buffer+5
1440
1441   07C4 2134F0                  lxi h,user_SP
1442   07C7 2238F0                  shld current_register
1443
1444   07CA 2A34F0                  lhld user_SP
```

```
1445   07CD CD930A                  call read_register
1446
1447   07D0 C9                      ret
1448
1449   07D1           register5:
1450   07D1 FE05                    cpi 5
1451   07D3 C2F807                  jnz register6
1452
1453   07D6 3E73                    mvi a,73h       ; user PC
1454   07D8 3214F0                  sta buffer+4
1455   07DB 3E39                    mvi a,39h
1456   07DD 3215F0                  sta buffer+5
1457
1458   07E0 212AF0                  lxi h,user_PC
1459   07E3 2238F0                  shld current_register
1460
1461   07E6 2A2AF0                  lhld user_PC
1462   07E9 CD930A                  call read_register
1463
1464   07EC C9                      ret
1465
1466                  ;---- display carry flag ------------------------------------------------
1467
1468   07ED C2F507    put_flag:     jnz put_high1
1469   07F0 3E3F                    mvi a,3fh
1470   07F2 C3F707                  jmp skip_put_high1
1471
1472   07F5 3E06      put_high1:    mvi a,06h
1473
1474   07F7           skip_put_high1:
1475   07F7 C9                      ret
1476
1477   07F8           register6:
1478   07F8 FE06                    cpi 6
1479   07FA C22008                  jnz register7
1480
1481   07FD 3E39                    mvi a,39h       ; carry flag
1482   07FF 3210F0                  sta buffer
1483   0802 3E6E                    mvi a,6eh
1484   0804 3211F0                  sta buffer+1
1485   0807 3E48                    mvi a,48h
1486   0809 3212F0                  sta buffer+2
1487
1488   080C 2A2CF0                  lhld user_AF
1489   080F 7D                      mov a,l
1490
1491   0810 E601                    ani 1
1492   0812 CDED07                  call put_flag
1493   0815 3213F0                  sta buffer+3
1494
1495   0818 AF                      xra a
1496   0819 3214F0                  sta buffer+4
1497   081C 3215F0                  sta buffer+5
1498
1499   081F C9                      ret
1500
1501   0820           register7:
1502   0820 FE07                    cpi 7
1503   0822 C24B08                  jnz register8
1504
1505   0825 3E49                    mvi a,49h       ; zero flag
1506   0827 3210F0                  sta buffer
1507   082A 3E79                    mvi a,79h
1508   082C 3211F0                  sta buffer+1
1509   082F 3E50                    mvi a,50h
1510   0831 3212F0                  sta buffer+2
1511   0834 3E5C                    mvi a,5ch
1512   0836 3213F0                  sta buffer+3
1513   0839 3E48                    mvi a,48h
1514   083B 3214F0                  sta buffer+4
1515
1516
1517   083E 2A2CF0                  lhld user_AF
1518   0841 7D                      mov a,l
1519
1520   0842 E640                    ani 40h
```

```
0844 CDED07                 call put_flag
0847 3215F0                 sta  buffer+5

084A C9                     ret


084B            register8:
084B FE08                   cpi 8
084D C27C08                 jnz register9

0850 3E6D                   mvi a,6dh       ; sign flag
0852 3210F0                 sta buffer
0855 3E11                   mvi a,11h
0857 3211F0                 sta buffer+1
085A 3E6F                   mvi a,6fh
085C 3212F0                 sta buffer+2
085F 3E54                   mvi a,54h
0861 3213F0                 sta buffer+3
0864 3E48                   mvi a,48h
0866 3214F0                 sta buffer+4


0869 2A2CF0                 lhld user_AF
086C 7D                     mov a,l

086D 17                     ral
086E DA7608                 jc  put_high2
0871 3E3F                   mvi a,3fh
0873 C37808                 jmp skip_put_high2

0876 3E06       put_high2:  mvi a,06h
0878            skip_put_high2:
0878 3215F0                 sta  buffer+5

087B C9                     ret


087C            register9:
087C FE09                   cpi 9
087E C2A408                 jnz register10

0881 3E77                   mvi a,77h       ; AC flag
0883 3210F0                 sta buffer
0886 3E39                   mvi a,39h
0888 3211F0                 sta buffer+1
088B 3E48                   mvi a,48h
088D 3212F0                 sta buffer+2

0890 2A2CF0                 lhld user_AF
0893 7D                     mov a,l

0894 E610                   ani 10h
0896 CDED07                 call put_flag
0899 3213F0                 sta  buffer+3
089C AF                     xra  a
089D 3214F0                 sta buffer+4
08A0 3215F0                 sta buffer+5

08A3 C9                     ret

08A4            register10:
08A4 FE0A                   cpi 10
08A6 C2CA08                 jnz break

08A9 3E73                   mvi a,73h       ; Parity flag
08AB 3210F0                 sta buffer
08AE 3E48                   mvi a,48h
08B0 3211F0                 sta buffer+1

08B3 2A2CF0                 lhld user_AF
08B6 7D                     mov a,l

08B7 E604                   ani 4
08B9 CDED07                 call put_flag
08BC 3212F0                 sta  buffer+2
08BF AF                     xra  a
```

```
1597   08C0 3213F0                sta buffer+3
1598   08C3 3214F0                sta buffer+4
1599   08C6 3215F0                sta buffer+5
1600   08C9 C9                    ret
1601
1602
1603                ; ------------------ ALT B SET BREAK POINT --------------------------
1604   08CA FE0B    break:        cpi 11
1605   08CC C20409                jnz clear_break
1606
1607   08CF 2A2AF0                lhld user_PC  ; save user PC
1608   08D2 223EF0                shld break_address
1609
1610   08D5 7E                    mov a,m       ; get user code
1611   08D6 3240F0                sta break_opcode ; save it
1612
1613   08D9 E5                    push h
1614   08DA CDBE0A                call read_memory
1615   08DD E1                    pop h
1616   08DE 3EFF                  mvi a,0FFh    ; RST 7 opcode
1617   08E0 77                    mov m,a       ; replace user code with RST 7
1618
1619   08E1 F5                    push psw
1620
1621   08E2 3A10F0                lda buffer
1622   08E5 F680                  ori 80h
1623   08E7 3210F0                sta buffer
1624
1625   08EA 3A11F0                lda buffer+1
1626   08ED F680                  ori 80h
1627   08EF 3211F0                sta buffer+1
1628
1629   08F2 3A12F0                lda buffer+2
1630   08F5 F680                  ori 80h
1631   08F7 3212F0                sta buffer+2
1632
1633   08FA 3A13F0                lda buffer+3
1634   08FD F680                  ori 80h
1635   08FF 3213F0                sta buffer+3
1636   0902 F1                    pop psw
1637
1638   0903 C9                    ret
1639
1640                ;----------------- ALT C CLEAR BREAK POINT --------------------------
1641
1642   0904 FE0C    clear_break:  cpi 12
1643   0906 C21F09                jnz insert_byte
1644
1645   0909 2140F0                lxi h,break_opcode ; restore user code
1646   090C 7E                    mov a,m
1647
1648   090D 2A3EF0                lhld break_address
1649   0910 77                    mov m,a
1650   0911 222AF0                shld user_PC
1651   0914 CDBE0A                call read_memory
1652   0917 AF                    xra a
1653   0918 3226F0                sta entry_mode
1654   091B CD020B                call mode_indicator
1655
1656   091E C9                    ret
1657
1658                ;--------------- ALT E insert byte ---------------------------------
1659                ; insert byte within 512 bytes from current location
1660
1661   091F FE0E    insert_byte: cpi 14      ; test with key E
1662   0921 C24A09               jnz delete_byte
1663
1664   0924 2A2AF0               lhld user_PC
1665   0927 E5             push h         ; save PC to stack
1666
1667   0928 110002               lxi d,512
1668   092B 19             dad d
1669   092C E5             push h
1670   092D C1             pop b          ; copy HL to BC
1671   092E 0B             dcx b
1672
```

```
1673   092F 110002          lxi d,512      ; load counter with 512 bytes
1674
1675   0932           insert_byte1:
1676   0932 0A              ldax b
1677   0933 77              mov m,a
1678   0934 2B              dcx h
1679   0935 0B              dcx b
1680   0936 1B              dcx d
1681   0937 7B              mov a,e
1682   0938 B2              ora d         ; check DE ==0
1683   0939 C23209          jnz insert_byte1
1684
1685   093C E1              pop h         ; restore user PC
1686   093D AF              xra a
1687   093E 77              mov m,a       ; store 00 at insert byte
1688   093F CDBE0A                 call read_memory
1689   0942 AF                     xra a
1690   0943 3226F0                 sta entry_mode
1691   0946 CD020B                 call mode_indicator
1692
1693   0949 C9                      ret
1694
1695
1696                  ;--------------- ALT D delete byte --------------------------------
1697                  ; delete byte within 512 bytes
1698
1699   094A FE0D      delete_byte: cpi 13
1700   094C C26F09                 jnz beep_chk
1701
1702   094F 2A2AF0                 lhld user_PC
1703   0952 E5                     push h
1704   0953 E5              push h
1705   0954 C1              pop b
1706
1707   0955 03              inx b
1708   0956 110002          lxi d,512
1709
1710   0959           delete_byte1:
1711   0959 0A              ldax b
1712   095A 77              mov m,a
1713   095B 23              inx h
1714   095C 03              inx b
1715   095D 1B              dcx d
1716   095E 7B              mov a,e
1717   095F B2              ora d   ; check if DE ==0
1718   0960 C25909          jnz delete_byte1
1719
1720   0963 E1                     pop h
1721   0964 CDBE0A          call read_memory
1722   0967 AF                     xra a
1723   0968 3226F0                 sta entry_mode
1724   096B CD020B                 call mode_indicator
1725
1726   096E C9                     ret
1727
1728                  ;--------------- ALT F BEEP/NO BEEP  --------------------------------
1729   096F FE0F      beep_chk:   cpi 15
1730   0971 C27D09                jnz option1
1731
1732   0974 3A24F0          lda beep_flag
1733   0977 EE01            xri 1
1734   0979 3224F0          sta beep_flag
1735   097C C9              ret
1736
1737
1738   097D           option1:
1739   097D CD6006                  call display_off ; no service key
1740   0980 C9                      ret
1741
1742
1743
1744   0981 210081    home:         lxi h,home_address
1745   0984 222AF0                  shld user_PC
1746   0987 2110F0                  lxi h,buffer
1747   098A CDBE0A                  call read_memory
1748   098D AF                      xra a
```

```
1749   098E 3226F0                sta entry_mode
1750   0991 CD020B                call mode_indicator
1751   0994 C9                    ret
1752
1753
1754   0995 3E2A      debug:      mvi a,"*"
1755   0997 CD2F12                call cout
1756   099A C9                    ret
1757
1758                  ; go function, jump from monitor program to user program
1759                  ; save system stack and load user stack
1760                  ; load CPU registers with user registers before jump
1761
1762   099B           go:
1763   099B 210000                lxi h,0
1764
1765   099E 39                    dad sp        ; save system stack
1766   099F 2257F0                shld save_stack
1767
1768   09A2 2A34F0                lhld user_SP  ; get user stack
1769   09A5 F9                    sphl          ; load user stack
1770
1771   09A6 2A2AF0                lhld user_PC
1772   09A9 E5                    push h
1773   09AA 2A2CF0                lhld user_AF
1774   09AD E5                    push h
1775   09AE 2A2EF0                lhld user_BC
1776   09B1 E5                    push h
1777   09B2 2A30F0                lhld user_DE
1778   09B5 E5                    push h
1779   09B6 2A32F0                lhld user_HL
1780   09B9 E5                    push h
1781
1782   09BA E1                    pop h
1783   09BB D1                    pop d
1784   09BC C1                    pop b
1785   09BD F1                    pop psw
1786
1787   09BE C9                    ret            ; jump to user program
1788
1789
1790                  ; single step
1791                  ; load CPU registers with user registers, enable trap signal then jump to
1792                  ; program
1793                  ; disassemble line to be executed
1794
1795   09BF           single_step:
1796   09BF 210000                lxi h,0
1797
1798   09C2 39                    dad sp        ; save system stack
1799   09C3 2257F0                shld save_stack
1800
1801   09C6 2A34F0                lhld user_SP
1802   09C9 F9                    sphl          ; load user stack
1803
1804
1805   09CA 2A2AF0                lhld user_PC   ; get address to be executed
1806   09CD E5                    push h         ; save to stack
1807
1808   09CE 3A25F0                lda uart_found
1809   09D1 FE00                  cpi 0
1810   09D3 CADF09                jz skip2       ; if no uart, skip disassemble
1811
1812   09D6 CDDA12                call new_line
1813   09D9 CDEB12                call send_tab
1814   09DC CD3A03                call disassemble
1815
1816   09DF           skip2:
1817
1818   09DF E1                    pop h
1819   09E0 222AF0                shld user_PC
1820
1821   09E3 2A2AF0                lhld user_PC
1822   09E6 E5                    push h
1823   09E7 2A2CF0                lhld user_AF
1824   09EA E5                    push h
```

```
1825   09EB 2A2EF0                lhld user_BC
1826   09EE E5                    push h
1827   09EF 2A30F0                lhld user_DE
1828   09F2 E5                    push h
1829   09F3 2A32F0                lhld user_HL
1830   09F6 E5                    push h
1831
1832   09F7 E1                    pop h
1833   09F8 D1                    pop d
1834   09F9 C1                    pop b
1835
1836   09FA 3EBF                  mvi a,0bfh         ; make port_c.6 low to enable trap
1837   09FC D312                  out system_port_c  ;
1838
1839                  ; now the shift register 74LS164 is running
1840                  ; within 8 ALE, trap will be high, trap will be recorgnized after instruc
1841                  ; followed RET was executed
1842
1843   09FE 00                    nop            ; 1 cycles
1844   09FF F1                    pop psw        ; 5 cycles
1845   0A00 C9                    ret            ; 3 cycles
1846
1847
1848                  ; set mode to 1
1849
1850   0A01 3E01      address_mode: mvi a,1
1851   0A03 3226F0                sta entry_mode
1852   0A06 CDBE0A                call read_memory
1853   0A09 AF                    xra a
1854   0A0A 3228F0                sta counter2
1855   0A0D C9                    ret
1856
1857   0A0E AF        data_mode:    xra a
1858   0A0F 3226F0                sta entry_mode
1859   0A12 3227F0                sta counter1
1860   0A15 CDBE0A                call read_memory
1861   0A18 C9                    ret
1862
1863   0A19 3E02      function_2nd: mvi a,2
1864   0A1B 3226F0                sta entry_mode
1865   0A1E 3E77                  mvi a,77h
1866   0A20 3210F0                sta buffer
1867   0A23 3E38                  mvi a,38h
1868   0A25 3211F0                sta buffer+1
1869   0A28 3E78                  mvi a,78h
1870   0A2A 3212F0                sta buffer+2
1871   0A2D 3E00                  mvi a,0
1872   0A2F 3213F0                sta buffer+3
1873   0A32 3E00                  mvi a,0
1874   0A34 3214F0                sta buffer+4
1875   0A37 3E00                  mvi a,0
1876   0A39 3215F0                sta buffer+5
1877   0A3C C9                    ret
1878
1879
1880                  ; set entry mode to 3
1881                  ; hex data will be used for register modifying
1882
1883   0A3D           modify_register:
1884
1885   0A3D F5                    push psw
1886
1887   0A3E 3E03                  mvi a,3
1888   0A40 3226F0                sta entry_mode
1889
1890   0A43 3A13F0            lda buffer+3
1891   0A46 F680                  ori 80h
1892   0A48 3213F0                sta buffer+3
1893
1894   0A4B 3A12F0            lda buffer+2
1895   0A4E F680                  ori 80h
1896   0A50 3212F0                sta buffer+2
1897
1898   0A53 3A11F0            lda buffer+1
1899   0A56 F680                  ori 80h
1900   0A58 3211F0                sta buffer+1
```

```
1901
1902   0A5B 3A10F0          lda buffer
1903   0A5E F680                   ori 80h
1904   0A60 3210F0                 sta buffer
1905
1906   0A63 F1                     pop psw
1907
1908   0A64 C9                     ret
1909
1910
1911
1912
1913
1914
1915
1916
1917                  ; increment key works with mode0 or mode1 display
1918
1919   0A65 3E00      increment:   mvi a,0
1920   0A67 3226F0                 sta entry_mode  ; switch to data mode
1921   0A6A 3227F0                 sta counter1    ; clear event counter1
1922
1923   0A6D 2A2AF0                 lhld user_PC
1924   0A70 23                     inx  h
1925   0A71 222AF0                 shld user_PC
1926   0A74 CDBE0A                 call read_memory
1927   0A77 C9                     ret
1928
1929                  ; decrement key works with mode0 or mode1 display
1930
1931   0A78 3E00      decrement:  mvi a,0
1932   0A7A 3226F0                sta entry_mode  ; switch to data mode
1933   0A7D 2A2AF0                lhld user_PC
1934   0A80 2B                    dcx  h
1935   0A81 222AF0                shld user_PC
1936   0A84 CDBE0A                call read_memory
1937   0A87 C9                    ret
1938
1939
1940                  ; convert nibble 0-F to 8-bit seven segment code
1941                  ; entry: A
1942                  ; exit: A
1943
1944   0A88           to_seven_segment:
1945
1946   0A88 E60F                  ani 0fh       ; get only low nibble as the index
1947   0A8A 218E1D                lxi h,convert
1948   0A8D 5F                    mov e,a
1949   0A8E 1600                  mvi d,0
1950   0A90 19                    dad d
1951   0A91 7E                    mov a,m        ; get code
1952   0A92 C9                    ret
1953
1954                  ; convert [HL] to display buffer 0-3
1955                  ; for register display
1956                  ; entry: HL
1957
1958   0A93           read_register:
1959   0A93 E5                     push h
1960   0A94 7C                     mov a,h
1961   0A95 F5                     push psw
1962   0A96 0F                     rrc
1963   0A97 0F                     rrc
1964   0A98 0F                     rrc
1965   0A99 0F                     rrc
1966   0A9A CD880A                 call to_seven_segment
1967   0A9D 3210F0                 sta buffer
1968
1969   0AA0 F1                     pop psw
1970   0AA1 CD880A                 call to_seven_segment
1971   0AA4 3211F0                 sta buffer+1
1972
1973   0AA7 E1                     pop h
1974
1975   0AA8 E5                     push h
1976
```

```
0AA9 7D                     mov a,l
0AAA F5                     push psw
0AAB 0F                     rrc
0AAC 0F                     rrc
0AAD 0F                     rrc
0AAE 0F                     rrc
0AAF CD880A                 call to_seven_segment
0AB2 3212F0                 sta buffer+2
0AB5 F1                     pop psw
0AB6 CD880A                 call to_seven_segment
0AB9 3213F0                 sta buffer+3

0ABC E1                     pop h
0ABD C9                     ret


                ; convert current address and data to display buffer
                ;
0ABE 2A2AF0     read_memory: lhld user_PC
0AC1 E5                     push h
0AC2 7C                     mov a,h
0AC3 F5                     push psw
0AC4 0F                     rrc
0AC5 0F                     rrc
0AC6 0F                     rrc
0AC7 0F                     rrc
0AC8 CD880A                 call to_seven_segment
0ACB 3210F0                 sta buffer

0ACE F1                     pop psw
0ACF CD880A                 call to_seven_segment
0AD2 3211F0                 sta buffer+1

