Experiment No. 01

1.1 Experiment Name

Introduction to Assembly Language Programming and Kit Mode of MDA 8086 Trainer Board

1.2 Objectives

- To understand the structure of Assembly Language programming
- To learn about the microprocessor emulator "Emu 8086" and its operation
- To get acquainted with the "MDA 8086" Trainer Board and its operation
- To learn how to implement program in “MDA 8086” Trainer Board and interconnect it with “Emu 8086”

1.3 Theory

The Assembly language is a low-level computer programming language that consists primarily of symbolic versions of a computer's machine language. Different manufacturers' computers use different machine languages and require different assemblers and assembly languages.

A microprocessor is a type of computer processor that contains the capabilities of a central processing unit on an integrated circuit (IC) or a few integrated circuits that comprises the arithmetic, logic, and control circuitry required to perform the central processing unit functions of a digital computer.

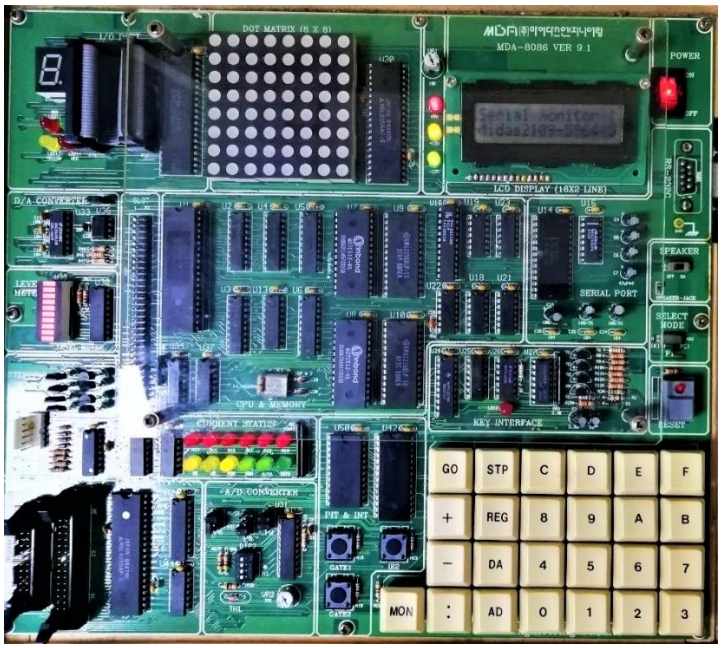

Fig 1.1: MDA 8086 Trainer Microprocessor Kit

The 8086 microprocessor has eight general-purpose registers. They are as follows: AX, BX, CX, DX, SP, BP, SI, and DI. The accumulator is AX. To perform 8-bit instructions, it is separated into two 8-bit registers, AH (upper byte) and AL (lower byte). The MOV operation moves the data item specified by its second operand (register, memory) to the location

specified by its first operand. MOV AX, 10D, for example. The source operand is 10D, while the destination operand is AX. The final result is really accumulated at the destination. In 1978, Intel debuted the 8086 microprocessors. This 16-bit microprocessor marked a significant advancement over the previous generation.

The MDA 8086 microprocessor has the following features:

- A 16-bit microprocessor
- A 20-bit address bus allows it to directly access 220 bits, or 1 MB, of memory.
- The 8086 has fourteen sixteen-bit registers.
- Clock frequency (5 to 10 MHz)

The MDA 8086 consists of a central processing unit (CPU), ROM, SRAM, display, keyboard, speaker, DOT matrix LED, A/D & D/A converter, stepping motor.

It also has function key which operate as follows

[AD]	Set memory address	[GO]	Go to user's final program instruction
[DA]	Update segment & offset, and input data to memory	[REG]	Register display
[STP]	Executing user's program in single steps	[MON]	Immediately break user's program and non-makeable interrupt
[:]	Offset data		

The working operation are as follows

- After imitating the code, it was saved in the software emu8086
- It executed the software step by step by clicking the single step
- The value 10D was first recorded in register AX
- By clicking the single step, the value 17H was recorded in register BX
- Finally, after clicking a single button, the addition of AX and BX, or 10D and 17H, was saved in register AX and displayed on the screen as 21H
- The same phenomenon happens in the MDA 8086 microprocessor kit, and it can be verified step by step.

