Experiment

1

Flight 8086 Training Board

Objective

The aim of this lab experiment is to familiarize the students with Flight 8086 training board.

Equipment

Flight 8086 training board, PC with Flight86 software, download cable.

Tasks to be Performed

- Connecting the 8086 training board to PC (using COM1 port)
- Study of different commands provided by the training board
- Program Entry, Execution and Debugging
- Assembling and disassembling of a program
- Displaying the contents of registers and memory locations
- Modifying the registers and memory contents
- Single-step execution and Breakpoint insertion
- Downloading & uploading a program file.
- Running simple programs to perform
    1. Arithmetic operations
    2. Finding the smallest/largest number from a given list of numbers
    3. Searching for a given number in a list of numbers.

1.1 Background

The FLIGHT 86 Trainer System is designed to simplify the teaching of the 8086 CPU and some of its commonly used peripherals. It can be linked to most PCs with a simple serial line, so that code may be assembled and debugged in a supportive software environment before being downloaded into the RAM on the board. The board itself may then be linked to other peripheral devices. A block diagram of this mode of operation is shown in Figure 1.1.

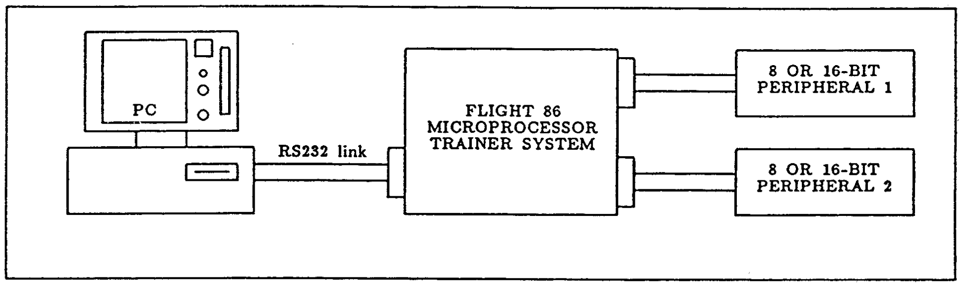

Figure 1.1: Block Diagram of the FLIGHT-86 Trainer System

Once downloaded, the code may be executed and examined in a system which is accessible to the user. Data may be manipulated on the board and the effects viewed on the PC. The software which handles this two-way transfer is supplied with the board, in the form of a monitor program resident on the board in EPROM, and a disk containing the "host" software for the PC.

1.2 Connecting the Training Board to PC

Figure 1.2 shows the FLIGHT-86 Trainer Board layout. The first step is to connect the *serial socket* (P3) on the training board to COM1 in the PC using RS323 cable. Next, connect the power cable to the *power supply connector* (P6). Finally, load the program F86GO.BAT on the PC. This should run and report the amount of RAM and EPROM on the FLIGHT-86 board, before returning the prompt as shown in Figure1.3.

1.3 Commands Provided by Flight-86

A '-' prompt on the screen means that the host is ready to accept a command. Table1.1 gives a summary of the commands that will be used during this experiment.