0AD5 E1                     pop h

0AD6 E5                     push h

0AD7 7D                     mov a,l
0AD8 F5                     push psw
0AD9 0F                     rrc
0ADA 0F                     rrc
0ADB 0F                     rrc
0ADC 0F                     rrc
0ADD CD880A                 call to_seven_segment
0AE0 3212F0                 sta buffer+2
0AE3 F1                     pop psw
0AE4 CD880A                 call to_seven_segment
0AE7 3213F0                 sta buffer+3

0AEA E1                     pop h
0AEB 7E                     mov a,m   ; read from memory

0AEC F5                     push psw
0AED 0F                     rrc
0AEE 0F                     rrc
0AEF 0F                     rrc
0AF0 0F                     rrc
0AF1 CD880A                 call to_seven_segment
0AF4 3214F0                 sta buffer+4
0AF7 F1                     pop psw


0AF8 CD880A                 call to_seven_segment
0AFB 3215F0                 sta buffer+5

0AFE CD020B                 call mode_indicator

0B01 C9                     ret

0B02            mode_indicator:

0B02 F5                     push psw

0B03 3A26F0                 lda entry_mode
```

```
0B06 FE00                   cpi 0
0B08 C23D0B                 jnz mode1

0B0B 3A15F0                 lda buffer+5   ; mode 0 indicator
0B0E F680                   ori 80h
0B10 3215F0                 sta buffer+5

0B13 3A14F0                 lda buffer+4
0B16 F680           ori 80h
0B18 3214F0         sta buffer+4


0B1B 3A13F0                 lda buffer+3
0B1E E67F                   ani 7fh
0B20 3213F0                 sta buffer+3

0B23 3A12F0                 lda buffer+2
0B26 E67F           ani 7fh
0B28 3212F0         sta buffer+2

0B2B 3A11F0         lda buffer+1
0B2E E67F           ani 7fh
0B30 3211F0         sta buffer+1

0B33 3A10F0         lda buffer
0B36 E67F           ani 7fh
0B38 3210F0         sta buffer




0B3B F1                     pop psw
0B3C C9                     ret

0B3D FE01      mode1:       cpi 1
0B3F C2740B                 jnz mode2

0B42 3A15F0                 lda buffer+5   ; mode 1 indicator
0B45 E67F                   ani 7fh
0B47 3215F0                 sta buffer+5

0B4A 3A14F0         lda buffer+4
0B4D E67F           ani 7fh
0B4F 3214F0         sta buffer+4

0B52 3A13F0                 lda buffer+3
0B55 F680                   ori 80h
0B57 3213F0                 sta buffer+3

0B5A 3A12F0         lda buffer+2
0B5D F680                   ori 80h
0B5F 3212F0                 sta buffer+2

0B62 3A11F0         lda buffer+1
0B65 F680                   ori 80h
0B67 3211F0                 sta buffer+1

0B6A 3A10F0         lda buffer
0B6D F680                   ori 80h
0B6F 3210F0                 sta buffer




0B72 F1                     pop psw
0B73 C9                     ret

0B74 F1        mode2:       pop psw
0B75 C9                     ret

0B76 0E07      cold_boot: mvi c,7
0B78 21940B                lxi h,title


0B7B 1650      cold2:     mvi d,50h
```

```
2129
2130   0B7D CDAE0C    cold1:     call scan
2131   0B80 15                   dcr d
2132   0B81 C27D0B               jnz cold1
2133
2134   0B84 23                   inx h
2135   0B85 0D                   dcr c
2136   0B86 C27B0B               jnz cold2
2137
2138   0B89 2B                   dcx h
2139
2140   0B8A 0E00                 mvi c,0
2141   0B8C CDAE0C    cold3:     call scan
2142   0B8F 0D                   dcr c
2143   0B90 C28C0B               jnz cold3
2144
2145   0B93 C9                   ret
2146
2147   0B94 0000000000title:   dfb 0,0,0,0,0,0,7fh,3fh,7fh,6dh,0,0
2148
2149
2150                 ; display data read from memory pointed to by HL on LED
2151                 ; entry: HL
2152                 ;
2153
2154   0BA0 1605      demo:      mvi d,5
2155
2156   0BA2 CDAE0C    demo1_2:   call scan
2157   0BA5 15                   dcr d
2158   0BA6 C2A20B               jnz demo1_2
2159   0BA9 23                   inx h
2160   0BAA C9                   ret
2161
2162
2163                 ; convert position key to internal key code 0-F for data entry and 10-19H
2164                 ; function keys
2165                 ; entry: A = scan code
2166                 ; exit: A  = internal code
2167
2168   0BAB FE02      get_key_code:  cpi 2
2169   0BAD C2B30B                   jnz code1
2170   0BB0 3E00                     mvi a,0
2171   0BB2 C9                       ret
2172
2173   0BB3 FE0A      code1:         cpi 0ah
2174   0BB5 C2BB0B                   jnz code2
2175   0BB8 3E01                     mvi a,1
2176   0BBA C9                       ret
2177
2178   0BBB FE12      code2:         cpi 12h
2179   0BBD C2C30B                   jnz code3
2180   0BC0 3E02                     mvi a,2
2181   0BC2 C9                       ret
2182
2183   0BC3 FE1A      code3:         cpi 1ah
2184   0BC5 C2CB0B                   jnz code4
2185   0BC8 3E03                     mvi a,3
2186   0BCA C9                       ret
2187
2188   0BCB FE03      code4:         cpi 3
2189   0BCD C2D30B                   jnz code5
2190   0BD0 3E04                     mvi a,4
2191   0BD2 C9                       ret
2192
2193   0BD3 FE0B      code5:         cpi 0bh
2194   0BD5 C2DB0B                   jnz code6
2195   0BD8 3E05                     mvi a,5
2196   0BDA C9                       ret
2197
2198   0BDB FE13      code6:         cpi 13h
2199   0BDD C2E30B                   jnz code7
2200   0BE0 3E06                     mvi a,6
2201   0BE2 C9                       ret
2202
2203   0BE3 FE1B      code7:         cpi 1bh
2204   0BE5 C2EB0B                   jnz code8
```

```
2205   0BE8 3E07                    mvi a,7
2206   0BEA C9                      ret
2207
2208   0BEB FE04      code8:        cpi 4
2209   0BED C2F30B                  jnz code9
2210   0BF0 3E08                    mvi a,8
2211   0BF2 C9                      ret
2212
2213   0BF3 FE0C      code9:        cpi 0ch
2214   0BF5 C2FB0B                  jnz code10
2215   0BF8 3E09                    mvi a,9
2216   0BFA C9                      ret
2217
2218   0BFB FE14      code10:       cpi 14h
2219   0BFD C2030C                  jnz code11
2220   0C00 3E0A                    mvi a,0ah
2221   0C02 C9                      ret
2222
2223   0C03 FE1C      code11:       cpi 1ch
2224   0C05 C20B0C                  jnz code12
2225   0C08 3E0B                    mvi a,0bh
2226   0C0A C9                      ret
2227
2228   0C0B FE05      code12:       cpi 5
2229   0C0D C2130C                  jnz code13
2230   0C10 3E0C                    mvi a,0ch
2231   0C12 C9                      ret
2232
2233   0C13 FE0D      code13:       cpi 0dh
2234   0C15 C21B0C                  jnz code14
2235   0C18 3E0D                    mvi a,0dh
2236   0C1A C9                      ret
2237
2238   0C1B FE15      code14:       cpi 15h
2239   0C1D C2230C                  jnz code15
2240   0C20 3E0E                    mvi a,0eh
2241   0C22 C9                      ret
2242
2243   0C23 FE1D      code15:       cpi 1dh
2244   0C25 C22B0C                  jnz code16
2245   0C28 3E0F                    mvi a,0fh
2246   0C2A C9                      ret
2247
2248   0C2B FE10      code16:       cpi 10h
2249   0C2D C2330C                  jnz code17
2250   0C30 3E10                    mvi a,10h
2251   0C32 C9                      ret
2252
2253   0C33 FE18      code17:       cpi 18h
2254   0C35 C23B0C                  jnz code18
2255   0C38 3E11                    mvi a,11h
2256   0C3A C9                      ret
2257
2258   0C3B FE01      code18:       cpi 1
2259   0C3D C2430C                  jnz code19
2260   0C40 3E12                    mvi a,12h
2261   0C42 C9                      ret
2262
2263   0C43 FE00      code19:       cpi 0
2264   0C45 C24B0C                  jnz code20
2265   0C48 3E13                    mvi a,13h
2266   0C4A C9                      ret
2267
2268   0C4B FE08      code20:       cpi 8
2269   0C4D C2530C                  jnz code21
2270   0C50 3E14                    mvi a,14h
2271   0C52 C9                      ret
2272
2273   0C53 FE09      code21:       cpi 9
2274   0C55 C25B0C                  jnz code22
2275   0C58 3E15                    mvi a,15h
2276   0C5A C9                      ret
2277
2278   0C5B FE11      code22:       cpi 11h
2279   0C5D C2630C                  jnz code23
2280   0C60 3E16                    mvi a,16h
```

```
0C62 C9                     ret

0C63 FE19      code23:      cpi 19h
0C65 C26B0C                 jnz code24
0C68 3E17                   mvi a,17h
0C6A C9                     ret

0C6B FE2E      code24:      cpi 2eh
0C6D C2730C                 jnz code25
0C70 3E18                   mvi a,18h
0C72 C9                     ret

0C73 FE2F      code25:      cpi 2fh
0C75 C27B0C                 jnz code26
0C78 3E19                   mvi a,19h
0C7A C9                     ret

0C7B 3EFF      code26:      mvi a,0ffh
0C7D C9                     ret

               ; scan display and keyboard unitl key was pressed

0C7E           scan_key:  ; mvi d,50       ; number of loop for timeout if key still pre

0C7E           scan_key4:  ;push d         ; save d

0C7E 2110F0                lxi h,buffer
0C81 CDAE0C                call scan
0C84 3A21F0                lda key
0C87 FEFF                  cpi 0ffh
0C89 C27E0C                jnz  scan_key4  ; loop if key still pressed
                           ;pop  d

0C8C F28F0C         jp  scan_key3

0C8F           scan_key2:  ;pop d

                   ; dcr d

                   ; jp scan_key4   ; no repeat function

                                       ; repeat if still pressed when timeout


0C8F CDA70C    scan_key3:  call debounce  ; debounce after released

0C92 2110F0                lxi h,buffer
0C95 CDAE0C    scan_key1:  call scan
0C98 3A21F0                lda key
0C9B FEFF                  cpi 0ffh
0C9D CA950C                jz  scan_key1  ; loop until key will be pressed

0CA0 CDA70C                call debounce


0CA3 CDAB0B                call get_key_code

               ;            call out2x
0CA6 C9                    ret


0CA7 0614      debounce:   mvi b,20
0CA9 05        debounce1:  dcr b
0CAA C2A90C                jnz debounce1
0CAD C9                    ret


               ; subroutine scan keyboard and display
               ; input: hl pointer to buffer
               ; exit: key = scan code
               ;            -1 no key pressed
               ;

0CAE E5        scan:    push h
0CAF C5                 push b
0CB0 D5                 push d
```

```
2357
2358   0CB1 0E06              mvi c,6    ; for 6-digit LED
2359   0CB3 1E00              mvi e,0    ; digit scan code appears at 4-to-10 decoder
2360   0CB5 1600              mvi d,0    ; key position
2361   0CB7 3EFF              mvi a,0ffh ; put -1 to key
2362   0CB9 3221F0            sta key    ; key = -1
2363
2364
2365   0CBC 7B        scan1:  mov a,e
2366   0CBD F6F0              ori 0f0h   ; high nibble must be 1111
2367   0CBF D312              out system_port_c ; active digit first
2368   0CC1 7E                mov a,m        ; load a with [hl]
2369   0CC2 D311              out system_port_b ; then turn segment on
2370
2371   0CC4 060A              mvi b,10    ; delay for transition process
2372   0CC6 05        wait1:  dcr b
2373   0CC7 C2C60C            jnz wait1
2374
2375   0CCA AF          xra a
2376   0CCB D311        out system_port_b ; turn off segment
2377
2378
2379   0CCD DB10              in  system_port_a  ; read input port
2380
2381   0CCF 0608              mvi b,8            ; check all 8-row
2382   0CD1 1F        shift_key: rar             ; rotate right through carry
2383   0CD2 DADB0C            jc  next_key       ; if carry = 1 then no key pressed
2384
2385   0CD5 F5                push psw
2386   0CD6 7A                mov a,d
2387   0CD7 3221F0            sta key            ; save key position
2388   0CDA F1                pop psw
2389
2390   0CDB           next_key:
2391   0CDB 14                inr d               ; next key position
2392
2393   0CDC 05                dcr b               ; until 8-bit was shifted
2394   0CDD C2D10C            jnz shift_key
2395
2396                         ; mvi a,0             ; clear a
2397                         ; out system_port_b   ; turn off led
2398
2399   0CE0 1C                inr e               ; next digit scan code
2400   0CE1 23                inx h               ; next location
2401
2402   0CE2 0D                dcr c               ; next column
2403   0CE3 C2BC0C            jnz scan1
2404
2405   0CE6 CDED0C            call serial_command
2406
2407   0CE9 D1                pop d
2408   0CEA C1                pop b
2409   0CEB E1                pop h
2410   0CEC C9                ret
2411
2412
2413
2414                  ;----------- serial commands with 9600 8n1 terminal --------------------
2415                  ; check if serial buffer has command
2416                  ;
2417
2418   0CED           serial_command:
2419   0CED 3A25F0            lda uart_found
2420   0CF0 FE00              cpi 0
2421   0CF2 CA310D            jz  skip_serial
2422
2423   0CF5 CD4512            call get_command
2424   0CF8 CD500D            call download
2425   0CFB CD320D            call prompting
2426   0CFE CDF112            call hex_dump
2427   0D01 CDEE11            call help
2428   0D04 CDC711            call quick_home
2429   0D07 CDDC11            call io_address
2430   0D0A CDAA11            call new_location
2431   0D0D CD2711            call edit_location
2432   0D10 CDDD10            call jump_to_user_pgm
```

```
0D13 CDC410             call monitor_function
0D16 CD720D             call ascii_print
0D19 CD8010             call fill_memory
0D1C CD4C0F             call register_display
0D1F CD0C0F             call stack_display
0D22 CD1303             call disassemble1
0D25 CD000F             call single_step_
0D28 CDF10E             call print_watch
0D2B CDD60E             call clear_watch
0D2E CDA40D             call set_user_register


0D31            skip_serial:

0D31 C9                 ret

0D32 3A22F0     prompting: lda command
0D35 FE0D                  cpi cr
0D37 C24F0D                jnz exit_prompting

0D3A            send_prompt:

0D3A CDDA12                call new_line
0D3D 2A3CF0                lhld pointer        ; user_PC
0D40 7C                    mov a,h
0D41 CDCB12                call out2x
0D44 7D                    mov a,l
0D45 CDCB12                call out2x
0D48 21E41D                lxi h,prompt_text
0D4B CD5812                call put_str
0D4E C9                    ret

0D4F C9         exit_prompting: ret


                ; command execute
                ; get command from serial port

0D50 3A22F0     download: lda command
0D53 FE6C                 cpi "l"
0D55 C2710D               jnz  exit_download

0D58 CDD514               call clear_bcd1     ; reset bcd counter1
0D5B 3E01                 mvi a,1
0D5D 323AF0               sta temp

0D60 AF                   xra a
0D61 3220F0               sta bcs             ; clear byte chekc sum error

0D64 21CD1D               lxi h,download_text
0D67 CD5812               call put_str
0D6A CD0314               call get_record
0D6D CD3A0D               call send_prompt
0D70 C9                   ret

0D71 C9         exit_download ret

                ; display printable ASCII code, 20H-7FH

0D72 3A22F0     ascii_print: lda command
0D75 FE61                    cpi "a"
0D77 C2A30D                  jnz exit_ascii_print


0D7A 215E1E                  lxi h, ascii_text
0D7D CD5812                  call put_str

0D80 CDDA12                  call new_line
0D83 CDDA12                  call new_line

0D86 2E20                    mvi l,20h
0D88 0E60                    mvi c,96

0D8A            ascii_print1:
```

```
2509
2510   0D8A 7D                  mov a,l
2511   0D8B CD2F12              call cout
2512
2513   0D8E 3E3D                mvi a,"="
2514   0D90 CD2F12              call cout
2515   0D93 7D                  mov a,l
2516   0D94 CDCB12              call out2x
2517   0D97 CDE512              call space
2518   0D9A 2C                  inr l
2519   0D9B 0D                  dcr c
2520   0D9C C28A0D              jnz ascii_print1
2521
2522   0D9F CD3A0D              call send_prompt
2523   0DA2 C9                  ret
2524
2525   0DA3           exit_ascii_print:
2526
2527   0DA3 C9                  ret
2528
2529                  ;------------------- set value to user registers ---------------------
2530                  ; set value to user register AF, BC, DE, HL, SP, PC
2531
2532   0DA4           set_user_register:
2533
2534   0DA4 3A22F0              lda command
2535   0DA7 FE73                cpi "s"
2536   0DA9 C2B80E              jnz exit_set_user
2537
2538   0DAC 210C1F              lxi h, set_register_text
2539   0DAF CD5812              call put_str
2540
2541   0DB2 CD3B12              call cin
2542   0DB5 FE61                cpi "a"
2543   0DB7 C2E00D              jnz set_user1
2544
2545   0DBA CDDA12              call new_line
2546   0DBD 21B01E              lxi h,af_text
2547   0DC0 CD5812              call put_str
2548   0DC3 2A2CF0              lhld user_AF
2549   0DC6 7C                  mov a,h
2550   0DC7 CDCB12              call out2x
2551   0DCA 7D                  mov a,l
2552   0DCB CDCB12              call out2x
2553   0DCE CDEB12              call send_tab
2554   0DD1 CD9E13              call get_hex2
2555   0DD4 67                  mov h,a
2556   0DD5 CD9E13              call get_hex2
2557   0DD8 6F                  mov l,a
2558   0DD9 222CF0              shld user_AF
2559
2560   0DDC CD3A0D              call send_prompt
2561   0DDF C9                  ret
2562
2563   0DE0           set_user1:
2564   0DE0 FE62                cpi "b"
2565   0DE2 C20B0E              jnz set_user2
2566
2567   0DE5 CDDA12              call new_line
2568   0DE8 21B41E              lxi h,bc_text
2569   0DEB CD5812              call put_str
2570   0DEE 2A2EF0              lhld user_BC
2571   0DF1 7C                  mov a,h
2572   0DF2 CDCB12              call out2x
2573   0DF5 7D                  mov a,l
2574   0DF6 CDCB12              call out2x
2575   0DF9 CDEB12              call send_tab
2576   0DFC CD9E13              call get_hex2
2577   0DFF 67                  mov h,a
2578   0E00 CD9E13              call get_hex2
2579   0E03 6F                  mov l,a
2580   0E04 222EF0              shld user_BC
2581
2582   0E07 CD3A0D              call send_prompt
2583   0E0A C9                  ret
2584
```

```
0E0B           set_user2:
0E0B FE64                  cpi "d"
0E0D C2360E                jnz set_user3