1.4 Apparatus

- Emu 8086 - Microprocessor Emulator
- MDA 8086 - Trainer Board

1.5 Emulator Code

MOV AX, 10D; MOVE 10D TO AX

MOV BX, 17H; MOVE 1122H TO BX

ADD AX, BX; ADD AX AND BX, AND THE ADDITION WILL BE STORED IN AX

1.6 Output

Emu 8086

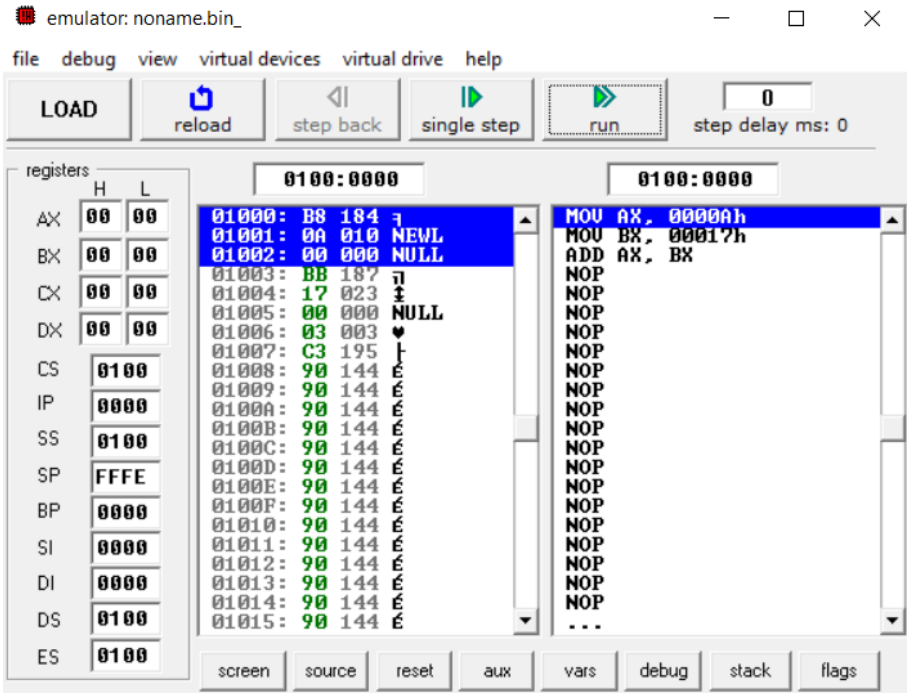

Fig 1.1: Emulator output for Assembly Language Program (After Step 0)

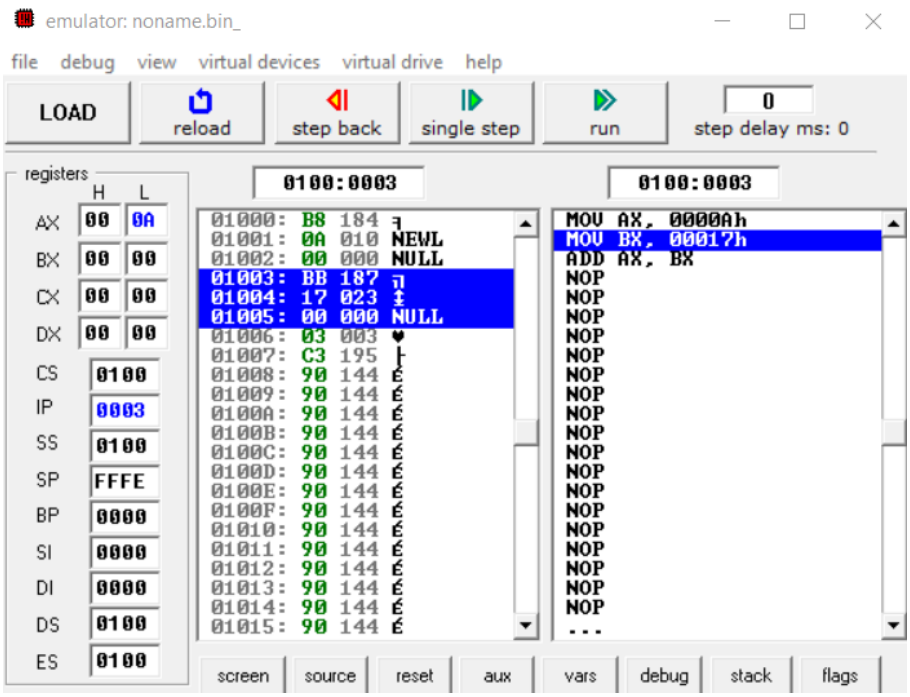

Fig 1.2: Emulator output for Assembly Language Program (After Step 1)

