Contents

[1.Title: 2](#_Toc83253221)

[2. Abstract: 3](#_Toc83253222)

[3. Objective: 3](#_Toc83253223)

[4. Results: 3](#_Toc83253224)

[4.1. Simulation Environment: 3](#_Toc83253225)

[4.2. Simulation Results: 5](#_Toc83253226)

[a. Exchange program 5](#_Toc83253227)

[b. Addition Program: 6](#_Toc83253228)

[c. Subtraction Program: 7](#_Toc83253229)

[d. Loop B1: 8](#_Toc83253230)

[4.3. Discussion/Comments on Simulation Results: 8](#_Toc83253231)

[5. Lab Task: 9](#_Toc83253232)

[Lab Task 1: 9](#_Toc83253233)

[Lab Task 2: 9](#_Toc83253234)

[Lab Task 3: 10](#_Toc83253235)

[a. Immediate addressing mode 11](#_Toc83253236)

[b. Register addressing mode 12](#_Toc83253237)

[c. Direct memory addressing mode 12](#_Toc83253238)

[d. Register based indirect addressing mode 12](#_Toc83253239)

[e. Register relative addressing mode 12](#_Toc83253240)

[f. Base indexed addressing mode 12](#_Toc83253241)

[g. Relative based indexed addressing mode 13](#_Toc83253242)

[h. Implied addressing mode 13](#_Toc83253243)

[Conclusion : 14](#_Toc83253244)

[References: 14](#_Toc83253245)

**MICROPROCESSOR & EMBEDDED SYSTEM LAB**

**Fall 21-22**

**Lab Report-1**

# 1.Title:

Introduction to Microprocessor 8086, 8086 instructions and programming with 8086.

|  |  |  |
| --- | --- | --- |
| **Group Member’s Name** | **ID** | **Department** |
| Ananya Chowdhury**(Author)** | 18-39028-3 | CSE |
| Durjoy Dey**(Author)** | 18-39013-3 | CSE |
| Nurul Huda Bhuiyan Rokon | 18-38983-3 | CSE |
| Tonima Hossain Pia | 18-38892-3 | CSE |
| Joy Karmakar | 18-39263-3 | CSE |
| Sazid Al Farabi | 19-39478-1 | CSE |
| Sharika Parvin Joba | 18-38721-3 | CSE |
| Salma Jahan Sahara | 18-38788-3 | CSE |

**Performance Date:** 16.09.21

**Submission Date:** 23.09.21

**Faculty Name:** DR. FERDOUS JAHAN SHAUN

**Section:** M

**Group:** B

**Submitted By:** Ananya Chowdhury, ID: 18-39028-3

# 2. Abstract:

From this experiment, our main goal is to understand about the working principle of MTS-86c& MDA 8086 and to be familiar with emulator EMU8086 by using a simple program to test its different kinds of uses and introduction to segmented memory technology used by Microprocessor 8086. 8086 is a 16bit microprocessor that has a memory of one MB. We have known about the three steps of the microprocessor which are fetch, decode and execute. In the experiment, we got to know about the internal architecture of the 8086 microprocessor which is divided into two parts. They are bus interface unit and execution unit which we have known from this experiment. From this experiment, we got to know about the read and write operation. How the emu8086 runs the program of codes and executes them we will also understand that.