0E10 CDDA12                call new_line
0E13 21B81E                lxi h,de_text
0E16 CD5812                call put_str
0E19 2A30F0                lhld user_DE
0E1C 7C                    mov a,h
0E1D CDCB12                call out2x
0E20 7D                    mov a,l
0E21 CDCB12                call out2x
0E24 CDEB12                call send_tab
0E27 CD9E13                call get_hex2
0E2A 67                    mov h,a
0E2B CD9E13                call get_hex2
0E2E 6F                    mov l,a
0E2F 2230F0                shld user_DE

0E32 CD3A0D                call send_prompt
0E35 C9                    ret

0E36           set_user3:
0E36 FE68                  cpi "h"
0E38 C2610E                jnz set_user4

0E3B CDDA12                call new_line
0E3E 21BC1E                lxi h,hl_text
0E41 CD5812                call put_str
0E44 2A32F0                lhld user_HL
0E47 7C                    mov a,h
0E48 CDCB12                call out2x
0E4B 7D                    mov a,l
0E4C CDCB12                call out2x
0E4F CDEB12                call send_tab
0E52 CD9E13                call get_hex2
0E55 67                    mov h,a
0E56 CD9E13                call get_hex2
0E59 6F                    mov l,a
0E5A 2232F0                shld user_HL

0E5D CD3A0D                call send_prompt
0E60 C9                    ret

0E61           set_user4:
0E61 FE73                  cpi "s"
0E63 C28C0E                jnz set_user5

0E66 CDDA12                call new_line
0E69 21C01E                lxi h,sp_text
0E6C CD5812                call put_str
0E6F 2A34F0                lhld user_SP
0E72 7C                    mov a,h
0E73 CDCB12                call out2x
0E76 7D                    mov a,l
0E77 CDCB12                call out2x
0E7A CDEB12                call send_tab
0E7D CD9E13                call get_hex2
0E80 67                    mov h,a
0E81 CD9E13                call get_hex2
0E84 6F                    mov l,a
0E85 2234F0                shld user_SP

0E88 CD3A0D                call send_prompt
0E8B C9                    ret

0E8C           set_user5:
0E8C FE70                  cpi "p"
0E8E C2B70E                jnz set_user6

0E91 CDDA12                call new_line
0E94 21C91E                lxi h,pc_text
0E97 CD5812                call put_str
0E9A 2A2AF0                lhld user_PC
0E9D 7C                    mov a,h
0E9E CDCB12                call out2x
```

```
0EA1 7D                 mov a,l
0EA2 CDCB12             call out2x
0EA5 CDEB12             call send_tab
0EA8 CD9E13             call get_hex2
0EAB 67                 mov h,a
0EAC CD9E13             call get_hex2
0EAF 6F                 mov l,a
0EB0 222AF0             shld user_PC

0EB3 CD3A0D             call send_prompt
0EB6 C9                 ret

0EB7 C9         set_user6: ret

0EB8            exit_set_user:

0EB8 C9                  ret









0EB9            print_watch_ram:

0EB9 CDDA12             call new_line
0EBC 2100F0             lxi h, watch_ram
0EBF 0E10               mvi c,16
0EC1 7C                 mov a,h
0EC2 CDCB12             call out2x
0EC5 7D                 mov a,l
0EC6 CDCB12             call out2x
0EC9            watch1:
0EC9 CDE512             call space
0ECC 7E                 mov a,m
0ECD CDCB12             call out2x
0ED0 23                 inx h
0ED1 0D                 dcr c
0ED2 C2C90E             jnz watch1
0ED5 C9                 ret


                ;----------- clear watch variables-----------------------------------

0ED6 3A22F0     clear_watch: lda command
0ED9 FE63                    cpi "c"
0EDB C2F00E                  jnz exit_clear_watch
0EDE 2100F0                  lxi h, watch_ram
0EE1 0E10                    mvi c,16

0EE3 AF         clear1:      xra a
0EE4 77                      mov m,a
0EE5 23                      inx h
0EE6 0D                      dcr c
0EE7 C2E30E                  jnz clear1

0EEA CDB90E                  call print_watch_ram
0EED CD3A0D                  call send_prompt

0EF0 C9         exit_clear_watch: ret









                ;----------- print watch variables -----------------------------------

0EF1            print_watch:

0EF1 3A22F0             lda command
```

```
0EF4 FE77                cpi "w"
0EF6 C2FF0E              jnz exit_watch

0EF9 CDB90E              call print_watch_ram
0EFC CD3A0D              call send_prompt

0EFF C9         exit_watch: ret


                ;---------- single step running with key space ----------------------
0F00            single_step_:

0F00 3A22F0              lda command
0F03 FE20                cpi " "
0F05 C20B0F              jnz exit_step
0F08 CDBF09              call single_step

0F0B            exit_step:
0F0B C9                  ret

                ;------- display stack area from top of stack to initial -------------

0F0C            stack_display:
0F0C 3A22F0              lda command
0F0F FE6B                cpi "k"
0F11 C24B0F              jnz exit_stack

                ;          lxi h,stack_text
                ;          call put_str

0F14 21391E              lxi h,edit_text2
0F17 CD5812              call put_str
0F1A CDDA12              call new_line

0F1D 2A34F0              lhld user_SP

0F20            stack_display1:

0F20 7C                  mov a,h
0F21 CDCB12              call out2x
0F24 7D                  mov a,l
0F25 CDCB12              call out2x

0F28 CDE512              call space
0F2B CDE512              call space
0F2E 3E5B                mvi a,"["
0F30 CD2F12              call cout

0F33 7E                  mov a,m
0F34 CDCB12              call out2x

0F37 3E5D                mvi a,"]"
0F39 CD2F12              call cout

0F3C CDDA12              call new_line

0F3F 23                  inx h

0F40 119AF0              lxi d, user_stack+32+1  ; load base of user stack

0F43 7D                  mov a,l
0F44 AB                  xra e
0F45 C2200F              jnz stack_display1

0F48 CD3A0D              call send_prompt

0F4B            exit_stack:
0F4B C9                 ret


                ;---------------- registers display ---------------------------------

0F4C            register_display:

0F4C 3A22F0              lda command
```

```
0F4F FE72               cpi "r"
0F51 C27110             jnz exit_register

0F54            register_display1:

                ;           lda uart_found
                ;           cpi 0
                ;           jz exit_register ; exit of no uart


0F54 CDDA12             call new_line

0F57            register_display2:
0F57 CDDA12             call new_line

0F5A 21B01E             lxi h,af_text
0F5D CD5812             call put_str
0F60 2A2CF0             lhld user_AF
0F63 7C                 mov a,h
0F64 CDCB12             call out2x
0F67 7D                 mov a,l
0F68 CDCB12             call out2x
0F6B CDE512             call space

0F6E 21B41E             lxi h,bc_text
0F71 CD5812             call put_str
0F74 2A2EF0             lhld user_BC
0F77 7C                 mov a,h
0F78 CDCB12             call out2x
0F7B 7D                 mov a,l
0F7C CDCB12             call out2x

0F7F CDE512             call space

0F82 21B81E             lxi h,de_text
0F85 CD5812             call put_str
0F88 2A30F0             lhld user_DE
0F8B 7C                 mov a,h
0F8C CDCB12             call out2x
0F8F 7D                 mov a,l
0F90 CDCB12             call out2x
0F93 CDE512             call space

0F96 21BC1E             lxi h,hl_text
0F99 CD5812             call put_str
0F9C 2A32F0             lhld user_HL
0F9F 7C                 mov a,h
0FA0 CDCB12             call out2x
0FA3 7D                 mov a,l
0FA4 CDCB12             call out2x

0FA7 CDE512             call space

0FAA 21C01E             lxi h,sp_text
0FAD CD5812             call put_str
0FB0 2A34F0             lhld user_SP
0FB3 7C                 mov a,h
0FB4 CDCB12             call out2x
0FB7 7D                 mov a,l
0FB8 CDCB12             call out2x

0FBB CDE512             call space

                        ;  lxi h,tos_text
                        ;  call put_str
                        ;  lhld tos
                        ;  mov a,h
                        ;  call out2x
                        ;  mov a,l
                        ;  call out2x
                        ;  call space

0FBE 21C91E               lxi h,pc_text
0FC1 CD5812               call put_str
0FC4 2A2AF0               lhld user_PC
0FC7 7C                   mov a,h
```

```
0FC8 CDCB12             call out2x
0FCB 7D                 mov a,l
0FCC CDCB12             call out2x

0FCF CDE512             call space

0FD2 21E31E             lxi h,sign_text
0FD5 CD5812             call put_str
0FD8 2A2CF0             lhld user_AF
0FDB 7D                 mov a,l
0FDC E680               ani 80h
0FDE C2E90F             jnz register_flag1
0FE1 3E30               mvi a,"0"
0FE3 CD2F12             call cout
0FE6 C3EE0F             jmp register_flag2

0FE9            register_flag1:
0FE9 3E31               mvi a,"1"
0FEB CD2F12             call cout

0FEE            register_flag2:
0FEE CDE512             call space

0FF1 217210             lxi h,zero_text
0FF4 CD5812             call put_str
0FF7 2A2CF0             lhld user_AF
0FFA 7D                 mov a,l
0FFB E640               ani 40h
0FFD C20810             jnz register_flag3
1000 3E30               mvi a,"0"
1002 CD2F12             call cout
1005 C30D10             jmp register_flag4

1008            register_flag3:
1008 3E31               mvi a,"1"
100A CD2F12             call cout

100D            register_flag4:
100D CDE512             call space

1010 217510             lxi h,AC_text
1013 CD5812             call put_str
1016 2A2CF0             lhld user_AF
1019 7D                 mov a,l
101A E610               ani 10h
101C C22710             jnz register_flag5
101F 3E30               mvi a,"0"
1021 CD2F12             call cout
1024 C32C10             jmp register_flag6

1027            register_flag5:
1027 3E31               mvi a,"1"
1029 CD2F12             call cout

102C            register_flag6:
102C CDE512             call space

102F 217910             lxi h,P_text
1032 CD5812             call put_str
1035 2A2CF0             lhld user_AF
1038 7D                 mov a,l
1039 E604               ani 4
103B C24610             jnz register_flag7
103E 3E30               mvi a,"0"
1040 CD2F12             call cout
1043 C34B10             jmp register_flag8

1046            register_flag7:
1046 3E31               mvi a,"1"
1048 CD2F12             call cout

104B            register_flag8:
104B CDE512             call space

104E 217C10             lxi h,CY_text
1051 CD5812             call put_str
```

```
2965   1054 2A2CF0              lhld user_AF
2966   1057 7D                  mov a,l
2967   1058 E601                ani 1
2968   105A C26510              jnz register_flag9
2969   105D 3E30                mvi a,"0"
2970   105F CD2F12              call cout
2971   1062 C36A10              jmp register_flag10
2972
2973   1065           register_flag9:
2974   1065 3E31                mvi a,"1"
2975   1067 CD2F12              call cout
2976
2977   106A           register_flag10:
2978   106A CDE512              call space
2979
2980   106D CD3A0D              call send_prompt
2981   1070 C9                  ret
2982
2983   1071           exit_register:
2984   1071 C9                  ret
2985
2986   1072 5A3D00    zero_text   dfb "Z=",0
2987   1075 41433D00  AC_text     dfb "AC=",0
2988   1079 503D00    P_text      dfb "P=",0
2989   107C 43593D00  CY_text     dfb "CY=",0
2990
2991
2992                  ;----------------- fill constant to memory --------------------------
2993
2994   1080           fill_memory:
2995
2996   1080 3A22F0              lda command
2997   1083 FE66                cpi "f"
2998   1085 C2C310              jnz exit_fill
2999   1088 216F1E              lxi h,fill_text1
3000   108B CD5812              call put_str
3001
3002   108E CD8513              call get_hex1
3003   1091 67                  mov h,a
3004   1092 CD8513              call get_hex1
3005   1095 6F                  mov l,a
3006   1096 E5                  push h         ; save begin address to stack
3007
3008   1097 21801E              lxi h,fill_text2
3009   109A CD5812              call put_str
3010
3011   109D CD8513              call get_hex1
3012   10A0 67                  mov h,a
3013   10A1 CD8513              call get_hex1
3014   10A4 6F                  mov l,a
3015   10A5 E5                  push h        ; save end address to stack
3016
3017   10A6 21901E              lxi h,fill_text3
3018   10A9 CD5812              call put_str
3019   10AC CD8513              call get_hex1
3020
3021   10AF 47                  mov b,a       ; byte save to B
3022
3023   10B0 D1                  pop d         ; end address in DE
3024
3025   10B1 E1                  pop h         ; begin address in HL
3026
3027   10B2           fill_memory1:
3028
3029   10B2 78                  mov a,b
3030   10B3 77                  mov m,a
3031   10B4 23                  inx h
3032
3033   10B5 7D                  mov a,l
3034   10B6 BB                  cmp e
3035   10B7 C2B210              jnz fill_memory1
3036
3037   10BA 7C                  mov a,h
3038   10BB BA                  cmp d
3039   10BC C2B210              jnz fill_memory1
3040
```

```
3041
3042   10BF CD3A0D            call send_prompt
3043   10C2 C9                ret
3044
3045   10C3           exit_fill:
3046
3047   10C3 C9                ret
3048
3049
3050
3051
3052                  ;---------------- monitor function list ------------------------------
3053
3054   10C4           monitor_function:
3055
3056   10C4 3A22F0            lda command
3057   10C7 FE6D              cpi "m"
3058   10C9 C2DC10            jnz exit_monitor
3059
3060   10CC CDDA12            call new_line
3061   10CF 216C22            lxi h,monitor_text
3062   10D2 CD5812            call put_str
3063   10D5 CDDA12            call new_line
3064   10D8 CD3A0D            call send_prompt
3065   10DB C9                ret
3066
3067   10DC           exit_monitor:
3068   10DC C9                ret
3069
3070                  ;----------------------- jump to user program --------------------
3071
3072   10DD 3A22F0    jump_to_user_pgm:   lda command
3073   10E0 FE6A                  cpi "j"
3074   10E2 C22611                jnz exit_jump
3075
3076   10E5 21471E                lxi h, jump_text1
3077   10E8 CD5812                call put_str
3078
3079   10EB 2A2AF0                lhld user_PC
3080   10EE 7C                    mov a,h
3081   10EF CDCB12                call out2x
3082   10F2 7D                    mov a,l
3083   10F3 CDCB12                call out2x
3084
3085   10F6 21591E                lxi h,jump_text2
3086   10F9 CD5812                call put_str
3087
3088   10FC CD9E13                call get_hex2
3089
3090   10FF F5                    push psw
3091
3092   1100 3A23F0                lda flag1
3093   1103 E601                  ani 1
3094   1105 C21711                jnz skip_load_PC
3095
3096   1108 F1                    pop psw
3097
3098   1109 67                    mov h,a
3099   110A CD9E13                call get_hex2
3100   110D 6F                    mov l,a
3101   110E 222AF0                shld user_PC
3102   1111 CDDA12                call new_line
3103   1114 C39B09                jmp go
3104
3105   1117           skip_load_PC:
3106   1117 F1                    pop psw
3107   1118 3A23F0                lda flag1
3108   111B E6FE                  ani 0feh
3109   111D 3223F0                sta flag1
3110   1120 CDDA12                call new_line
3111   1123 C39B09                jmp go
3112
3113   1126 C9        exit_jump:      ret
3114
3115                  ;------------ edit memory ----------------------------------------
3116
```

```
3117
3118   1127 3A22F0    edit_location: lda command
3119   112A FE65                     cpi "e"
3120   112C C2A911                   jnz exit_edit
3121
3122   112F 21F61D                   lxi h, edit_text
3123   1132 CD5812                   call put_str
3124   1135 CD8513                   call get_hex1
3125   1138 67                       mov h,a
3126   1139 CD8513                   call get_hex1
3127   113C 6F                       mov l,a
3128   113D 223CF0                   shld pointer    ;user_PC
3129
3130   1140 210E1E                   lxi h, edit_text1
3131   1143 CD5812                   call put_str
3132
3133   1146 21391E                   lxi h, edit_text2
3134   1149 CD5812                   call put_str
3135
3136   114C CDDA12    edit1:         call new_line
3137
3138   114F 2A3CF0                   lhld pointer    ;user_PC
3139   1152 7C                       mov a,h
3140   1153 CDCB12                   call out2x
3141   1156 7D                       mov a,l
3142   1157 CDCB12                   call out2x
3143   115A CDE512                   call space
3144   115D CDE512                   call space
3145   1160 3E5B                     mvi a,"["
3146   1162 CD2F12                   call cout
3147   1165 7E                       mov a,m
3148   1166 CDCB12                   call out2x
3149   1169 3E5D                     mvi a,"]"
3150   116B CD2F12                   call cout
3151
3152   116E CDE512                   call space
3153
3154   1171 CD9E13                   call get_hex2
3155
3156   1174 F5                       push psw
3157
3158   1175 3A23F0                   lda flag1
3159   1178 E601                     ani 1
3160   117A C29A11                   jnz exit_edit1    ; Enter key?
3161
3162   117D 3A23F0                   lda flag1
3163   1180 E602                     ani 2
3164   1182 C28A11                   jnz skip_edit1    ; SPACE key?
3165
3166   1185 F1                       pop psw
3167
3168   1186 77                       mov m,a
3169   1187 C39311                   jmp skip_edit2
3170
3171   118A F1        skip_edit1: pop psw
3172
3173   118B 3A23F0                   lda flag1
3174   118E E6FD                     ani 0fdh
3175   1190 3223F0                   sta flag1
3176
3177   1193           skip_edit2:
3178   1193 23                       inx h
3179   1194 223CF0                   shld pointer    ;user_PC
3180   1197 C34C11                   jmp edit1
3181
3182   119A F1        exit_edit1: pop psw
3183
3184   119B 3A23F0                lda flag1
3185   119E E6FE                  ani 0feh
3186   11A0 3223F0                sta flag1
3187
3188   11A3 CDDA12                call new_line
3189   11A6 CD3A0D                call send_prompt
3190
3191   11A9 C9        exit_edit:  ret
3192
```

```
3193
3194   11AA 3A22F0    new_location: lda command
3195   11AD FE6E                    cpi "n"
3196   11AF C2C611                  jnz exit_new_location
3197   11B2 21E61D                  lxi h,new_text
3198   11B5 CD5812                  call put_str
3199   11B8 CD8513                  call get_hex1
3200   11BB 67                      mov h,a
3201   11BC CD8513                  call get_hex1
3202   11BF 6F                      mov l,a
3203   11C0 223CF0                  shld pointer       ; user_PC
3204   11C3 CD3A0D                  call send_prompt
3205
3206   11C6           exit_new_location:
3207
3208   11C6 C9                      ret
3209
3210
3211
3212
3213   11C7           quick_home:
3214   11C7 3A22F0                  lda command
3215   11CA FE71                    cpi "q"
3216   11CC C2DB11                  jnz exit_quick_home
3217
3218   11CF 210081                  lxi h,home_address
3219   11D2 222AF0                  shld user_PC
3220   11D5 223CF0                  shld pointer
3221   11D8 CD3A0D                  call send_prompt
3222
3223   11DB           exit_quick_home:
3224   11DB C9                      ret
3225
3226                  ; i/o address map
3227
3228   11DC 3A22F0    io_address:  lda command
3229   11DF FE69                    cpi "i"
3230   11E1 C2ED11                  jnz exit_io
3231
3232   11E4 210321                  lxi h,io_text
3233   11E7 CD5812                  call put_str
3234   11EA CD3A0D                  call send_prompt
3235
3236   11ED C9        exit_io:  ret
3237
3238
3239                  ; help listing
3240
3241   11EE 3A22F0    help:     lda command
3242   11F1 FE3F                    cpi "?"
3243   11F3 C20512                  jnz exit_help
3244
3245   11F6 216F12                  lxi h,prompt3
3246   11F9 CD6612                  call alt_put_str
3247   11FC 216E1F                  lxi h,help_text1
3248   11FF CD5812                  call put_str
3249   1202 CD3A0D                  call send_prompt
3250
3251   1205 C9        exit_help:  ret
3252
3253
3254                  ; initialize 16C550 uart to 9600 8n1 with 2MHz clock
3255                  ; 2MHz/13 = 153846Hz
3256
3257   1206           init_uart:
3258
3259   1206 3E83              mvi a,83h
3260   1208 D343              out uart_lcr         ; set DLAB bit to access divider
3261
3262   120A 3E0D              mvi a,13
3263   120C D340              out uart_divisor_lsb
3264   120E 3E00              mvi a,0
3265   1210 D341              out uart_divisor_msb ; 2MHz/13 = 153846 Hz
3266                                               ; 153846Hz/16 = 9615Hz
3267   1212 3E07              mvi a,7
3268   1214 D342              out uart_fifo        ; init fifo and clear all buffers
```