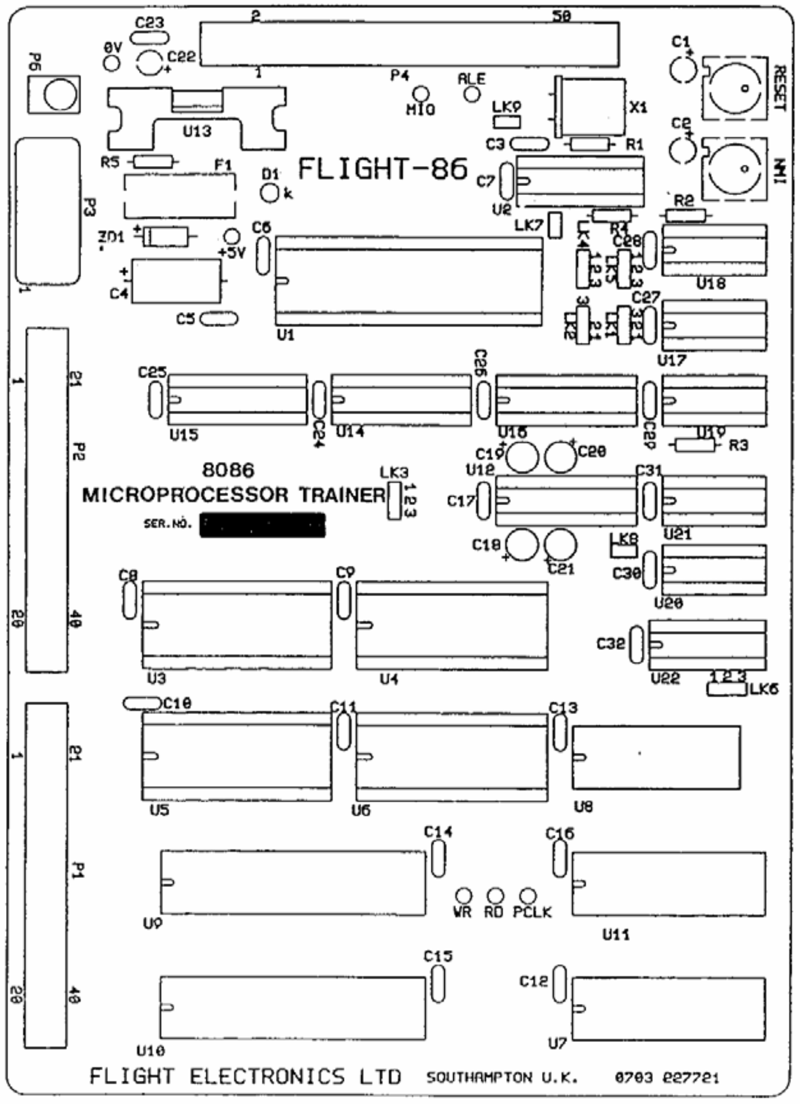

Figure 1.2: Layout of the FLIGHT-86 Training Board

```
Loading FLIGHT86 host program, please wait...
FLIGHT86 Controller Board, Host Program Version 2.1
Press ? and Enter for help - waiting for controller board response...
ROM found at F000:C000 to F000:FFFF Flight Monitor ROM version 2.0
RAM found at 0000:0000 to 0000:FFFF
-
```

Figure 1.3: Starting Message of the FLIGHT-86 Training Board

Table 1.1: Summary of some commands provided by FLIGHT-86

KEY	PARAMETER	DESCRIPTION
ESC		Press the Escape button to stop the current command
X		Resets the training board
Q		Terminates running of the board software and returns control to the operating system
?	[*command letter*]	Help
R	[*register*]	Allows the user to display or change the content of a register
M	[W][*segment*:] *address*1 [*address*2]	Allows the user to display or change one or more memory locations
A	[[*segment*:] *address*]	Allows the user to write 8086 assembly code directly into the training board
Z	[[V] [*segment*:] *address*1 [*address*2]]	
G	[[*segment*:] *address*]	Allows the user to execute code that has been downloaded into RAM
B	? | R | S [*segment*:] *address*	Allows the user to Display/Clear/Set break points inside his code
S	[R][[*segment*:] *address*]	Allows the user to step through code one instruction at a time
:	[*drive*:\path] *filename*	Loads an Extended Intel Hex file from disk into the memory of the training board

1.4 The First Program

Assembling a Program (Command A)

The assemble command (A [*segment*:] *address*) allows you to type in 8086 assembly code, and this code will be assembled and stored into the board memory. The following example shows you how to write a simple program using this command

Example 1.1: Using the assemble command, write a program that will add the content of two memory locations (words) and store the result in a third memory location.

1. Start the line assembler at the desired address by entering **A 0050:0100** (Note that the origin address for user RAM on the FLIGHT-86 is 0050:0100)

2. The FLIGHT-86 responds by echoing the address **0050:0100**

3. Now enter the assembly code one instruction at a time hitting ENTER after each instruction

4. Each time, the FLIGHT-86 responds by echoing the next address

5. When you are done exit from the line assembler by pressing ESC button

The screen will now look like

```
A 0050:0100
0050:0100  DW  0002
0050:0102  DW  0003
0050:0104  DW  0000
0050:0106  MOV AX,[0100]
0050:0109  ADD AX,[0102]
0050:010D  MOV [0104], AX
0050:0111  INT 5
0050:0113
-
```

Disassembling a Program (Command Z)

You can examine what you have entered using the disassemble command. If you type **Z 0050:0100 0111**, then the content of the memory locations between the addresses 0050:0100 and 0050:0111 will be disassembled as follows:

```
0050:0100  02  00          ADD AL, [BX+SI]
0050:0102  03  00          ADD AX, [BX+SI]
0050:0104  00  00          ADD [BX+SI], AL
0050:0106  A1  01  00      MOV AX,[0100]
0050:0109  03  06  02  01  ADD AX,[0102]
0050:010D  89  06  04  01  MOV [0104], AX
0050:0111  CD  05          INT 5
```

The HEX numbers between the addresses and the instructions represent the opcodes of the disassembled instructions. Notice that memory words entered as DW directives have been disassembled as ADD instructions with different parameters. This because the values of these memory words are equivalent to the opcode of the ADD instruction with the shown parameters.

Running a Program (Command G)

To run the above program enter **G 0050:0100** and press the ENTER key. The program will now run, load the word at address 0050:0100 into AX, add the content of the word at address 0050:0102 to the content of AX, store the result into the word at address 0050:0104, and terminate. Note that the instruction **INT 5** is responsible for terminating the program.

Displaying/Modifying Memory Locations (Command M)

To test the result of the above program enter **M W 0050:0104** and press the Enter key. This will display the memory word at address 0050:0104 where the result of the above program is stored. Exit from this command by pressing the ESC key.

Lets now change the content of the memory words stored at addresses 0050:0100 and 0050:0102. At the command prompt '-', enter M W 0050:0100 and press the Enter key. The content of the memory word at address 0050:0100 is displayed. To change the content of this memory location, enter a HEX number (say 0005) and press the Enter key. The content of the next memory location is displayed. Enter another HEX number (say 0007) and press the Enter key. When the content of the next memory location is displayed, press the ESC key to go back to the command prompt. These steps are shown below:

```
-M W 0050:0100
0050:0100 0002  0005
0050:0102 0003  0007
0050:0104 0005
-
```

Now run the program again and test the content of the memory word at address 0050:0104.