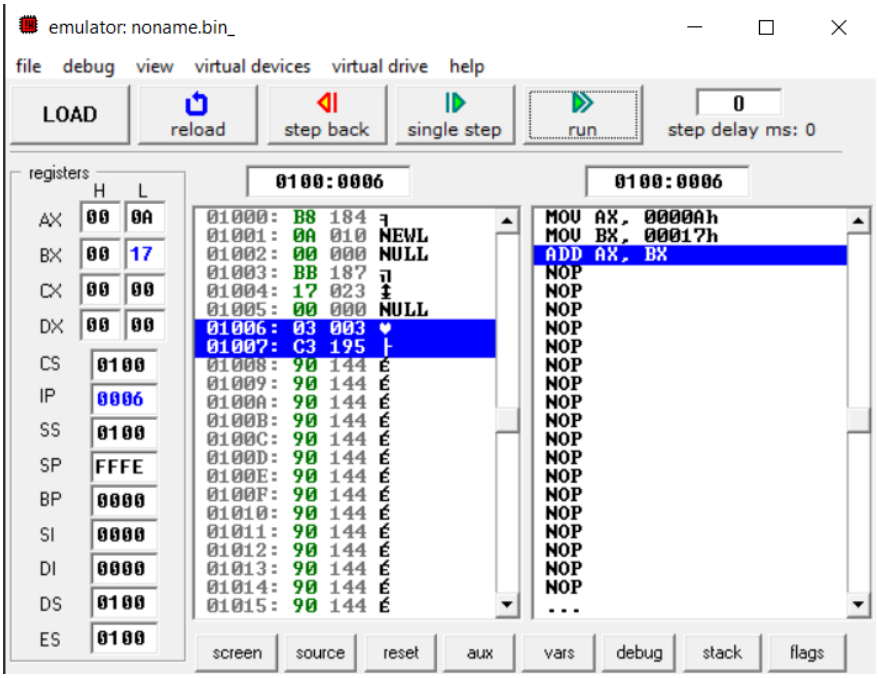

Fig 1.3: Emulator output for Assembly Language Program (After Step 2)

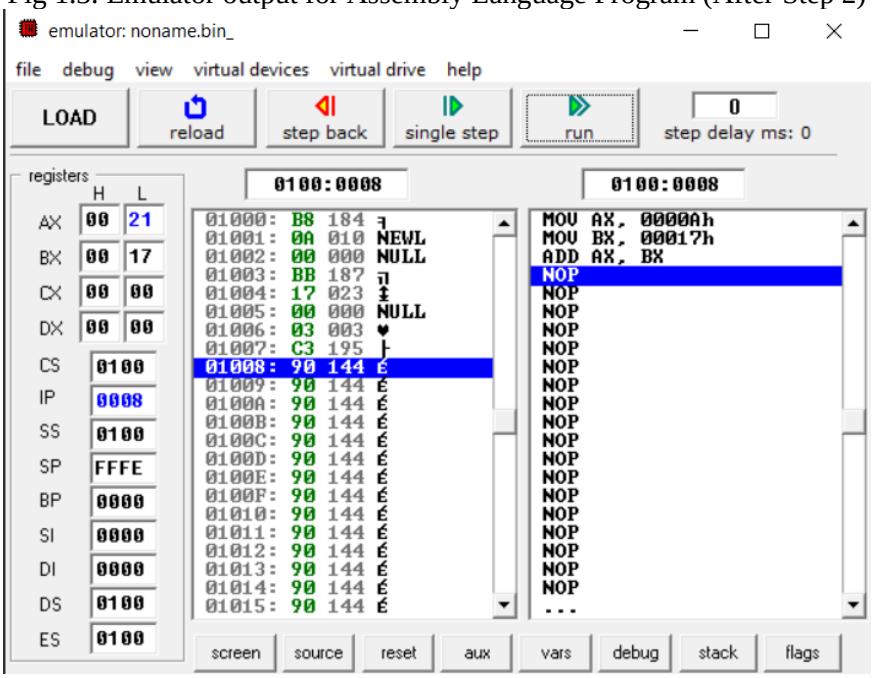

Fig 1.4: Emulator output for Assembly Language Program (After Step-3)

MDA 8086

Fig 1.4: Trainer kit output for Assembly Language Program

1.7 Discussion & Conclusion

In this experiment, the MDA-8086 kit used includes a central processor unit and other components. During this experiment, we became acquainted with the MDA-8086 microprocessor kit and the EMU program.

The instructions are initially stored in memory in a sequential order. The microprocessor reads the instructions from memory, decodes them, and executes them until the STOP command is reached. It then transmits the binary result to the output port. The register stores temporary data between these steps, while the Assembly Language performs computing functions, in this case, we utilized the ADD instruction. In the end, output from emulator and trainer kit was compared and was similar.