```
3269   1216 3E03              mvi a,03h
3270   1218 D343              out uart_lcr         ; clar DLAB
3271
3272                  ; check uart line status, if the byte is FF then no uart
3273                  ;
3274                  ;
3275   121A AF                xra a
3276   121B D347              out uart_scr        ; check if there is uart
3277   121D DB47              in uart_scr
3278   121F FE00              cpi 0
3279   1221 CA2912            jz found
3280   1224 AF                xra a
3281   1225 3225F0            sta uart_found
3282   1228 C9                ret
3283
3284   1229 3E01      found   mvi a,1
3285   122B 3225F0            sta uart_found
3286   122E C9                ret
3287
3288   122F 47        cout:   mov b,a              ; save a
3289
3290   1230 DB45      cout1:  in uart_line_status
3291   1232 E620              ani 20h               ; transmitter ready?
3292   1234 CA3012            jz cout1
3293
3294   1237 78                mov a,b             ; restore a
3295   1238 D340              out uart_buffer
3296   123A C9                ret
3297
3298   123B DB45      cin:    in uart_line_status
3299   123D E601              ani 1                  ; data available?
3300   123F CA3B12            jz  cin
3301   1242 DB40              in uart_buffer
3302   1244 C9                ret
3303
3304
3305   1245 DB45      get_command: in uart_line_status
3306   1247 E601                   ani 1
3307   1249 CA5212                 jz no_data
3308   124C DB40                   in uart_buffer
3309   124E 3222F0                 sta command      ; command = ASCII code
3310   1251 C9                     ret
3311
3312   1252 3EFF      no_data:     mvi a,0ffh       ; command == -1
3313   1254 3222F0                 sta command
3314   1257 C9                     ret
3315
3316
3317                  ; print string terminated by 0
3318                  ; input: HL
3319
3320   1258 7E        put_str:  mov a,m    ; get A from [HL]
3321   1259 FE00                cpi 0
3322   125B C25F12              jnz put_str1
3323   125E C9                  ret
3324
3325   125F CD2F12    put_str1: call cout
3326   1262 23                  inx h
3327   1263 F25812              jp put_str
3328
3329   1266 7E        alt_put_str:  mov a,m    ; get A from [HL]
3330   1267 EEAA                    xri 0aah
3331   1269 FE00                    cpi 0
3332   126B C2A512                  jnz put_str2
3333   126E C9                      ret
3334
3335   126F A7A0A0E7FEprompt3:    dfb 0A7h,0A0h,0A0h,0E7h,0FEh,0E1h,087h,092h,09Fh,08Ah,092h,09
3336   127F E3E9F8E5FA            dfb 0E3h,0E9h,0F8h,0E5h,0FAh,0F8h,0E5h,0E9h,0EFh,0F9h,0F9h,0E
3337   128F EBE3E4E3E4            dfb 0EBh,0E3h,0E4h,0E3h,0E4h,0EDh,08Ah,0E1h,0E3h,0FEh,08Ah,08
3338   129F E6FA83A7A0            dfb 0E6h,0FAh,083h,0A7h,0a0h,0aah
3339
3340   12A5 CD2F12    put_str2: call cout
3341   12A8 23                  inx h
3342   12A9 F26612              jp alt_put_str
3343
3344
```

```
3345
3346   12AC 21581D    send_prompt1: lxi h,prompt1
3347   12AF CD5812                  call put_str
3348   12B2 C9                      ret
3349
3350   12B3 216F12    send_prompt3: lxi h,prompt3
3351   12B6 CD6612                   call alt_put_str
3352   12B9 C9                       ret
3353
3354
3355
3356   12BA F5        out1x:         push psw
3357   12BB E60F                     ani  0fh
3358   12BD C630                     adi  "0"
3359   12BF FE3A                     cpi  3Ah
3360   12C1 DAC612                   jc  out1x1
3361   12C4 C607                     adi  7
3362
3363   12C6 CD2F12    out1x1:        call cout
3364   12C9 F1                       pop psw
3365   12CA C9                ret
3366
3367   12CB           out2x:
3368   12CB 0F                       rrc
3369   12CC 0F                       rrc
3370   12CD 0F                       rrc
3371   12CE 0F                       rrc
3372   12CF CDBA12            call out1x
3373   12D2 07                       rlc
3374   12D3 07                       rlc
3375   12D4 07                       rlc
3376   12D5 07                       rlc
3377   12D6 CDBA12            call out1x
3378   12D9 C9                ret
3379
3380                  ; new_line
3381
3382   12DA 3E0D      new_line:   mvi a,cr
3383   12DC CD2F12           call cout
3384   12DF 3E0A                   mvi a,lf
3385   12E1 CD2F12           call cout
3386   12E4 C9               ret
3387
3388   12E5 3E20      space:       mvi a," "
3389   12E7 CD2F12                 call cout
3390   12EA C9                     ret
3391
3392   12EB 3E09      send_tab:    mvi a,9
3393   12ED CD2F12                 call cout
3394   12F0 C9                     ret
3395
3396
3397
3398   12F1 3A22F0    hex_dump:  lda command
3399   12F4 FE68                 cpi "h"
3400   12F6 C25013               jnz exit_hex_dump
3401
3402   12F9 CDDA12               call new_line
3403
3404   12FC 0E08                 mvi c,8        ; 8 lines
3405
3406   12FE C5        hex_dump2:  push b
3407   12FF CDDA12               call new_line
3408   1302 2A3CF0               lhld pointer     ;user_PC
3409   1305 7C                   mov a,h
3410   1306 CDCB12               call out2x
3411   1309 7D                   mov a,l
3412   130A CDCB12               call out2x
3413   130D CDE512               call space
3414
3415   1310 0E10                 mvi c,16
3416
3417   1312 CDE512    hex_dump1:  call space
3418   1315 7E                   mov a,m
3419   1316 CDCB12               call out2x
3420   1319 23                   inx h
```

```
3421   131A 0D                 dcr c
3422   131B C21213             jnz hex_dump1
3423
3424   131E CDE512             call space
3425   1321 CDE512             call space
3426   1324 CDE512             call space
3427
3428                   ; print ASCII representation 20H-7FH
3429                   ; outside such range, print . instead
3430
3431   1327 11F0FF             lxi d,0FFF0h   ; load DE with -16
3432   132A 19                 dad d          ; ADD HL,DE
3433
3434   132B 0E10               mvi c,16
3435
3436   132D 7E        hex_dump5: mov a,m
3437
3438   132E FE20               cpi 20h        ; <20H?
3439   1330 DA3813             jc hex_dump3
3440   1333 FE80               cpi 80h
3441   1335 DA3A13             jc hex_dump4
3442   1338 3E2E      hex_dump3: mvi a,"."
3443   133A CD2F12    hex_dump4: call cout
3444
3445   133D 23                 inx h
3446   133E 0D                 dcr c
3447   133F C22D13             jnz hex_dump5
3448
3449   1342 223CF0             shld pointer   ;user_PC
3450
3451   1345 C1                 pop  b
3452   1346 0D                 dcr c
3453   1347 C2FE12             jnz hex_dump2
3454
3455   134A CDDA12             call new_line
3456   134D CD3A0D             call send_prompt
3457
3458   1350 C9        exit_hex_dump:  ret
3459
3460
3461   1351 210081    dump_memory: lxi h,8100h
3462   1354 0E64                   mvi c,100   ; 100 bytes display
3463
3464   1356 CDDA12                 call new_line
3465   1359 7E        dump1:       mov a,m
3466   135A CDCB12           call out2x
3467   135D CDE512           call space
3468   1360 23                     inx h
3469   1361 0D                     dcr c
3470   1362 C25913                 jnz dump1
3471   1365 C9               ret
3472
3473                   ; convert ASCII letter to one nibble 0-F
3474                   ; 0-9 -> al-30
3475                   ; A-F -> al-7
3476                   ; entry: A
3477                   ; exit: A
3478
3479   1366 D630      to_hex:  sui "0"
3480   1368 FE0A               cpi 10
3481   136A DA7113             jc zero_nine
3482   136D E6DF               ani 11011111b
3483   136F D607               sui 7        ; convert to A-F
3484   1371           zero_nine:
3485
3486   1371 C9                 ret
3487
3488                   ; read two ASCII bytes and convert them to one bye 8-bit data
3489                   ; exit: A
3490                   ; used: A, E
3491
3492   1372 CD3B12    get_hex: call cin
3493   1375 CD6613      call to_hex
3494   1378 0F                 rrc
3495   1379 0F                 rrc
3496   137A 0F                 rrc
```

```
3497   137B 0F                 rrc
3498   137C 5F                 mov e,a
3499   137D CD3B12             call cin
3500   1380 CD6613       call to_hex
3501   1383 83                 add e
3502   1384 C9           ret
3503
3504                   ; read two ASCII bytes echo to screen and convert them to one bye 8-bit d
3505                   ; exit: A
3506
3507   1385 CD3B12     get_hex1: call cin
3508   1388 CD2F12               call cout
3509   138B CD6613       call to_hex
3510   138E 0F                 rrc
3511   138F 0F                 rrc
3512   1390 0F                 rrc
3513   1391 0F                 rrc
3514   1392 5F                 mov e,a
3515   1393 CD3B12             call cin
3516   1396 CD2F12             call cout
3517   1399 CD6613             call to_hex
3518   139C 83                 add e
3519   139D C9           ret
3520
3521                   ; read two ASCII bytes echo to screen and convert them to one bye 8-bit d
3522                   ; exit: A
3523
3524   139E 3A23F0     get_hex2: lda flag1
3525   13A1 E6FC                 ani 0fch      ; clear flag1.1 and flag1.0
3526   13A3 3223F0               sta flag1
3527
3528   13A6 CD3B12             call cin
3529   13A9 FE0D               cpi cr
3530   13AB CAF113             jz  exit_get_hex2
3531
3532   13AE FE20               cpi " "
3533   13B0 CAFA13             jz exit_get_hex3
3534
3535   13B3 FE30               cpi 30h        ; hex must be 0-9 and A-F
3536   13B5 DA9E13             jc  get_hex2
3537
3538   13B8 FE40               cpi 40h
3539   13BA DAC713             jc  ascii_0_9
3540
3541   13BD FE61               cpi 97         ; < 97?
3542   13BF DA9E13             jc  get_hex2
3543
3544   13C2 FE67               cpi 103        ; >= 103?
3545   13C4 D29E13             jnc get_hex2
3546
3547   13C7            ascii_0_9:
3548
3549   13C7 CD2F12             call cout
3550   13CA CD6613       call to_hex
3551   13CD 0F                 rrc
3552   13CE 0F                 rrc
3553   13CF 0F                 rrc
3554   13D0 0F                 rrc
3555   13D1 5F                 mov e,a
3556
3557   13D2            get_2nd_hex:
3558
3559   13D2 CD3B12             call cin
3560
3561   13D5 FE30               cpi 30h        ; hex must be 0-9 and A-F
3562   13D7 DAD213             jc  get_2nd_hex
3563
3564   13DA FE40               cpi 40h
3565   13DC DAE913             jc  ok_0_9
3566
3567   13DF FE61               cpi 97         ; < 97?
3568   13E1 DAD213             jc  get_2nd_hex
3569
3570   13E4 FE67               cpi 103        ; >= 103?
3571   13E6 D2D213             jnc get_2nd_hex
3572
```

```
13E9           ok_0_9:
13E9 CD2F12             call cout
13EC CD6613             call to_hex
13EF 83                 add e
13F0 C9           ret

13F1           exit_get_hex2:

13F1 3A23F0             lda flag1
13F4 F601               ori 1
13F6 3223F0             sta flag1  ; Q key has been pressed
13F9 C9                 ret

13FA           exit_get_hex3:

13FA 3A23F0              lda flag1
13FD F602               ori 2
13FF 3223F0             sta flag1 ; SPACE key has been pressed
1402 C9                 ret


                ; add check sum


                ; get record, write to SRAM and jump to 8000h
                ; entry: A= byte received, B= byte check sum

                add_bcs:    macro    ; add accumulator with byte check sum stored in B
                            push psw
                            add b
                            mov b,a
                            pop psw
                            endm

001B =          esc     equ 1bh

1403 CD3B12     get_record: call cin
1406 FE1B                   cpi 27
1408 CA5A14                 jz  esc_quit

140B FE3A                   cpi ":"
140D C20314                 jnz get_record      ; wait until begin of record found

1410 0600                   mvi b,0             ; byte check sum

1412 CD7213         call get_hex ; get number of byte
1415 4F                     mov c,a             ; put to c

1416                        add_bcs
1416 F5                     push psw
1417 80                     add b
1418 47                     mov b,a
1419 F1                     pop psw
141A                        endm

141A CD7213         call get_hex ; get destination address, put to bx register
141D 67                     mov  h,a            ; save high byte

141E                        add_bcs
141E F5                     push psw
141F 80                     add b
1420 47                     mov b,a
1421 F1                     pop psw
1422                        endm

1422 CD7213                 call get_hex
1425 6F                     mov  l,a            ; and low byte

1426                        add_bcs
1426 F5                     push psw
1427 80                     add b
1428 47                     mov b,a
1429 F1                     pop psw
142A                        endm

142A CD7213         call get_hex
```

```
3649
3650   142D                     add_bcs
3651   142D F5                  push psw
3652   142E 80                  add b
3653   142F 47                  mov b,a
3654   1430 F1                  pop psw
3655   1431                     endm
3656
3657   1431 FE01                cpi  1              ; end of record type is 01 ?
3658   1433 C25B14              jnz  data_record    ; jump if not 01
3659
3660   1436 CD3B12    wait_cr:  call cin
3661   1439 FE0D                cpi  cr
3662   143B C23614              jnz wait_cr     ; until end of record sending! with cr detect
3663
3664   143E 3EFF                mvi a, 0ffh     ; turn speaker off
3665   1440 D312                out system_port_c
3666   1442 AF                  xra a
3667   1443 D300                out 0           ; turn off GPIO
3668
3669   1445 CDB914              call print_bcd1
3670   1448 CDE512              call space
3671   144B 3A20F0              lda bcs
3672   144E CDC901              call pint8u
3673   1451 21991E              lxi h,error_text
3674   1454 CD5812              call put_str
3675   1457 CDDA12              call new_line
3676
3677   145A C9        esc_quit: ret
3678
3679
3680   145B CD7213    data_record: call get_hex       ; get data byte
3681   145E 77                   mov m,a            ; save to SRAM at [HL]
3682
3683   145F                      add_bcs
3684   145F F5                  push psw
3685   1460 80                  add b
3686   1461 47                  mov b,a
3687   1462 F1                  pop psw
3688   1463                     endm
3689
3690   1463 CD8914               call inc_bcd1
3691
3692                         ;    ori 7fh
3693                         ;    out system_port_c   ; make buzzer sound
3694
3695
3696   1466 23                   inx h              ; next location
3697
3698   1467 0D                   dcr c
3699   1468 C25B14               jnz data_record    ; until c = 0
3700
3701   146B 78                   mov a,b
3702   146C 2F                   cma
3703   146D 47                   mov b,a
3704   146E 04                   inr b              ; compute two's complement
3705
3706   146F CD7213               call get_hex       ; get check sum
3707
3708   1472 B8                   cmp b
3709   1473 CA7D14               jz  skip_error
3710
3711   1476 3A20F0               lda bcs
3712   1479 3C                   inr a
3713   147A 3220F0               sta bcs
3714
3715   147D           skip_error:
3716   147D 3A3AF0               lda temp          ; then shift into temp8
3717   1480 07                   rlc
3718   1481 323AF0               sta temp
3719   1484 D300                 out 0             ; send to GPIO
3720
3721   1486 C30314              jmp get_record  ; back to next record
3722
3723
3724                  ;---------------- increment BCD counter1 -------------------------
```

```
3725
3726   1489 E5        inc_bcd1:    push h
3727
3728   148A 2141F0                 lxi h,bcd_counter1
3729   148D AF                     xra a
3730
3731   148E 7E                     mov a,m
3732   148F C601                   adi 1
3733   1491 27                     daa
3734   1492 77                     mov m,a
3735   1493 23                     inx h
3736
3737   1494 7E                     mov a,m
3738   1495 CE00                   aci 0
3739   1497 27                     daa
3740   1498 77                     mov m,a
3741   1499 23                     inx h
3742   149A 7E                     mov a,m
3743   149B CE00                   aci 0
3744   149D 27                     daa
3745   149E 77                     mov m,a
3746
3747   149F E1                     pop h
3748
3749   14A0 C9                     ret
3750
3751   14A1 E5        inc_bcd2:    push h
3752
3753   14A2 2144F0                 lxi h,bcd_counter2
3754   14A5 AF                     xra a
3755
3756   14A6 7E                     mov a,m
3757   14A7 C601                   adi 1
3758   14A9 27                     daa
3759   14AA 77                     mov m,a
3760   14AB 23                     inx h
3761
3762   14AC 7E                     mov a,m
3763   14AD CE00                   aci 0
3764   14AF 27                     daa
3765   14B0 77                     mov m,a
3766   14B1 23                     inx h
3767   14B2 7E                     mov a,m
3768   14B3 CE00                   aci 0
3769   14B5 27                     daa
3770   14B6 77                     mov m,a
3771
3772   14B7 E1                     pop h
3773
3774   14B8 C9                     ret
3775
3776
3777   14B9           print_bcd1:
3778   14B9 2143F0                 lxi h,bcd_counter1+2
3779   14BC 7E                     mov a,m
3780   14BD CDCB12                 call out2x
3781   14C0 2142F0                 lxi h,bcd_counter1+1
3782   14C3 7E                     mov a,m
3783   14C4 CDCB12                 call out2x
3784   14C7 2141F0                 lxi h,bcd_counter1
3785   14CA 7E                     mov a,m
3786   14CB CDCB12                 call out2x
3787   14CE 21E61E                 lxi h,byte_text
3788   14D1 CD5812                 call put_str
3789   14D4 C9                     ret
3790
3791   14D5 210000    clear_bcd1:  lxi h,0
3792   14D8 2241F0                 shld bcd_counter1
3793   14DB 2242F0                 shld bcd_counter1+1
3794   14DE 2243F0                 shld bcd_counter1+2
3795   14E1 C9                     ret
3796
3797   14E2           print_bcd2:
3798   14E2 2146F0                 lxi h,bcd_counter2+2
3799   14E5 7E                     mov a,m
3800   14E6 CDCB12                 call out2x
```