Breakpoint Insertion (Command B)

This command is intended for debugging user code. A breakpoint is an **INT 3** instruction inserted at an opcode position. The original opcode at this address is saved. When the code is executed it runs normally, at full speed, until it reaches this location. Then, original opcode is restored and the registers, address and first opcode byte are displayed. The user may set another break point and continue with a G instruction.

As an example, enter the command **B S 0050:010D** and press the Enter key. This will set a breakpoint at address 0050:010D in the previous program (i.e. a breakpoint is set at the instruction MOV [0104], AX). Now, run the program using the command **G 0050:0100**. Notice that the program terminates and the message "Monitor breakpoint at 0050:010D" is displayed. This means that the execution of the program stopped at location 0050:010D. You can resume the execution of the program using the command G, but let us first modify the content of register AX. At the command prompt '-', enter the command **R AX** and press the Enter key. This will display the content of AX which is 000D (i.e. 0005+0007). Modify this value by entering 0001 next to 000D and press the Enter key then ESC to go back to the command prompt. Now, continue the execution of the program from address 0050:010D using the command **G 0050:010D**. Check the content of memory word at address 0050:0104.

The previous steps are shown below:

```
-B S 0050:010D
-G 0050:0100
Monitor Breakpoint at 0050:010D
-R AX
AX 000C  0001
BX 0000
-G 0050:010D
User Break at 0050:0111
-M W 0050:0104
0050:0104  0001
0050:0106  00A1
-
```

Single-Step Execution (Command S)

This command is provided to allow the user to step through code one instruction at a time for debugging purposes. The display will be the next instruction address and opcode byte with, optionally, registers content. Once the command has started, pressing the Enter key will execute the next instruction. As an example, enter the command **S R 0050:0100** and press the Enter key. This will execute the first instruction and terminate with registers content shown on the screen. When you press Enter again, the next instruction is executed. Continue pressing the Enter key until all instructions in the program get executed, or press the ESC key to terminate the command.

1.5 Writing a Program Using Assembler on a PC

In the pervious section, we have used the assemble command to write and load simple assembly instructions into the board memory. However, for more sophisticated applications, you need to write and assemble programs on a PC before downloading them into the board memory. For this purpose, you need the following programs:

- **MASM:** as the assembler and linker
- **EXE2BIN:** to convert from and executable file into a binary file
- **OBJECT86:** to convert the binary file into an INTEL HEX file for download to the FLIGHT-86

Example 1.2: Write a program to search for a given number in a list of numbers. You should define the list as a sequence of memory bytes labeled with the letter A. The number to be searched is passed through register DL. When the program terminate, BX should contain the index of the number in the list if the number is in the list.

```
COMSEG SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:COMSEG, DS:COMSEG, ES:COMSEG, SS:COMSEG
ORG 0100h
start:
      MOV   AX,   CS
      MOV   DS,   AX ; Set the data segment

      MOV   BX,   0  ; Set BX to index of the 1st element in
                     ; the list
L1:   CMP   BX,   8  ; if BX exceeds the indices of the list
      JZ    L2       ; then end the search

      CMP   DL,   A[BX] ; if the number is found in the list
      JZ    L2          ; then end the search
      INC   BX          ; else increment BX
      JMP   L1
L2:   INT   5           ; terminate the program
A     DB    4
      DB    2
      DB    7
      DB    6
      DB    3
      DB    5
      DB    1
      DB    8
COMSEG ENDS
END start
```

Using any text editor on the PC enter the previous code. Notice that the code shown in Bold is required for every program using **MASM** and can be thought of as a template. Now, save this file as **SEARCH.ASM**. Using the Assembler, i.e. MASM, assemble and link this file to produce **SEARCH.EXE**, and using **EXE2BIN** create the binary code file **SEARCH.BIN**. Now, using **OBJECT86**, convert this binary file to the Intel hex format file **SEARCH.HEX**. Finally load the HEX file into the board memory using the command "**:SEARCH.HEX**". Note, you may need to specify the drive and the path of the file if it is not in the same directory as the F86GO software (e.g. :C:\MyProjects\Search.hex).

To run this program, first load the required number into DX using the command **R DX**. Next, run the program using the command **G 0050:0100**. Finally, use the command **RX BX** to check result of the search (i.e. the value of BX represents the index of the given number in the list). These steps are shown below.

```
-R DX
DX 0000  0003
SP 0500
-G 0050:0100
User Break at 0050:011A
-R BX
BX 0004
-
```

Exercises

1.1. Modify the program in **Example 1.1** to perform the four basic operations: *addition*, *subtraction*, *multiplication*, and *division*. The required operation is specified by loading DX with the appropriate value (i.e. 1 for addition, 2 for subtraction, 3 for multiplication, and 4 for division).

1.2. Write a program to find the smallest number from a given list of numbers. Load this program into the FLIGTH-86 and test it.