```
3801   14E9 2145F0               lxi h,bcd_counter2+1
3802   14EC 7E                   mov a,m
3803   14ED CDCB12               call out2x
3804   14F0 2144F0               lxi h,bcd_counter2
3805   14F3 7E                   mov a,m
3806   14F4 CDCB12               call out2x
3807   14F7 21E61E               lxi h,byte_text
3808   14FA CD5812               call put_str
3809   14FD C9                   ret
3810
3811   14FE 210000    clear_bcd2:  lxi h,0
3812   1501 2244F0               shld bcd_counter2
3813   1504 2245F0               shld bcd_counter2+1
3814   1507 2246F0               shld bcd_counter2+2
3815   150A C9                   ret
3816
3817                         ; constants
3818
3819   150B           INS_TABLE:
3820
3821   150B 0B17              DWL C0 ;     "NOP",TAB,RS            ; 00
3822   150D 1017              DWL C1 ;     "LXI",TAB,"B,",RS       ; 01
3823   150F 1717              DWL C2 ;     "STAX",TAB,"B",RS       ; 02
3824   1511 1E17              DWL C3 ;     "INX",TAB,"B",RS        ; 03
3825   1513 2417              DWL C4 ;     "INR",TAB,"B",RS        ; 04
3826   1515 2A17              DWL C5 ;     "DCR",TAB,"B",RS        ; 05
3827   1517 3017              DWL C6 ;     "MVI",TAB,"B,",RS       ; 06
3828   1519 3717              DWL C7 ;     "RLC",TAB,RS            ; 07
3829   151B 3C17              DWL C8 ;     "DFB",TAB,RS            ; 08
3830   151D 4117              DWL C9 ;     "DAD",TAB,"B",RS        ; 09
3831   151F 4717              DWL CA ;     "LDAX",TAB,"B",RS       ; 0A
3832   1521 4E17              DWL CB ;     "DCX",TAB,"B",RS        ; 0B
3833   1523 5417              DWL CC ;     "INR",TAB,"C",RS        ; 0C
3834   1525 5A17              DWL CD ;     "DCR",TAB,"C",RS        ; 0D
3835   1527 6017              DWL CE ;     "MVI",TAB,"C,",RS       ; 0E
3836   1529 6717              DWL CF ;     "RRC",TAB,RS            ; 0F
3837   152B 6C17              DWL C10 ;    "DFB",TAB,RS            ; 10
3838   152D 7117              DWL C11 ;    "LXI",TAB,"D,",RS       ; 11
3839   152F 7817              DWL C12 ;    "STAX",TAB,"D",RS       ; 12
3840   1531 7F17              DWL C13 ;    "INX",TAB,"D",RS        ; 13
3841   1533 8517              DWL C14 ;    "INR",TAB,"D",RS        ; 14
3842   1535 8B17              DWL C15 ;    "DCR",TAB,"D",RS        ; 15
3843   1537 9117              DWL C16 ;    "MVI",TAB,"D,",RS       ; 16
3844   1539 9817              DWL C17 ;    "RAL",TAB,RS            ; 17
3845   153B 9D17              DWL C18 ;    "DFB",TAB,RS             ; 18
3846   153D A217              DWL C19 ;    "DAD",TAB,"D",RS        ; 19
3847   153F A817              DWL C1A ;    "LDAX",TAB,"D",RS       ; 1A
3848   1541 AF17              DWL C1B ;    "DCX",TAB,"D",RS        ; 1B
3849   1543 B517              DWL C1C ;    "INR",TAB,"E",RS        ; 1C
3850   1545 BB17              DWL C1D ;    "DCR",TAB,"E",RS        ; 1D
3851   1547 C117              DWL C1E ;    "MVI",TAB,"E,",RS       ; 1E
3852   1549 C817              DWL C1F ;    "RAR",TAB,RS            ; 1F
3853   154B CD17              DWL C20 ;    "RIM",TAB,RS            ; 20
3854   154D D217              DWL C21 ;    "LXI",TAB,"H,",RS       ; 21
3855   154F D917              DWL C22 ;    "SHLD",TAB,RS           ; 22
3856   1551 DF17              DWL C23 ;    "INX",TAB,"H",RS        ; 23
3857   1553 E517              DWL C24 ;    "INR",TAB,"H",RS        ; 24
3858   1555 EB17              DWL C25 ;    "DCR",TAB,"H",RS        ; 25
3859   1557 F117              DWL C26 ;    "MVI",TAB,"H,",RS       ; 26
3860   1559 F817              DWL C27 ;    "DAA",TAB,RS            ; 27
3861   155B FD17              DWL C28 ;    "DFB",TAB,RS             ; 28
3862   155D 0218              DWL C29 ;    "DAD",TAB,"H",RS        ; 29
3863   155F 0818              DWL C2A ;    "LHLD",TAB,RS           ; 2A
3864   1561 0E18              DWL C2B ;    "DCX",TAB,"H",RS        ; 2B
3865   1563 1418              DWL C2C ;    "INR",TAB,"L",RS        ; 2C
3866   1565 1A18              DWL C2D ;    "DCR",TAB,"L",RS        ; 2D
3867   1567 2018              DWL C2E ;    "MVI",TAB,"L,",RS       ; 2E
3868   1569 2718              DWL C2F ;    "CMA",TAB,RS            ; 2F
3869   156B 2C18              DWL C30 ;    "SIM",TAB,RS            ; 30
3870   156D 3118              DWL C31 ;    "LXI",TAB,"SP,",RS      ; 31
3871   156F 3918              DWL C32 ;    "STA",TAB,RS            ; 32
3872   1571 3E18              DWL C33 ;    "INX",TAB,"SP",RS       ; 33
3873   1573 4518              DWL C34 ;    "INR",TAB,"M",RS        ; 34
3874   1575 4B18              DWL C35 ;    "DCR",TAB,"M",RS        ; 35
3875   1577 5118              DWL C36 ;    "MVI",TAB,"M,",RS       ; 36
3876   1579 5818              DWL C37 ;    "STC",TAB,RS            ; 37
```

```
3877   157B 5D18            DWL C38 ;    "DFB",TAB,RS             ; 38
3878   157D 6218            DWL C39 ;    "DAD",TAB,"SP",RS        ; 39
3879   157F 6918            DWL C3A ;    "LDA",TAB,RS             ; 3A
3880   1581 6E18            DWL C3B ;    "DCX",TAB,"SP",RS        ; 3B
3881   1583 7518            DWL C3C ;    "INR",TAB,"A",RS         ; 3C
3882   1585 7B18            DWL C3D ;    "DCR",TAB,"A",RS         ; 3D
3883   1587 8118            DWL C3E ;    "MVI",TAB,"A,",RS        ; 3E
3884   1589 8818            DWL C3F ;    "CMC",TAB,RS             ; 3F
3885   158B 8D18            DWL C40 ;    "MOV",TAB,"B,B",RS       ; 40
3886   158D 9518            DWL C41 ;    "MOV",TAB,"B,C",RS       ; 41
3887   158F 9D18            DWL C42 ;    "MOV",TAB,"B,D",RS       ; 42
3888   1591 A518            DWL C43 ;    "MOV",TAB,"B,E",RS       ; 43
3889   1593 AD18            DWL C44 ;    "MOV",TAB,"B,H",RS       ; 44
3890   1595 B518            DWL C45 ;    "MOV",TAB,"B,L",RS       ; 45
3891   1597 BD18            DWL C46 ;    "MOV",TAB,"B,M",RS       ; 46
3892   1599 C518            DWL C47 ;    "MOV",TAB,"B,A",RS       ; 47
3893   159B CD18            DWL C48 ;    "MOV",TAB,"C,B",RS       ; 48
3894   159D D518            DWL C49 ;    "MOV",TAB,"C,C",RS       ; 49
3895   159F DD18            DWL C4A ;    "MOV",TAB,"C,D",RS       ; 4A
3896   15A1 E518            DWL C4B ;    "MOV",TAB,"C,E",RS       ; 4B
3897   15A3 ED18            DWL C4C ;    "MOV",TAB,"C,H",RS       ; 4C
3898   15A5 F518            DWL C4D ;    "MOV",TAB,"C,L",RS       ; 4D
3899   15A7 FD18            DWL C4E ;    "MOV",TAB,"C,M",RS       ; 4E
3900   15A9 0519            DWL C4F ;    "MOV",TAB,"C,A",RS       ; 4F
3901   15AB 0D19            DWL C50 ;    "MOV",TAB,"D,B",RS       ; 50
3902   15AD 1519            DWL C51 ;    "MOV",TAB,"D,C",RS       ; 51
3903   15AF 1D19            DWL C52 ;    "MOV",TAB,"D,D",RS       ; 52
3904   15B1 2519            DWL C53 ;    "MOV",TAB,"D,E",RS       ; 53
3905   15B3 2D19            DWL C54 ;    "MOV",TAB,"D,H",RS       ; 54
3906   15B5 3519            DWL C55 ;    "MOV",TAB,"D,L",RS       ; 55
3907   15B7 3D19            DWL C56 ;    "MOV",TAB,"D,M",RS       ; 56
3908   15B9 4519            DWL C57 ;    "MOV",TAB,"D,A",RS       ; 57
3909   15BB 4D19            DWL C58 ;    "MOV",TAB,"E,B",RS       ; 58
3910   15BD 5519            DWL C59 ;    "MOV",TAB,"E,C",RS       ; 59
3911   15BF 5D19            DWL C5A ;    "MOV",TAB,"E,D",RS       ; 5A
3912   15C1 6519            DWL C5B ;    "MOV",TAB,"E,E",RS       ; 5B
3913   15C3 6D19            DWL C5C ;    "MOV",TAB,"E,H",RS       ; 5C
3914   15C5 7519            DWL C5D ;    "MOV",TAB,"E,L",RS       ; 5D
3915   15C7 7D19            DWL C5E ;    "MOV",TAB,"E,M",RS       ; 5E
3916   15C9 8519            DWL C5F ;    "MOV",TAB,"E,A",RS       ; 5F
3917   15CB 8D19            DWL C60 ;    "MOV",TAB,"H,B",RS       ; 60
3918   15CD 9519            DWL C61 ;    "MOV",TAB,"H,C",RS       ; 61
3919   15CF 9D19            DWL C62 ;    "MOV",TAB,"H,D",RS       ; 62
3920   15D1 A519            DWL C63 ;    "MOV",TAB,"H,E",RS       ; 63
3921   15D3 AD19            DWL C64 ;    "MOV",TAB,"H,H",RS       ; 64
3922   15D5 B519            DWL C65 ;    "MOV",TAB,"H,L",RS       ; 65
3923   15D7 BD19            DWL C66 ;    "MOV",TAB,"H,M",RS       ; 66
3924   15D9 C519            DWL C67 ;    "MOV",TAB,"H,A",RS       ; 67
3925   15DB CD19            DWL C68 ;    "MOV",TAB,"L,B",RS       ; 68
3926   15DD D519            DWL C69 ;    "MOV",TAB,"L,C",RS       ; 69
3927   15DF DD19            DWL C6A ;    "MOV",TAB,"L,D",RS       ; 6A
3928   15E1 E519            DWL C6B ;    "MOV",TAB,"L,E",RS       ; 6B
3929   15E3 ED19            DWL C6C ;    "MOV",TAB,"L,H",RS       ; 6C
3930   15E5 F519            DWL C6D ;    "MOV",TAB,"L,L",RS       ; 6D
3931   15E7 FD19            DWL C6E ;    "MOV",TAB,"L,M",RS       ; 6E
3932   15E9 051A            DWL C6F ;    "MOV",TAB,"L,A",RS       ; 6F
3933   15EB 0D1A            DWL C70 ;    "MOV",TAB,"M,B",RS       ; 70
3934   15ED 151A            DWL C71 ;    "MOV",TAB,"M,C",RS       ; 71
3935   15EF 1D1A            DWL C72 ;    "MOV",TAB,"M,D",RS       ; 72
3936   15F1 251A            DWL C73 ;    "MOV",TAB,"M,E",RS       ; 73
3937   15F3 2D1A            DWL C74 ;    "MOV",TAB,"M,H",RS       ; 74
3938   15F5 351A            DWL C75 ;    "MOV",TAB,"M,L",RS       ; 75
3939   15F7 3D1A            DWL C76 ;    "HLT",TAB,RS             ; 76
3940   15F9 421A            DWL C77 ;    "MOV",TAB,"M,A",RS       ; 77
3941   15FB 4A1A            DWL C78 ;    "MOV",TAB,"A,B",RS       ; 78
3942   15FD 521A            DWL C79 ;    "MOV",TAB,"A,C",RS       ; 79
3943   15FF 5A1A            DWL C7A ;    "MOV",TAB,"A,D",RS       ; 7A
3944   1601 621A            DWL C7B ;    "MOV",TAB,"A,E",RS       ; 7B
3945   1603 6A1A            DWL C7C ;    "MOV",TAB,"A,H",RS       ; 7C
3946   1605 721A            DWL C7D ;    "MOV",TAB,"A,L",RS       ; 7D
3947   1607 7A1A            DWL C7E ;    "MOV",TAB,"A,M",RS       ; 7E
3948   1609 821A            DWL C7F ;    "MOV",TAB,"A,A",RS       ; 7F
3949   160B 8A1A            DWL C80 ;    "ADD",TAB,"B",RS         ; 80
3950   160D 901A            DWL C81 ;    "ADD",TAB,"C",RS         ; 81
3951   160F 961A            DWL C82 ;    "ADD",TAB,"D",RS         ; 82
3952   1611 9C1A            DWL C83 ;    "ADD",TAB,"E",RS         ; 83
```

```
3953   1613 A21A              DWL C84 ;    "ADD",TAB,"H",RS        ; 84
3954   1615 A81A              DWL C85 ;    "ADD",TAB,"L",RS        ; 85
3955   1617 AE1A              DWL C86 ;    "ADD",TAB,"M",RS        ; 86
3956   1619 B41A              DWL C87 ;    "ADD",TAB,"A",RS        ; 87
3957   161B BA1A              DWL C88 ;    "ADC",TAB,"B",RS        ; 88
3958   161D C01A              DWL C89 ;    "ADC",TAB,"C",RS        ; 89
3959   161F C61A              DWL C8A ;    "ADC",TAB,"D",RS        ; 8A
3960   1621 CC1A              DWL C8B ;    "ADC",TAB,"E",RS        ; 8B
3961   1623 D21A              DWL C8C ;    "ADC",TAB,"H",RS        ; 8C
3962   1625 D81A              DWL C8D ;    "ADC",TAB,"L",RS        ; 8D
3963   1627 DE1A              DWL C8E ;    "ADC",TAB,"M",RS        ; 8E
3964   1629 E41A              DWL C8F ;    "ADC",TAB,"A",RS        ; 8F
3965   162B EA1A              DWL C90 ;    "SUB",TAB,"B",RS        ; 90
3966   162D F01A              DWL C91 ;    "SUB",TAB,"C",RS        ; 91
3967   162F F61A              DWL C92 ;    "SUB",TAB,"D",RS        ; 92
3968   1631 FC1A              DWL C93 ;    "SUB",TAB,"E",RS        ; 93
3969   1633 021B              DWL C94 ;    "SUB",TAB,"H",RS        ; 94
3970   1635 081B              DWL C95 ;    "SUB",TAB,"L",RS        ; 95
3971   1637 0E1B              DWL C96 ;    "SUB",TAB,"M",RS        ; 96
3972   1639 141B              DWL C97 ;    "SUB",TAB,"A",RS        ; 97
3973   163B 1A1B              DWL C98 ;    "SBB",TAB,"B",RS        ; 98
3974   163D 201B              DWL C99 ;    "SBB",TAB,"C",RS        ; 99
3975   163F 261B              DWL C9A ;    "SBB",TAB,"D",RS        ; 9A
3976   1641 2C1B              DWL C9B ;    "SBB",TAB,"E",RS        ; 9B
3977   1643 321B              DWL C9C ;    "SBB",TAB,"H",RS        ; 9C
3978   1645 381B              DWL C9D ;    "SBB",TAB,"L",RS        ; 9D
3979   1647 3E1B              DWL C9E ;    "SBB",TAB,"M",RS        ; 9E
3980   1649 441B              DWL C9F ;    "SBB",TAB,"A",RS        ; 9F
3981   164B 4A1B              DWL CA0 ;    "ANA",TAB,"B",RS        ; A0
3982   164D 501B              DWL CA1 ;    "ANA",TAB,"C",RS        ; A1
3983   164F 561B              DWL CA2 ;    "ANA",TAB,"D",RS        ; A2
3984   1651 5C1B              DWL CA3 ;    "ANA",TAB,"E",RS        ; A3
3985   1653 621B              DWL CA4 ;    "ANA",TAB,"H",RS        ; A4
3986   1655 681B              DWL CA5 ;    "ANA",TAB,"L",RS        ; A5
3987   1657 6E1B              DWL CA6 ;    "ANA",TAB,"M",RS        ; A6
3988   1659 741B              DWL CA7 ;    "ANA",TAB,"A",RS        ; A7
3989   165B 7A1B              DWL CA8 ;    "XRA",TAB,"B",RS        ; A8
3990   165D 801B              DWL CA9 ;    "XRA",TAB,"C",RS        ; A9
3991   165F 861B              DWL CAA ;    "XRA",TAB,"D",RS        ; AA
3992   1661 8C1B              DWL CAB ;    "XRA",TAB,"E",RS        ; AB
3993   1663 921B              DWL CAC ;    "XRA",TAB,"H",RS        ; AC
3994   1665 981B              DWL CAD ;    "XRA",TAB,"L",RS        ; AD
3995   1667 9E1B              DWL CAE ;    "XRA",TAB,"M",RS        ; AE
3996   1669 A41B              DWL CAF ;    "XRA",TAB,"A",RS        ; AF
3997   166B AA1B              DWL CB0 ;    "ORA",TAB,"B",RS        ; B0
3998   166D B01B              DWL CB1 ;    "ORA",TAB,"C",RS        ; B1
3999   166F B61B              DWL CB2 ;    "ORA",TAB,"D",RS        ; B2
4000   1671 BC1B              DWL CB3 ;    "ORA",TAB,"E",RS        ; B3
4001   1673 C21B              DWL CB4 ;    "ORA",TAB,"H",RS        ; B4
4002   1675 C81B              DWL CB5 ;    "ORA",TAB,"L",RS        ; B5
4003   1677 CE1B              DWL CB6 ;    "ORA",TAB,"M",RS        ; B6
4004   1679 D41B              DWL CB7 ;    "ORA",TAB,"A",RS        ; B7
4005   167B DA1B              DWL CB8 ;    "CMP",TAB,"B",RS        ; B8
4006   167D E01B              DWL CB9 ;    "CMP",TAB,"C",RS        ; B9
4007   167F E61B              DWL CBA ;    "CMP",TAB,"D",RS        ; BA
4008   1681 EC1B              DWL CBB ;    "CMP",TAB,"E",RS        ; BB
4009   1683 F21B              DWL CBC ;    "CMP",TAB,"H",RS        ; BC
4010   1685 F81B              DWL CBD ;    "CMP",TAB,"L",RS        ; BD
4011   1687 FE1B              DWL CBE ;    "CMP",TAB,"M",RS        ; BE
4012   1689 041C              DWL CBF ;    "CMP",TAB,"A",RS        ; BF
4013   168B 0A1C              DWL CC0 ;    "RNZ",TAB,RS            ; C0
4014   168D 0F1C              DWL CC1 ;    "POP",TAB,"B",RS        ; C1
4015   168F 151C              DWL CC2 ;    "JNZ",TAB,RS            ; C2
4016   1691 1A1C              DWL CC3 ;    "JMP",TAB,RS            ; C3
4017   1693 1F1C              DWL CC4 ;    "CNZ",TAB,RS            ; C4
4018   1695 241C              DWL CC5 ;    "PUSH",TAB,"B",RS       ; C5
4019   1697 2B1C              DWL CC6 ;    "ADI",TAB,RS            ; C6
4020   1699 301C              DWL CC7 ;    "RST",TAB,"0",RS        ; C7
4021   169B 361C              DWL CC8 ;    "RZ",TAB,RS             ; C8
4022   169D 3A1C              DWL CC9 ;    "RET",TAB,RS            ; C9
4023   169F 3F1C              DWL CCA ;    "JZ",TAB,RS             ; CA
4024   16A1 431C              DWL CCB ;    "DFB",TAB,RS             ; CB
4025   16A3 481C              DWL CCC ;    "CZ",TAB,RS             ; CC
4026   16A5 4C1C              DWL CCD ;    "CALL",TAB,RS           ; CD
4027   16A7 521C              DWL CCE ;    "ACI",TAB,RS            ; CE
4028   16A9 571C              DWL CCF ;    "RST",TAB,"1",RS        ; CF
```

```
4029   16AB 5D1C             DWL CD0 ;    "RNC",TAB,RS            ; D0
4030   16AD 621C             DWL CD1 ;    "POP",TAB,"D",RS        ; D1
4031   16AF 681C             DWL CD2 ;    "JNC",TAB,RS            ; D2
4032   16B1 6D1C             DWL CD3 ;    "OUT",TAB,RS            ; D3
4033   16B3 721C             DWL CD4 ;    "CNC",TAB,RS            ; D4
4034   16B5 771C             DWL CD5 ;    "PUSH",TAB,"D",RS       ; D5
4035   16B7 7E1C             DWL CD6 ;    "SUI",TAB,RS            ; D6
4036   16B9 831C             DWL CD7 ;    "RST",TAB,"2",RS        ; D7
4037   16BB 891C             DWL CD8 ;    "RC",TAB,RS             ; D8
4038   16BD 8D1C             DWL CD9 ;    "DFB",TAB,RS             ; D9
4039   16BF 921C             DWL CDA ;    "JC",TAB,RS             ; DA
4040   16C1 961C             DWL CDB ;    "IN",TAB,RS             ; DFB
4041   16C3 9A1C             DWL CDC ;    "CC",TAB,RS             ; DC
4042   16C5 9E1C             DWL CDD ;    "DFB",TAB,RS             ; DD
4043   16C7 A31C             DWL CDE ;    "SBI",TAB,RS            ; DE
4044   16C9 A81C             DWL CDF ;    "RST",TAB,"3",RS        ; DF
4045   16CB AE1C             DWL CE0 ;    "RPO",TAB,RS            ; E0
4046   16CD B31C             DWL CE1 ;    "POP",TAB,"H",RS        ; E1
4047   16CF B91C             DWL CE2 ;    "JPO",TAB,RS            ; E2
4048   16D1 BE1C             DWL CE3 ;    "XTHL",TAB,RS           ; E3
4049   16D3 C41C             DWL CE4 ;    "CPO",TAB,RS            ; E4
4050   16D5 C91C             DWL CE5 ;    "PUSH",TAB,"H",RS       ; E5
4051   16D7 D01C             DWL CE6 ;    "ANI",TAB,RS            ; E6
4052   16D9 D51C             DWL CE7 ;    "RST",TAB,"4",RS        ; E7
4053   16DB DB1C             DWL CE8 ;    "RPE",TAB,RS            ; E8
4054   16DD E01C             DWL CE9 ;    "PCHL",TAB,RS           ; E9
4055   16DF E61C             DWL CEA ;    "JPE",TAB,RS            ; EA
4056   16E1 EB1C             DWL CEB ;    "XCHG",TAB,RS           ; EB
4057   16E3 F11C             DWL CEC ;    "CPE",TAB,RS            ; EC
4058   16E5 F61C             DWL CED ;    "DFB",TAB,RS             ; ED
4059   16E7 FB1C             DWL CEE ;    "XRI",TAB,RS            ; EE
4060   16E9 001D             DWL CEF ;    "RST",TAB,"5",RS        ; EF
4061   16EB 061D             DWL CF0 ;    "RP",TAB,RS             ; F0
4062   16ED 0A1D             DWL CF1 ;    "POP",TAB,"PSW",RS      ; F1
4063   16EF 121D             DWL CF2 ;    "JP",TAB,RS             ; F2
4064   16F1 161D             DWL CF3 ;    "DI",TAB,RS             ; F3
4065   16F3 1A1D             DWL CF4 ;    "CP",TAB,RS             ; F4
4066   16F5 1E1D             DWL CF5 ;    "PUSH",TAB,"PSW",RS     ; F5
4067   16F7 271D             DWL CF6 ;    "ORI",TAB,RS            ; F6
4068   16F9 2C1D             DWL CF7 ;    "RST",TAB,"6",RS        ; F7
4069   16FB 321D             DWL CF8 ;    "RM",TAB,RS             ; F8
4070   16FD 361D             DWL CF9 ;    "SPHL",TAB,RS           ; F9
4071   16FF 3C1D             DWL CFA ;    "JM",TAB,RS             ; FA
4072   1701 401D             DWL CFB ;    "EI",TAB,RS             ; FB
4073   1703 441D             DWL CFC ;    "CM",TAB,RS             ; FC
4074   1705 481D             DWL CFD ;    "DFB",TAB,RS             ; FD
4075   1707 4D1D             DWL CFE ;    "CPI",TAB,RS            ; FE
4076   1709 521D             DWL CFF ;    "RST",TAB,"7",RS,     ; FF
4077
4078                  ; --------- mnemonic table -------------------------------------
4079
4080
4081   170B           MNEM
4082   170B 4E4F500900C0        DFB      "NOP",TAB,RS            ; 00
4083   1710 4C58490942C1        DFB      "LXI",TAB,"B,",RS       ; 01
4084   1717 5354415809C2        DFB      "STAX",TAB,"B",RS       ; 02
4085   171E 494E580942C3        DFB      "INX",TAB,"B",RS        ; 03
4086   1724 494E520942C4        DFB      "INR",TAB,"B",RS        ; 04
4087   172A 4443520942C5        DFB      "DCR",TAB,"B",RS        ; 05
4088   1730 4D56490942C6        DFB      "MVI",TAB,"B,",RS       ; 06
4089   1737 524C430900C7        DFB      "RLC",TAB,RS            ; 07
4090   173C 4446420900C8        DFB      "DFB",TAB,RS            ; 08
4091   1741 4441440942C9        DFB      "DAD",TAB,"B",RS        ; 09
4092   1747 4C44415809CA        DFB      "LDAX",TAB,"B",RS       ; 0A
4093   174E 4443580942CB        DFB      "DCX",TAB,"B",RS        ; 0B
4094   1754 494E520943CC        DFB      "INR",TAB,"C",RS        ; 0C
4095   175A 4443520943CD        DFB      "DCR",TAB,"C",RS        ; 0D
4096   1760 4D56490943CE        DFB      "MVI",TAB,"C,",RS       ; 0E
4097   1767 5252430900CF        DFB      "RRC",TAB,RS            ; 0F
4098   176C 4446420900C10        DFB      "DFB",TAB,RS            ; 10
4099   1771 4C58490944C11        DFB      "LXI",TAB,"D,",RS       ; 11
4100   1778 5354415809C12        DFB      "STAX",TAB,"D",RS       ; 12
4101   177F 494E580944C13        DFB      "INX",TAB,"D",RS        ; 13
4102   1785 494E520944C14        DFB      "INR",TAB,"D",RS        ; 14
4103   178B 4443520944C15        DFB      "DCR",TAB,"D",RS        ; 15
4104   1791 4D56490944C16        DFB      "MVI",TAB,"D,",RS       ; 16
```

```
4105   1798 52414C0900C17        DFB      "RAL",TAB,RS            ; 17
4106   179D 4446420900C18        DFB      "DFB",TAB,RS             ; 18
4107   17A2 4441440944C19        DFB      "DAD",TAB,"D",RS        ; 19
4108   17A8 4C44415809C1A        DFB      "LDAX",TAB,"D",RS       ; 1A
4109   17AF 4443580944C1B        DFB      "DCX",TAB,"D",RS        ; 1B
4110   17B5 494E520945C1C        DFB      "INR",TAB,"E",RS        ; 1C
4111   17BB 4443520945C1D        DFB      "DCR",TAB,"E",RS        ; 1D
4112   17C1 4D56490945C1E        DFB      "MVI",TAB,"E,",RS       ; 1E
4113   17C8 5241520900C1F        DFB      "RAR",TAB,RS            ; 1F
4114   17CD 52494D0900C20        DFB      "RIM",TAB,RS            ; 20
4115   17D2 4C58490948C21        DFB      "LXI",TAB,"H,",RS       ; 21
4116   17D9 53484C4409C22        DFB      "SHLD",TAB,RS           ; 22
4117   17DF 494E580948C23        DFB      "INX",TAB,"H",RS        ; 23
4118   17E5 494E520948C24        DFB      "INR",TAB,"H",RS        ; 24
4119   17EB 4443520948C25        DFB      "DCR",TAB,"H",RS        ; 25
4120   17F1 4D56490948C26        DFB      "MVI",TAB,"H,",RS       ; 26
4121   17F8 4441410900C27        DFB      "DAA",TAB,RS            ; 27
4122   17FD 4446420900C28        DFB      "DFB",TAB,RS             ; 28
4123   1802 4441440948C29        DFB      "DAD",TAB,"H",RS        ; 29
4124   1808 4C484C4409C2A        DFB      "LHLD",TAB,RS           ; 2A
4125   180E 4443580948C2B        DFB      "DCX",TAB,"H",RS        ; 2B
4126   1814 494E52094CC2C        DFB      "INR",TAB,"L",RS        ; 2C
4127   181A 444352094CC2D        DFB      "DCR",TAB,"L",RS        ; 2D
4128   1820 4D5649094CC2E        DFB      "MVI",TAB,"L,",RS       ; 2E
4129   1827 434D410900C2F        DFB      "CMA",TAB,RS            ; 2F
4130   182C 53494D0900C30        DFB      "SIM",TAB,RS            ; 30
4131   1831 4C58490953C31        DFB      "LXI",TAB,"SP,",RS      ; 31
4132   1839 5354410900C32        DFB      "STA",TAB,RS            ; 32
4133   183E 494E580953C33        DFB      "INX",TAB,"SP",RS       ; 33
4134   1845 494E52094DC34        DFB      "INR",TAB,"M",RS        ; 34
4135   184B 444352094DC35        DFB      "DCR",TAB,"M",RS        ; 35
4136   1851 4D5649094DC36        DFB      "MVI",TAB,"M,",RS       ; 36
4137   1858 5354430900C37        DFB      "STC",TAB,RS            ; 37
4138   185D 4446420900C38        DFB      "DFB",TAB,RS             ; 38
4139   1862 4441440953C39        DFB      "DAD",TAB,"SP",RS       ; 39
4140   1869 4C44410900C3A        DFB      "LDA",TAB,RS            ; 3A
4141   186E 4443580953C3B        DFB      "DCX",TAB,"SP",RS       ; 3B
4142   1875 494E520941C3C        DFB      "INR",TAB,"A",RS        ; 3C
4143   187B 4443520941C3D        DFB      "DCR",TAB,"A",RS        ; 3D
4144   1881 4D56490941C3E        DFB      "MVI",TAB,"A,",RS       ; 3E
4145   1888 434D430900C3F        DFB      "CMC",TAB,RS            ; 3F
4146   188D 4D4F560942C40        DFB      "MOV",TAB,"B,B",RS      ; 40
4147   1895 4D4F560942C41        DFB      "MOV",TAB,"B,C",RS      ; 41
4148   189D 4D4F560942C42        DFB      "MOV",TAB,"B,D",RS      ; 42
4149   18A5 4D4F560942C43        DFB      "MOV",TAB,"B,E",RS      ; 43
4150   18AD 4D4F560942C44        DFB      "MOV",TAB,"B,H",RS      ; 44
4151   18B5 4D4F560942C45        DFB      "MOV",TAB,"B,L",RS      ; 45
4152   18BD 4D4F560942C46        DFB      "MOV",TAB,"B,M",RS      ; 46
4153   18C5 4D4F560942C47        DFB      "MOV",TAB,"B,A",RS      ; 47
4154   18CD 4D4F560943C48        DFB      "MOV",TAB,"C,B",RS      ; 48
4155   18D5 4D4F560943C49        DFB      "MOV",TAB,"C,C",RS      ; 49
4156   18DD 4D4F560943C4A        DFB      "MOV",TAB,"C,D",RS      ; 4A
4157   18E5 4D4F560943C4B        DFB      "MOV",TAB,"C,E",RS      ; 4B
4158   18ED 4D4F560943C4C        DFB      "MOV",TAB,"C,H",RS      ; 4C
4159   18F5 4D4F560943C4D        DFB      "MOV",TAB,"C,L",RS      ; 4D
4160   18FD 4D4F560943C4E        DFB      "MOV",TAB,"C,M",RS      ; 4E
4161   1905 4D4F560943C4F        DFB      "MOV",TAB,"C,A",RS      ; 4F
4162   190D 4D4F560944C50        DFB      "MOV",TAB,"D,B",RS      ; 50
4163   1915 4D4F560944C51        DFB      "MOV",TAB,"D,C",RS      ; 51
4164   191D 4D4F560944C52        DFB      "MOV",TAB,"D,D",RS      ; 52
4165   1925 4D4F560944C53        DFB      "MOV",TAB,"D,E",RS      ; 53
4166   192D 4D4F560944C54        DFB      "MOV",TAB,"D,H",RS      ; 54
4167   1935 4D4F560944C55        DFB      "MOV",TAB,"D,L",RS      ; 55
4168   193D 4D4F560944C56        DFB      "MOV",TAB,"D,M",RS      ; 56
4169   1945 4D4F560944C57        DFB      "MOV",TAB,"D,A",RS      ; 57
4170   194D 4D4F560945C58        DFB      "MOV",TAB,"E,B",RS      ; 58
4171   1955 4D4F560945C59        DFB      "MOV",TAB,"E,C",RS      ; 59
4172   195D 4D4F560945C5A        DFB      "MOV",TAB,"E,D",RS      ; 5A
4173   1965 4D4F560945C5B        DFB      "MOV",TAB,"E,E",RS      ; 5B
4174   196D 4D4F560945C5C        DFB      "MOV",TAB,"E,H",RS      ; 5C
4175   1975 4D4F560945C5D        DFB      "MOV",TAB,"E,L",RS      ; 5D
4176   197D 4D4F560945C5E        DFB      "MOV",TAB,"E,M",RS      ; 5E
4177   1985 4D4F560945C5F        DFB      "MOV",TAB,"E,A",RS      ; 5F
4178   198D 4D4F560948C60        DFB      "MOV",TAB,"H,B",RS      ; 60
4179   1995 4D4F560948C61        DFB      "MOV",TAB,"H,C",RS      ; 61
4180   199D 4D4F560948C62        DFB      "MOV",TAB,"H,D",RS      ; 62
```

```
4181   19A5 4D4F560948C63        DFB      "MOV",TAB,"H,E",RS      ; 63
4182   19AD 4D4F560948C64        DFB      "MOV",TAB,"H,H",RS      ; 64
4183   19B5 4D4F560948C65        DFB      "MOV",TAB,"H,L",RS      ; 65
4184   19BD 4D4F560948C66        DFB      "MOV",TAB,"H,M",RS      ; 66
4185   19C5 4D4F560948C67        DFB      "MOV",TAB,"H,A",RS      ; 67
4186   19CD 4D4F56094CC68        DFB      "MOV",TAB,"L,B",RS      ; 68
4187   19D5 4D4F56094CC69        DFB      "MOV",TAB,"L,C",RS      ; 69
4188   19DD 4D4F56094CC6A        DFB      "MOV",TAB,"L,D",RS      ; 6A
4189   19E5 4D4F56094CC6B        DFB      "MOV",TAB,"L,E",RS      ; 6B
4190   19ED 4D4F56094CC6C        DFB      "MOV",TAB,"L,H",RS      ; 6C
4191   19F5 4D4F56094CC6D        DFB      "MOV",TAB,"L,L",RS      ; 6D
4192   19FD 4D4F56094CC6E        DFB      "MOV",TAB,"L,M",RS      ; 6E
4193   1A05 4D4F56094CC6F        DFB      "MOV",TAB,"L,A",RS      ; 6F
4194   1A0D 4D4F56094DC70        DFB      "MOV",TAB,"M,B",RS      ; 70
4195   1A15 4D4F56094DC71        DFB      "MOV",TAB,"M,C",RS      ; 71
4196   1A1D 4D4F56094DC72        DFB      "MOV",TAB,"M,D",RS      ; 72
4197   1A25 4D4F56094DC73        DFB      "MOV",TAB,"M,E",RS      ; 73
4198   1A2D 4D4F56094DC74        DFB      "MOV",TAB,"M,H",RS      ; 74
4199   1A35 4D4F56094DC75        DFB      "MOV",TAB,"M,L",RS      ; 75
4200   1A3D 484C540900C76        DFB      "HLT",TAB,RS            ; 76
4201   1A42 4D4F56094DC77        DFB      "MOV",TAB,"M,A",RS      ; 77
4202   1A4A 4D4F560941C78        DFB      "MOV",TAB,"A,B",RS      ; 78
4203   1A52 4D4F560941C79        DFB      "MOV",TAB,"A,C",RS      ; 79
4204   1A5A 4D4F560941C7A        DFB      "MOV",TAB,"A,D",RS      ; 7A
4205   1A62 4D4F560941C7B        DFB      "MOV",TAB,"A,E",RS      ; 7B
4206   1A6A 4D4F560941C7C        DFB      "MOV",TAB,"A,H",RS      ; 7C
4207   1A72 4D4F560941C7D        DFB      "MOV",TAB,"A,L",RS      ; 7D
4208   1A7A 4D4F560941C7E        DFB      "MOV",TAB,"A,M",RS      ; 7E
4209   1A82 4D4F560941C7F        DFB      "MOV",TAB,"A,A",RS      ; 7F
4210   1A8A 4144440942C80        DFB      "ADD",TAB,"B",RS        ; 80
4211   1A90 4144440943C81        DFB      "ADD",TAB,"C",RS        ; 81
4212   1A96 4144440944C82        DFB      "ADD",TAB,"D",RS        ; 82
4213   1A9C 4144440945C83        DFB      "ADD",TAB,"E",RS        ; 83
4214   1AA2 4144440948C84        DFB      "ADD",TAB,"H",RS        ; 84
4215   1AA8 414444094CC85        DFB      "ADD",TAB,"L",RS        ; 85
4216   1AAE 414444094DC86        DFB      "ADD",TAB,"M",RS        ; 86
4217   1AB4 4144440941C87        DFB      "ADD",TAB,"A",RS        ; 87
4218   1ABA 4144430942C88        DFB      "ADC",TAB,"B",RS        ; 88
4219   1AC0 4144430943C89        DFB      "ADC",TAB,"C",RS        ; 89
4220   1AC6 4144430944C8A        DFB      "ADC",TAB,"D",RS        ; 8A
4221   1ACC 4144430945C8B        DFB      "ADC",TAB,"E",RS        ; 8B
4222   1AD2 4144430948C8C        DFB      "ADC",TAB,"H",RS        ; 8C
4223   1AD8 414443094CC8D        DFB      "ADC",TAB,"L",RS        ; 8D
4224   1ADE 414443094DC8E        DFB      "ADC",TAB,"M",RS        ; 8E
4225   1AE4 4144430941C8F        DFB      "ADC",TAB,"A",RS        ; 8F
4226   1AEA 5355420942C90        DFB      "SUB",TAB,"B",RS        ; 90
4227   1AF0 5355420943C91        DFB      "SUB",TAB,"C",RS        ; 91
4228   1AF6 5355420944C92        DFB      "SUB",TAB,"D",RS        ; 92
4229   1AFC 5355420945C93        DFB      "SUB",TAB,"E",RS        ; 93
4230   1B02 5355420948C94        DFB      "SUB",TAB,"H",RS        ; 94
4231   1B08 535542094CC95        DFB      "SUB",TAB,"L",RS        ; 95
4232   1B0E 535542094DC96        DFB      "SUB",TAB,"M",RS        ; 96
4233   1B14 5355420941C97        DFB      "SUB",TAB,"A",RS        ; 97
4234   1B1A 5342420942C98        DFB      "SBB",TAB,"B",RS        ; 98
4235   1B20 5342420943C99        DFB      "SBB",TAB,"C",RS        ; 99
4236   1B26 5342420944C9A        DFB      "SBB",TAB,"D",RS        ; 9A
4237   1B2C 5342420945C9B        DFB      "SBB",TAB,"E",RS        ; 9B
4238   1B32 5342420948C9C        DFB      "SBB",TAB,"H",RS        ; 9C
4239   1B38 534242094CC9D        DFB      "SBB",TAB,"L",RS        ; 9D
4240   1B3E 534242094DC9E        DFB      "SBB",TAB,"M",RS        ; 9E
4241   1B44 5342420941C9F        DFB      "SBB",TAB,"A",RS        ; 9F
4242   1B4A 414E410942CA0        DFB      "ANA",TAB,"B",RS        ; A0
4243   1B50 414E410943CA1        DFB      "ANA",TAB,"C",RS        ; A1
4244   1B56 414E410944CA2        DFB      "ANA",TAB,"D",RS        ; A2
4245   1B5C 414E410945CA3        DFB      "ANA",TAB,"E",RS        ; A3
4246   1B62 414E410948CA4        DFB      "ANA",TAB,"H",RS        ; A4
4247   1B68 414E41094CCA5        DFB      "ANA",TAB,"L",RS        ; A5
4248   1B6E 414E41094DCA6        DFB      "ANA",TAB,"M",RS        ; A6
4249   1B74 414E410941CA7        DFB      "ANA",TAB,"A",RS        ; A7
4250   1B7A 5852410942CA8        DFB      "XRA",TAB,"B",RS        ; A8
4251   1B80 5852410943CA9        DFB      "XRA",TAB,"C",RS        ; A9
4252   1B86 5852410944CAA        DFB      "XRA",TAB,"D",RS        ; AA
4253   1B8C 5852410945CAB        DFB      "XRA",TAB,"E",RS        ; AB
4254   1B92 5852410948CAC        DFB      "XRA",TAB,"H",RS        ; AC
4255   1B98 585241094CCAD        DFB      "XRA",TAB,"L",RS        ; AD
4256   1B9E 585241094DCAE        DFB      "XRA",TAB,"M",RS        ; AE
```

```
4257   1BA4 5852410941CAF        DFB      "XRA",TAB,"A",RS          ; AF
4258   1BAA 4F52410942CB0        DFB      "ORA",TAB,"B",RS          ; B0
4259   1BB0 4F52410943CB1        DFB      "ORA",TAB,"C",RS          ; B1
4260   1BB6 4F52410944CB2        DFB      "ORA",TAB,"D",RS          ; B2
4261   1BBC 4F52410945CB3        DFB      "ORA",TAB,"E",RS          ; B3
4262   1BC2 4F52410948CB4        DFB      "ORA",TAB,"H",RS          ; B4
4263   1BC8 4F5241094CCB5        DFB      "ORA",TAB,"L",RS          ; B5
4264   1BCE 4F5241094DCB6        DFB      "ORA",TAB,"M",RS          ; B6
4265   1BD4 4F52410941CB7        DFB      "ORA",TAB,"A",RS          ; B7
4266   1BDA 434D500942CB8        DFB      "CMP",TAB,"B",RS          ; B8
4267   1BE0 434D500943CB9        DFB      "CMP",TAB,"C",RS          ; B9
4268   1BE6 434D500944CBA        DFB      "CMP",TAB,"D",RS          ; BA
4269   1BEC 434D500945CBB        DFB      "CMP",TAB,"E",RS          ; BB
4270   1BF2 434D500948CBC        DFB      "CMP",TAB,"H",RS          ; BC
4271   1BF8 434D50094CCBD        DFB      "CMP",TAB,"L",RS          ; BD
4272   1BFE 434D50094DCBE        DFB      "CMP",TAB,"M",RS          ; BE
4273   1C04 434D500941CBF        DFB      "CMP",TAB,"A",RS          ; BF
4274   1C0A 524E5A0900CC0        DFB      "RNZ",TAB,RS              ; C0
4275   1C0F 504F500942CC1        DFB      "POP",TAB,"B",RS          ; C1
4276   1C15 4A4E5A0900CC2        DFB      "JNZ",TAB,RS              ; C2
4277   1C1A 4A4D500900CC3        DFB      "JMP",TAB,RS              ; C3
4278   1C1F 434E5A0900CC4        DFB      "CNZ",TAB,RS              ; C4
4279   1C24 5055534809CC5        DFB      "PUSH",TAB,"B",RS         ; C5
4280   1C2B 4144490900CC6        DFB      "ADI",TAB,RS              ; C6
4281   1C30 5253540930CC7        DFB      "RST",TAB,"0",RS          ; C7
4282   1C36 525A0900  CC8        DFB      "RZ",TAB,RS               ; C8
4283   1C3A 5245540900CC9        DFB      "RET",TAB,RS              ; C9
4284   1C3F 4A5A0900  CCA        DFB      "JZ",TAB,RS               ; CA
4285   1C43 4446420900CCB        DFB      "DFB",TAB,RS               ; CB
4286   1C48 435A0900  CCC        DFB      "CZ",TAB,RS               ; CC
4287   1C4C 43414C4C09CCD        DFB      "CALL",TAB,RS             ; CD
4288   1C52 4143490900CCE        DFB      "ACI",TAB,RS              ; CE
4289   1C57 5253540931CCF        DFB      "RST",TAB,"1",RS          ; CF
4290   1C5D 524E430900CD0        DFB      "RNC",TAB,RS              ; D0
4291   1C62 504F500944CD1        DFB      "POP",TAB,"D",RS          ; D1
4292   1C68 4A4E430900CD2        DFB      "JNC",TAB,RS              ; D2
4293   1C6D 4F55540900CD3        DFB      "OUT",TAB,RS              ; D3
4294   1C72 434E430900CD4        DFB      "CNC",TAB,RS              ; D4
4295   1C77 5055534809CD5        DFB      "PUSH",TAB,"D",RS         ; D5
4296   1C7E 5355490900CD6        DFB      "SUI",TAB,RS              ; D6
4297   1C83 5253540932CD7        DFB      "RST",TAB,"2",RS          ; D7
4298   1C89 52430900  CD8        DFB      "RC",TAB,RS               ; D8
4299   1C8D 4446420900CD9        DFB      "DFB",TAB,RS               ; D9
4300   1C92 4A430900  CDA        DFB      "JC",TAB,RS               ; DA
4301   1C96 494E0900  CDB        DFB      "IN",TAB,RS               ; DFB
4302   1C9A 43430900  CDC        DFB      "CC",TAB,RS               ; DC
4303   1C9E 4446420900CDD        DFB      "DFB",TAB,RS               ; DD
4304   1CA3 5342490900CDE        DFB      "SBI",TAB,RS              ; DE
4305   1CA8 5253540933CDF        DFB      "RST",TAB,"3",RS          ; DF
4306   1CAE 52504F0900CE0        DFB      "RPO",TAB,RS              ; E0
4307   1CB3 504F500948CE1        DFB      "POP",TAB,"H",RS          ; E1
4308   1CB9 4A504F0900CE2        DFB      "JPO",TAB,RS              ; E2
4309   1CBE 5854484C09CE3        DFB      "XTHL",TAB,RS             ; E3
4310   1CC4 43504F0900CE4        DFB      "CPO",TAB,RS              ; E4
4311   1CC9 5055534809CE5        DFB      "PUSH",TAB,"H",RS         ; E5
4312   1CD0 414E490900CE6        DFB      "ANI",TAB,RS              ; E6
4313   1CD5 5253540934CE7        DFB      "RST",TAB,"4",RS          ; E7
4314   1CDB 5250450900CE8        DFB      "RPE",TAB,RS              ; E8
4315   1CE0 5043484C09CE9        DFB      "PCHL",TAB,RS             ; E9
4316   1CE6 4A50450900CEA        DFB      "JPE",TAB,RS              ; EA
4317   1CEB 5843484709CEB        DFB      "XCHG",TAB,RS             ; EB
4318   1CF1 4350450900CEC        DFB      "CPE",TAB,RS              ; EC
4319   1CF6 4446420900CED        DFB      "DFB",TAB,RS               ; ED
4320   1CFB 5852490900CEE        DFB      "XRI",TAB,RS              ; EE
4321   1D00 5253540935CEF        DFB      "RST",TAB,"5",RS          ; EF
4322   1D06 52500900  CF0        DFB      "RP",TAB,RS               ; F0
4323   1D0A 504F500950CF1        DFB      "POP",TAB,"PSW",RS        ; F1
4324   1D12 4A500900  CF2        DFB      "JP",TAB,RS               ; F2
4325   1D16 44490900  CF3        DFB      "DI",TAB,RS               ; F3
4326   1D1A 43500900  CF4        DFB      "CP",TAB,RS               ; F4
4327   1D1E 5055534809CF5        DFB      "PUSH",TAB,"PSW",RS       ; F5
4328   1D27 4F52490900CF6        DFB      "ORI",TAB,RS              ; F6
4329   1D2C 5253540936CF7        DFB      "RST",TAB,"6",RS          ; F7
4330   1D32 524D0900  CF8        DFB      "RM",TAB,RS               ; F8
4331   1D36 5350484C09CF9        DFB      "SPHL",TAB,RS             ; F9
4332   1D3C 4A4D0900  CFA        DFB      "JM",TAB,RS               ; FA
```

```
1D40 45490900  CFB        DFB      "EI",TAB,RS             ; FB
1D44 434D0900  CFC        DFB      "CM",TAB,RS             ; FC
1D48 4446420900CFD        DFB      "DFB",TAB,RS             ; FD
1D4D 4350490900CFE        DFB      "CPI",TAB,RS            ; FE
1D52 5253540937CFF        DFB      "RST",TAB,"7",RS,     ; FF


1D58 0D0A0A4D54prompt1:    dfb cr,lf,lf,"MTK-85 8085 MICROPROCESSOR TRAINING KIT (? HELP

1D8E 3F065B4F66convert  dfb 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh,77h,7ch,39h,5eh,79h,

1D9E 0000000000off_display: dfb 0,0,0,0,0,0

                ; lcd message     |--- 20 letters ---|
1DA4 4D544B2D38prompt2:     dfb "MTK-85 8085 MICROPRO",0
1DB9 434553534Ftext3:       dfb "CESSOR TRAINING KIT",0


1DCD 6C6F616420download_text: dfb "load Intel hex file...",0
1DE4 3E00      prompt_text:  dfb ">",0
1DE6 6E6577206Cnew_text:     dfb "new location = ",0
1DF6 6564697420edit_text:    dfb "edit memory location = ",0
1E0E 0D0A456E74edit_text1:   dfb cr,lf,"Enter to quit, SPACE key to view content",0
1E39 0D0A0A4144edit_text2:   dfb cr,lf,lf,"ADDR  DATA",0

1E47 6A756D7020jump_text1:   dfb "jump to address [",0
1E59 5D203D2000jump_text2:   dfb "] = ",0
1E5E 7072696E74ascii_text:   dfb "print ASCII code",0

1E6F 426567696Efill_text1:   dfb "Begin address = ",0
1E80 20456E6420fill_text2:   dfb " End address = ",0
1E90 2044617461fill_text3:   dfb " Data = ",0

1E99 206572726Ferror_text:   dfb " errors",0

1EA1 6469736173disassemble_text: dfb "disassemble...",0

0000 =         eos           equ 0

1EB0 41463D00  af_text:  dfb "AF=",eos
1EB4 42433D00  bc_text:  dfb "BC=",eos
1EB8 44453D00  de_text:  dfb "DE=",eos
1EBC 484C3D00  hl_text:  dfb "HL=",eos
1EC0 53503D00  sp_text:  dfb "SP=",eos
1EC4 544F533D00tos_text: dfb "TOS=",eos

1EC9 50433D00  pc_text:  dfb "PC=",eos
1ECD 5B53205A20flag_text: dfb "[S Z - AC  - P - CY]=",eos

1EE3 533D00    sign_text: dfb "S=",0

1EE6 2062797465byte_text: dfb " bytes loaded",0

1EF4 535441434Bstack_text: dfb "STACK Memory Contents..",0

1F0C 7365742076set_register_text: dfb "set value to user register (enter A for AF) ? ",0


1F3B 0D0A0A4D54help_text:  dfb cr,lf,lf,"MTK-85 8085 MICROPROCESSOR TRAINING KIT (? HELP
1F6E 0D0A41202Dhelp_text1: dfb cr,lf,   "A - ASCII code"
1F7E 0D0A43202D            dfb cr,lf,   "C - clear watch variables"
1F99 0D0A44202D            dfb cr,lf,   "D - disassemble"
1FAA 0D0A45202D            dfb cr,lf,   "E - edit memory"
1FBB 0D0A46202D            dfb cr,lf,   "F - fill constant"
1FCE 0D0A48202D            dfb cr,lf,   "H - hex dump"
1FDC 0D0A49202D            dfb cr,lf,   "I - i/o address map"
1FF1 0D0A4A202D            dfb cr,lf,   "J - jump to user program"
200B 0D0A4B202D            dfb cr,lf,   "K - display user STACK"
2023 0D0A4C202D            dfb cr,lf,   "L - load Intel hex file"
203C 0D0A4D202D            dfb cr,lf,   "M - monitor call number"
2055 0D0A4E202D            dfb cr,lf,   "N - new location pointer"
206F 0D0A51202D            dfb cr,lf,   "Q - quick home location"
2088 0D0A52202D            dfb cr,lf,   "R - user register display"
20A3 0D0A53202D            dfb cr,lf,   "S - set value to user register"
20C3 0D0A57202D            dfb cr,lf,   "W - watch variables"
```

```
20D8 0D0A535041           dfb cr,lf,   "SPACE BAR - single step"
20F1 0D0A3F202D           dfb cr,lf,   "? - help menu",cr,lf,0


2103 0D0A303048io_text    dfb cr,lf,"00H-0FH onboard 4-bit GPIO, D0-D3=output port"
2132 0D0A202020           dfb cr,lf,"        D4-D7=input port"
214C 0D0A                 dfb cr,lf
214E 0D0A313048           dfb cr,lf,"10H-13H 8255 system PPI, 10H=PORTA, 11H=PORTB, 12H
2195 0D0A                 dfb cr,lf
2197 0D0A323048           dfb cr,lf,"20H-23H 8254 programmable counter, 20H=counter0, 2
21D7 0D0A202020           dfb cr,lf,"        22H=counter2, 23H control register"
2203 0D0A                 dfb cr,lf
2205 0D0A333048           dfb cr,lf,"30H-33H 8255 user PPI, 30H=PORTA, 31H=PORTB, 32H=P
224A 0D0A                 dfb cr,lf
224C 0D0A343048           dfb cr,lf,"40H-47H C16550 UART registers",0


226C           monitor_text:

226C 7365652069           dfb  "see input parameters in user manual",cr,lf
2291 0D0A31456E           dfb  cr,lf,"1Enn MVI E,function_number"
22AD 0D0A434620           dfb  cr,lf,"CF   RST 1"
22B9 0D0A                 dfb  cr,lf
22BB 0D0A303020           dfb  cr,lf,"00 - demo"  ; #0 running LED with HL pointer
22C6 0D0A303120           dfb  cr,lf,"01 - delay" ; #1 simple delay routine
22D2 0D0A303220           dfb  cr,lf,"02 - cold_boot" ; #2 show 8085 running
22E2 0D0A303320           dfb  cr,lf,"03 - scan"    ; #3 scan display one cycle
22ED 0D0A303420           dfb  cr,lf,"04 - cin"     ; #4 get byte from console
22F7 0D0A303520           dfb  cr,lf,"05 - cout"    ; #5 print byte to console
2302 0D0A303620           dfb  cr,lf,"06 - put_str"  ; #6 print string with 0 terminatc
2310 0D0A303720           dfb  cr,lf,"07 - init_lcd" ; #7 initialize lcd
231F 0D0A303820           dfb  cr,lf,"08 - lcd_ready" ; #8 wait until lcd is ready
232F 0D0A303920           dfb  cr,lf,"09 - clear_lcd" ; #9 clear lcd display
233F 0D0A304120           dfb  cr,lf,"0A - goto_xy"   ; #10 set lcd cursor position
234D 0D0A304220           dfb  cr,lf,"0B - put_str_lcd" ; #11 print ASCII string on lcc
235F 0D0A304320           dfb  cr,lf,"0C - put_ch_lcd"  ; #12 print ASCII letter on lcc
2370 0D0A304420           dfb  cr,lf,"0D - demo2",0  ; #13 run GPIO LED
237D 0D0A304520           dfb  cr,lf,"0E - pint16u",0 ;#14 print 16-bit unsigned


                        ;  data segment

F000                      org system_ram

F000           watch_ram dfs 16       ; watch variable F000-F00F
F010           buffer   dfs 16        ; buffer display
F020           bcs      dfs 1         ; byte checksum
F021           key      dfs 1         ; key position
F022           command  dfs 1         ; serial command
F023           flag1    dfs 1         ; user flag
                                      ; flag1.0 Space key was pressed
                                      ; flag1.1 Enter key was pressed
F024           beep_flag dfs 1        ; beep/no beep

F025           uart_found dfs 1       ; 0 = no uart, 1 uart found

F026           entry_mode dfs 1       ; 0 for data mode
                                      ; 1 for address mode
                                      ; 2 register display
F027           counter1   dfs 1       ; event counter1 for data entry
F028           counter2   dfs 1       ; event counter2 for address entry
F029           warm_code  dfs 1       ; warm boot code

F02A           user_PC  dfs 2         ; user PC
F02C           user_AF  dfs 2         ; user AF
F02E           user_BC  dfs 2         ; user BC
F030           user_DE  dfs 2         ; user DE
F032           user_HL  dfs 2         ; user HL
F034           user_SP  dfs 2         ; user SP
F036           tos      dfs 2         ; Top of STACK
F038           current_register dfs 2 ; stores current displayed register

F03A           temp     dfs 2         ; temporary 16-bit storage

F03C           pointer  dfs 2         ; for hex dump
```

```
F03E            break_address dfs 2    ; break address
F040            break_opcode  dfs 1    ; opcode that saved

F041            bcd_counter1  dfs 3    ; 6-digit BCD counter1
F044            bcd_counter2  dfs 3    ; 6-digit BCD counter2

F047            line_buffer   dfs 16   ; reserved for terminal printing


F057            save_stack    dfs 2    ; for saving system stack
F059            system_stack  dfs 32
F079            user_stack    dfs 32

0000                     END
```

```
4501
4501   1075  AC_TEXT            0A01  ADDRESS_MODE       1EB0  AF_TEXT
4502   1266  ALT_PUT_STR        13C7  ASCII_0_9          0D72  ASCII_PRINT
4503   0D8A  ASCII_PRINT1       1E5E  ASCII_TEXT         F041  BCD_COUNTER1
4504   F044  BCD_COUNTER2       F020  BCS                1EB4  BC_TEXT
4505   096F  BEEP_CHK           F024  BEEP_FLAG          0640  BEEP_ON
4506   019B  BIN1               01A6  BIN2               0199  BIN2ASCII
4507   01B0  BIN3               01BB  BIN4               08CA  BREAK
4508   F03E  BREAK_ADDRESS      F040  BREAK_OPCODE       F010  BUFFER
4509   0080  BUSY               0631  BUZZER             0642  BUZZER1
4510   1EE6  BYTE_TEXT          170B  C0                 1710  C1
4511   176C  C10                1771  C11                1778  C12
4512   177F  C13                1785  C14                178B  C15
4513   1791  C16                1798  C17                179D  C18
4514   17A2  C19                17A8  C1A                17AF  C1B
4515   17B5  C1C                17BB  C1D                17C1  C1E
4516   17C8  C1F                1717  C2                 17CD  C20
4517   17D2  C21                17D9  C22                17DF  C23
4518   17E5  C24                17EB  C25                17F1  C26
4519   17F8  C27                17FD  C28                1802  C29
4520   1808  C2A                180E  C2B                1814  C2C
4521   181A  C2D                1820  C2E                1827  C2F
4522   171E  C3                 182C  C30                1831  C31
4523   1839  C32                183E  C33                1845  C34
4524   184B  C35                1851  C36                1858  C37
4525   185D  C38                1862  C39                1869  C3A
4526   186E  C3B                1875  C3C                187B  C3D
4527   1881  C3E                1888  C3F                1724  C4
4528   188D  C40                1895  C41                189D  C42
4529   18A5  C43                18AD  C44                18B5  C45
4530   18BD  C46                18C5  C47                18CD  C48
4531   18D5  C49                18DD  C4A                18E5  C4B
4532   18ED  C4C                18F5  C4D                18FD  C4E
4533   1905  C4F                172A  C5                 190D  C50
4534   1915  C51                191D  C52                1925  C53
4535   192D  C54                1935  C55                193D  C56
4536   1945  C57                194D  C58                1955  C59
4537   195D  C5A                1965  C5B                196D  C5C
4538   1975  C5D                197D  C5E                1985  C5F
4539   1730  C6                 198D  C60                1995  C61
4540   199D  C62                19A5  C63                19AD  C64
4541   19B5  C65                19BD  C66                19C5  C67
4542   19CD  C68                19D5  C69                19DD  C6A
4543   19E5  C6B                19ED  C6C                19F5  C6D
4544   19FD  C6E                1A05  C6F                1737  C7
4545   1A0D  C70                1A15  C71                1A1D  C72
4546   1A25  C73                1A2D  C74                1A35  C75
4547   1A3D  C76                1A42  C77                1A4A  C78
4548   1A52  C79                1A5A  C7A                1A62  C7B
4549   1A6A  C7C                1A72  C7D                1A7A  C7E
4550   1A82  C7F                173C  C8                 1A8A  C80
4551   1A90  C81                1A96  C82                1A9C  C83
4552   1AA2  C84                1AA8  C85                1AAE  C86
4553   1AB4  C87                1ABA  C88                1AC0  C89
4554   1AC6  C8A                1ACC  C8B                1AD2  C8C
4555   1AD8  C8D                1ADE  C8E                1AE4  C8F
4556   1741  C9                 1AEA  C90                1AF0  C91
4557   1AF6  C92                1AFC  C93                1B02  C94
4558   1B08  C95                1B0E  C96                1B14  C97
4559   1B1A  C98                1B20  C99                1B26  C9A
4560   1B2C  C9B                1B32  C9C                1B38  C9D
4561   1B3E  C9E                1B44  C9F                1747  CA
4562   1B4A  CA0                1B50  CA1                1B56  CA2
4563   1B5C  CA3                1B62  CA4                1B68  CA5
4564   1B6E  CA6                1B74  CA7                1B7A  CA8
4565   1B80  CA9                1B86  CAA                1B8C  CAB
4566   1B92  CAC                1B98  CAD                1B9E  CAE
4567   1BA4  CAF                174E  CB                 1BAA  CB0
4568   1BB0  CB1                1BB6  CB2                1BBC  CB3
4569   1BC2  CB4                1BC8  CB5                1BCE  CB6
4570   1BD4  CB7                1BDA  CB8                1BE0  CB9
4571   1BE6  CBA                1BEC  CBB                1BF2  CBC
4572   1BF8  CBD                1BFE  CBE                1C04  CBF
4573   1754  CC                 1C0A  CC0                1C0F  CC1
4574   1C15  CC2                1C1A  CC3                1C1F  CC4
4575   1C24  CC5                1C2B  CC6                1C30  CC7
```

```
4576  1C36  CC8                1C3A  CC9                1C3F  CCA
4577  1C43  CCB                1C48  CCC                1C4C  CCD
4578  1C52  CCE                1C57  CCF                175A  CD
4579  1C5D  CD0                1C62  CD1                1C68  CD2
4580  1C6D  CD3                1C72  CD4                1C77  CD5
4581  1C7E  CD6                1C83  CD7                1C89  CD8
4582  1C8D  CD9                1C92  CDA                1C96  CDB
4583  1C9A  CDC                1C9E  CDD                1CA3  CDE
4584  1CA8  CDF                1760  CE                 1CAE  CE0
4585  1CB3  CE1                1CB9  CE2                1CBE  CE3
4586  1CC4  CE4                1CC9  CE5                1CD0  CE6
4587  1CD5  CE7                1CDB  CE8                1CE0  CE9
4588  1CE6  CEA                1CEB  CEB                1CF1  CEC
4589  1CF6  CED                1CFB  CEE                1D00  CEF
4590  1767  CF                 1D06  CF0                1D0A  CF1
4591  1D12  CF2                1D16  CF3                1D1A  CF4
4592  1D1E  CF5                1D27  CF6                1D2C  CF7
4593  1D32  CF8                1D36  CF9                1D3C  CFA
4594  1D40  CFB                1D44  CFC                1D48  CFD
4595  1D4D  CFE                1D52  CFF                123B  CIN
4596  0EE3  CLEAR1             14D5  CLEAR_BCD1         14FE  CLEAR_BCD2
4597  0904  CLEAR_BREAK        0200  CLEAR_LCD          0ED6  CLEAR_WATCH
4598  0BB3  CODE1              0BFB  CODE10             0C03  CODE11
4599  0C0B  CODE12             0C13  CODE13             0C1B  CODE14
4600  0C23  CODE15             0C2B  CODE16             0C33  CODE17
4601  0C3B  CODE18             0C43  CODE19             0BBB  CODE2
4602  0C4B  CODE20             0C53  CODE21             0C5B  CODE22
4603  0C63  CODE23             0C6B  CODE24             0C73  CODE25
4604  0C7B  CODE26             0BC3  CODE3              0BCB  CODE4
4605  0BD3  CODE5              0BDB  CODE6              0BE3  CODE7
4606  0BEB  CODE8              0BF3  CODE9              0B7D  COLD1
4607  0B7B  COLD2              0B8C  COLD3              0B76  COLD_BOOT
4608  F022  COMMAND            0052  COMMAND_READ       0050  COMMAND_WRITE
4609  0023  CONTROL_8254       0034  CONTROL_WORD_8254  1D8E  CONVERT
4610  0020  COUNTER0_8254      F027  COUNTER1           0021  COUNTER1_8254
4611  F028  COUNTER2           0022  COUNTER2_8254      122F  COUT
4612  1230  COUT1              000D  CR                 F038  CURRENT_REGISTER
4613  107C  CY_TEXT            0609  DATA_KEY           0614  DATA_KEY1
4614  061C  DATA_KEY2          0624  DATA_KEY3          062C  DATA_KEY4
4615  0A0E  DATA_MODE          0053  DATA_READ          145B  DATA_RECORD
4616  0051  DATA_WRITE         0CA7  DEBOUNCE          0CA9  DEBOUNCE1
4617  0995  DEBUG              0A78  DECREMENT          0600  DELAY
4618  063B  DELAY_NOBEEP       0659  DELAY_US           065B  DELAY_US1
4619  094A  DELETE_BYTE        0959  DELETE_BYTE1       0BA0  DEMO
4620  0BA2  DEMO1_2            1EB8  DE_TEXT            0326  DIS2
4621  0390  DISASSEM1          039F  DISASSEM2          034E  DISASSEM3
4622  033A  DISASSEMBLE        0313  DISASSEMBLE1       1EA1  DISASSEMBLE_TEXT
4623  0660  DISPLAY_OFF        0D50  DOWNLOAD           1DCD  DOWNLOAD_TEXT
4624  1359  DUMP1              1351  DUMP_MEMORY        0406  D_DISASSEM1
4625  0415  D_DISASSEM2        03C4  D_DISASSEM3        03B0  D_DISASSEMBLE
4626  03DC  D_ONE_TAB          114C  EDIT1              1127  EDIT_LOCATION
4627  1DF6  EDIT_TEXT          1E0E  EDIT_TEXT1         1E39  EDIT_TEXT2
4628  0709  ENTER_ADDRESS      06CF  ENTER_DATA         06F4  ENTER_DATA1
4629  066F  ENTER_REGISTER     F026  ENTRY_MODE         0000  EOS
4630  1E99  ERROR_TEXT         001B  ESC                145A  ESC_QUIT
4631  0DA3  EXIT_ASCII_PRINT   0207  EXIT_CLEAR         0EF0  EXIT_CLEAR_WATCH
4632  0339  EXIT_DISASSEMBLE   0D71  EXIT_DOWNLOAD      11A9  EXIT_EDIT
4633  119A  EXIT_EDIT1         10C3  EXIT_FILL          13F1  EXIT_GET_HEX2
4634  13FA  EXIT_GET_HEX3      1205  EXIT_HELP          1350  EXIT_HEX_DUMP
4635  11ED  EXIT_IO            1126  EXIT_JUMP          10DC  EXIT_MONITOR
4636  11C6  EXIT_NEW_LOCATION  0D4F  EXIT_PROMPTING     11DB  EXIT_QUICK_HOME
4637  1071  EXIT_REGISTER      0EB8  EXIT_SET_USER      0F4B  EXIT_STACK
4638  0F0B  EXIT_STEP          0EFF  EXIT_WATCH         1080  FILL_MEMORY
4639  10B2  FILL_MEMORY1       1E6F  FILL_TEXT1         1E80  FILL_TEXT2
4640  1E90  FILL_TEXT3         F023  FLAG1              1ECD  FLAG_TEXT
4641  1229  FOUND              0598  FUNCTION1          05A3  FUNCTION2
4642  05AE  FUNCTION3          05B9  FUNCTION4          05C5  FUNCTION5
4643  05D0  FUNCTION6          05DB  FUNCTION7          05E6  FUNCTION8
4644  05F1  FUNCTION9          0A19  FUNCTION_2ND      058D  FUNCTION_KEY
4645  13D2  GET_2ND_HEX        1245  GET_COMMAND        1372  GET_HEX
4646  1385  GET_HEX1           139E  GET_HEX2           0BAB  GET_KEY_CODE
4647  0426  GET_NUMBER_OF_BYTE  1403  GET_RECORD         099B  GO
4648  022A  GOTO_XY            0239  GOTO_XY1           0244  GOTO_XY2
4649  024F  GOTO_XY3           025A  GOTO_XY4           0000  GPIO
4650  11EE  HELP               1F3B  HELP_TEXT          1F6E  HELP_TEXT1
4651  12F1  HEX_DUMP           1312  HEX_DUMP1          12FE  HEX_DUMP2
```

```
1338  HEX_DUMP3          133A  HEX_DUMP4          132D  HEX_DUMP5
1EBC  HL_TEXT            0981  HOME               8100  HOME_ADDRESS
0A65  INCREMENT          1489  INC_BCD1           14A1  INC_BCD2
018F  INIT_8254          0208  INIT_LCD           1206  INIT_UART
091F  INSERT_BYTE        0932  INSERT_BYTE1       150B  INS_TABLE
11DC  IO_ADDRESS         2103  IO_TEXT            0705  IT_IS_RAM
1E47  JUMP_TEXT1         1E59  JUMP_TEXT2         10DD  JUMP_TO_USER_PGM
F021  KEY                0581  KEY_EXECUTE        01F6  LCD_READY
01F7  LCD_READY1         000A  LF                 F047  LINE_BUFFER
0186  MAIN               170B  MNEM               0B3D  MODE1
0B74  MODE2              0B02  MODE_INDICATOR     0A3D  MODIFY_REGISTER
0261  MONITOR_CALL       10C4  MONITOR_FUNCTION   226C  MONITOR_TEXT
8000  MY_ROM             12DA  NEW_LINE           11AA  NEW_LOCATION
1DE6  NEW_TEXT           0CDB  NEXT_KEY           1252  NO_DATA
042E  NUMBER1            0476  NUMBER10           047E  NUMBER11
0486  NUMBER12           048E  NUMBER13           0496  NUMBER14
049E  NUMBER15           04A6  NUMBER16           04AE  NUMBER17
04B6  NUMBER18           04BE  NUMBER19           0436  NUMBER2
04C6  NUMBER20           04CE  NUMBER21           04D6  NUMBER22
04DE  NUMBER23           04E6  NUMBER24           04EE  NUMBER25
04F6  NUMBER26           04FE  NUMBER27           0506  NUMBER28
050E  NUMBER29           043E  NUMBER3            0516  NUMBER30
051E  NUMBER31           0526  NUMBER32           052E  NUMBER33
0536  NUMBER34           053E  NUMBER35           0546  NUMBER36
054E  NUMBER37           0556  NUMBER38           055E  NUMBER39
0446  NUMBER4            0566  NUMBER40           056E  NUMBER41
0576  NUMBER42           057E  NUMBER43           044E  NUMBER5
0456  NUMBER6            045E  NUMBER7            0466  NUMBER8
046E  NUMBER9            1D9E  OFF_DISPLAY        0663  OFF_DISPLAY1
13E9  OK_0_9             0366  ONE_TAB            097D  OPTION1
12BA  OUT1X              12C6  OUT1X1             12CB  OUT2X
1EC9  PC_TEXT            01E4  PINT1              01EF  PINT2
01C9  PINT8U             F03C  POINTER            14B9  PRINT_BCD1
14E2  PRINT_BCD2         0EF1  PRINT_WATCH        0EB9  PRINT_WATCH_RAM
1D58  PROMPT1            1DA4  PROMPT2            126F  PROMPT3
0D32  PROMPTING          1DE4  PROMPT_TEXT        025B  PUT_CH_LCD
07ED  PUT_FLAG           07F5  PUT_HIGH1          0876  PUT_HIGH2
1258  PUT_STR            125F  PUT_STR1           12A5  PUT_STR2
021A  PUT_STR_LCD        0221  PUT_STR_LCD1       1079  P_TEXT
11C7  QUICK_HOME         0ABE  READ_MEMORY        0A93  READ_REGISTER
0761  REGISTER1          08A4  REGISTER10         077D  REGISTER2
0799  REGISTER3          07B5  REGISTER4          07D1  REGISTER5
07F8  REGISTER6          0820  REGISTER7          084B  REGISTER8
087C  REGISTER9          0F4C  REGISTER_DISPLAY   0F54  REGISTER_DISPLAY1
0F57  REGISTER_DISPLAY2  0FE9  REGISTER_FLAG1     106A  REGISTER_FLAG10
0FEE  REGISTER_FLAG2     1008  REGISTER_FLAG3     100D  REGISTER_FLAG4
1027  REGISTER_FLAG5     102C  REGISTER_FLAG6     1046  REGISTER_FLAG7
104B  REGISTER_FLAG8     1065  REGISTER_FLAG9     0000  ROM
0000  RS                 F057  SAVE_STACK         0CAE  SCAN
0CBC  SCAN1              0C7E  SCAN_KEY           0C95  SCAN_KEY1
0C8F  SCAN_KEY2          0C8F  SCAN_KEY3          0C7E  SCAN_KEY4
0743  SELECT_REGISTER    0D3A  SEND_PROMPT        12AC  SEND_PROMPT1
12B3  SEND_PROMPT3       12EB  SEND_TAB           0CED  SERIAL_COMMAND
02B8  SERVICE_RST2       0293  SERVICE_RST7       02DD  SERVICE_TRAP
1F0C  SET_REGISTER_TEXT  0DE0  SET_USER1          0E0B  SET_USER2
0E36  SET_USER3          0E61  SET_USER4          0E8C  SET_USER5
0EB7  SET_USER6          0DA4  SET_USER_REGISTER  071F  SHIFT_ADDRESS
06E2  SHIFT_DATA         0CD1  SHIFT_KEY          067A  SHIFT_REGISTER
1EE3  SIGN_TEXT          09BF  SINGLE_STEP        0F00  SINGLE_STEP_
030E  SKIP1              09DF  SKIP2              0141  SKIP_BOOT
0164  SKIP_COLD_BOOT     118A  SKIP_EDIT1         1193  SKIP_EDIT2
147D  SKIP_ERROR         1117  SKIP_LOAD_PC       07F7  SKIP_PUT_HIGH1
0878  SKIP_PUT_HIGH2     017B  SKIP_SEND_PROMPT   0D31  SKIP_SERIAL
0020  SP                 12E5  SPACE              1EC0  SP_TEXT
0F0C  STACK_DISPLAY      0F20  STACK_DISPLAY1     1EF4  STACK_TEXT
0100  START              0010  SYSTEM_PORT_A      0011  SYSTEM_PORT_B
0012  SYSTEM_PORT_C      0013  SYSTEM_PORT_CONTROL  F000  SYSTEM_RAM
F059  SYSTEM_STACK       0009  TAB                F03A  TEMP
062D  TEST_BUZZER        05F2  TEST_LED           05F4  TEST_LED1
1DB9  TEXT3              0B94  TITLE              F036  TOS
1EC4  TOS_TEXT           1366  TO_HEX             0A88  TO_SEVEN_SEGMENT
0040  UART_BUFFER        0040  UART_DIVISOR_LSB   0041  UART_DIVISOR_MSB
0042  UART_FIFO          F025  UART_FOUND         0043  UART_LCR
0045  UART_LINE_STATUS   0047  UART_SCR           F02C  USER_AF
F02E  USER_BC            F030  USER_DE            F032  USER_HL
F02A  USER_PC            0030  USER_PORT_A        0031  USER_PORT_B
```

```
0032  USER_PORT_C        0033  USER_PORT_CONTROL  F034  USER_SP
F079  USER_STACK         0275  VECTOR_TABLE       0CC6  WAIT1
1436  WAIT_CR            F029  WARM_CODE          0EC9  WATCH1
F000  WATCH_RAM          1371  ZERO_NINE          1072  ZERO_TEXT
```