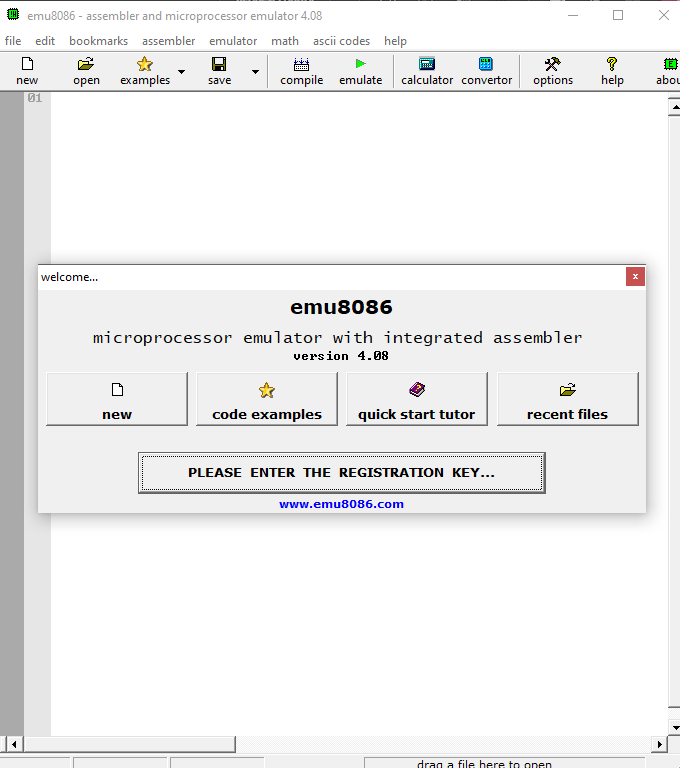
# 3. Objective:

There are some objectives of this experiment we should be looking like how the microprocessor 8086 will work with the pipelining option. Also, we will know how a microprocessor collects and store data. It also explains the architecture of the internal of it. We will also know general-purpose and segment registers from this experiment and also we get to know about the segment and offset address of them and the whole thing is called the logical address which we will know from this experiment. We will also know the assembly programs using 8086 Microprocessor. We will also know how the control system operates all the stages of microprocessors by sending the control signals. As this microprocessor is 16bit, there is 9 used register. They are also divided into two categories which are status and the control register which we will know from this experiment.

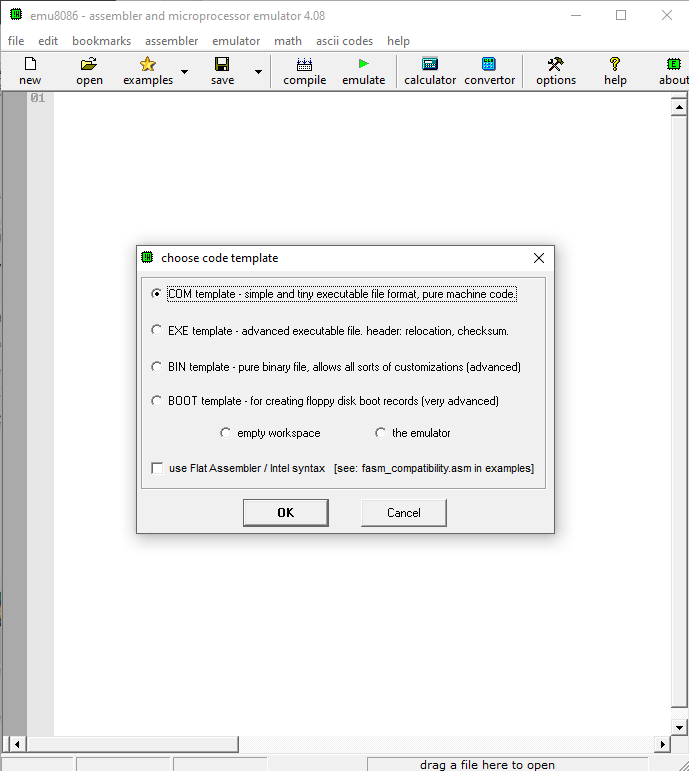
# 4. Results:

## 4.1. Simulation Environment:

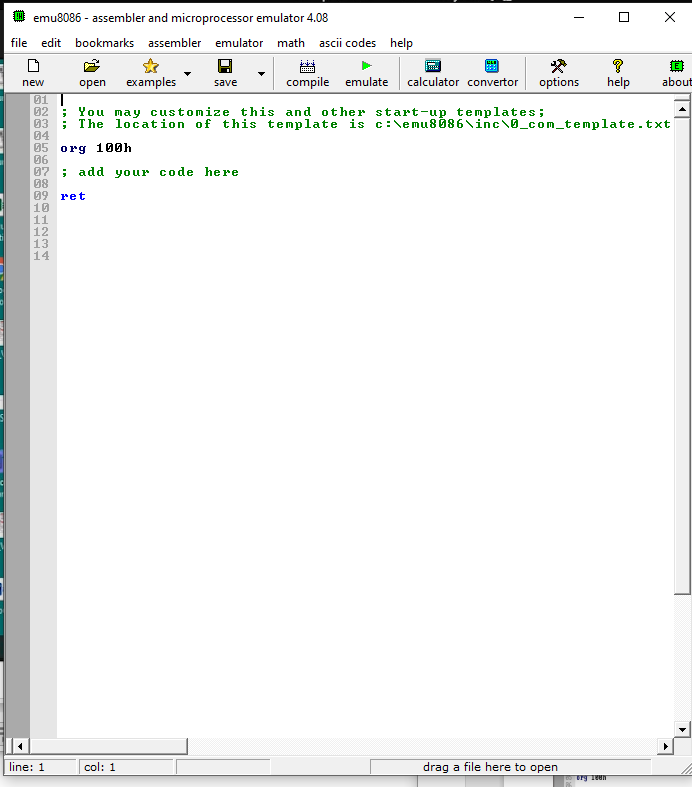
Here we are using EMU8086 for assembly programming and in below we are showing the start menu after opening the software.



Then we should click on the new tab.



After that select the first option and select ok then.



After that this page will appear and we will write our code.

## 4.2. Simulation Results:

Now we will write a program of exchanging the value of two registers and in below we will show how the program works.

## Exchange program

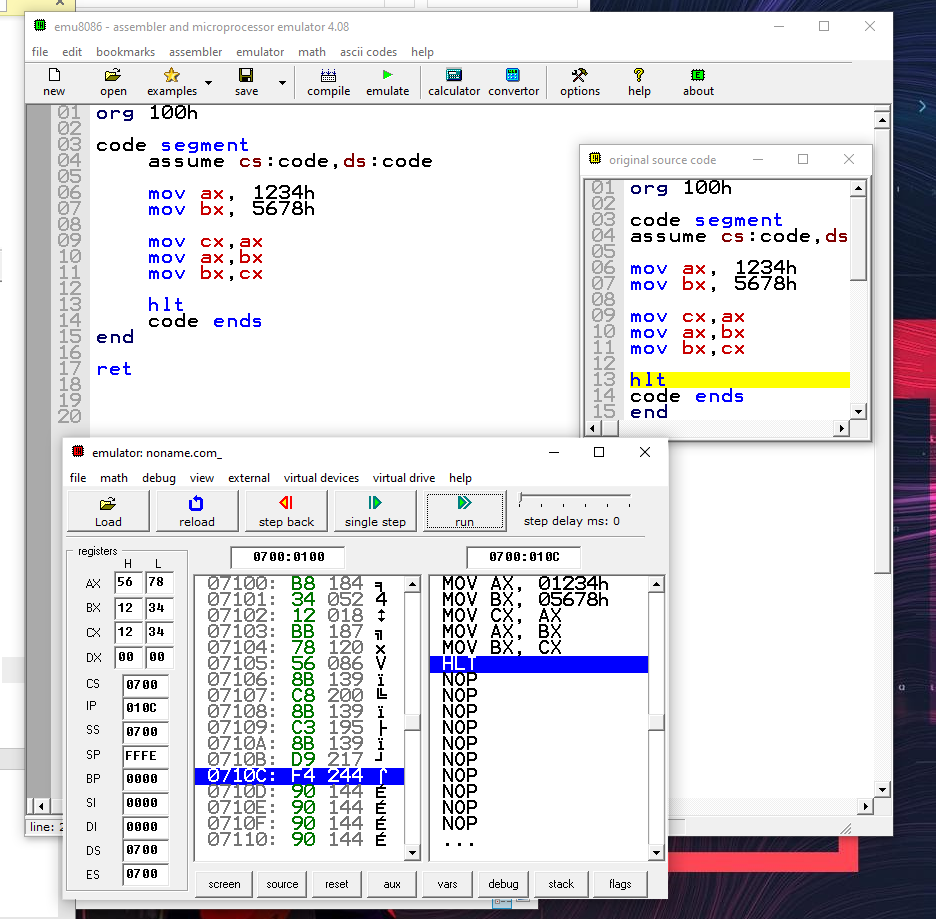


Figure 1 Exchange program 1

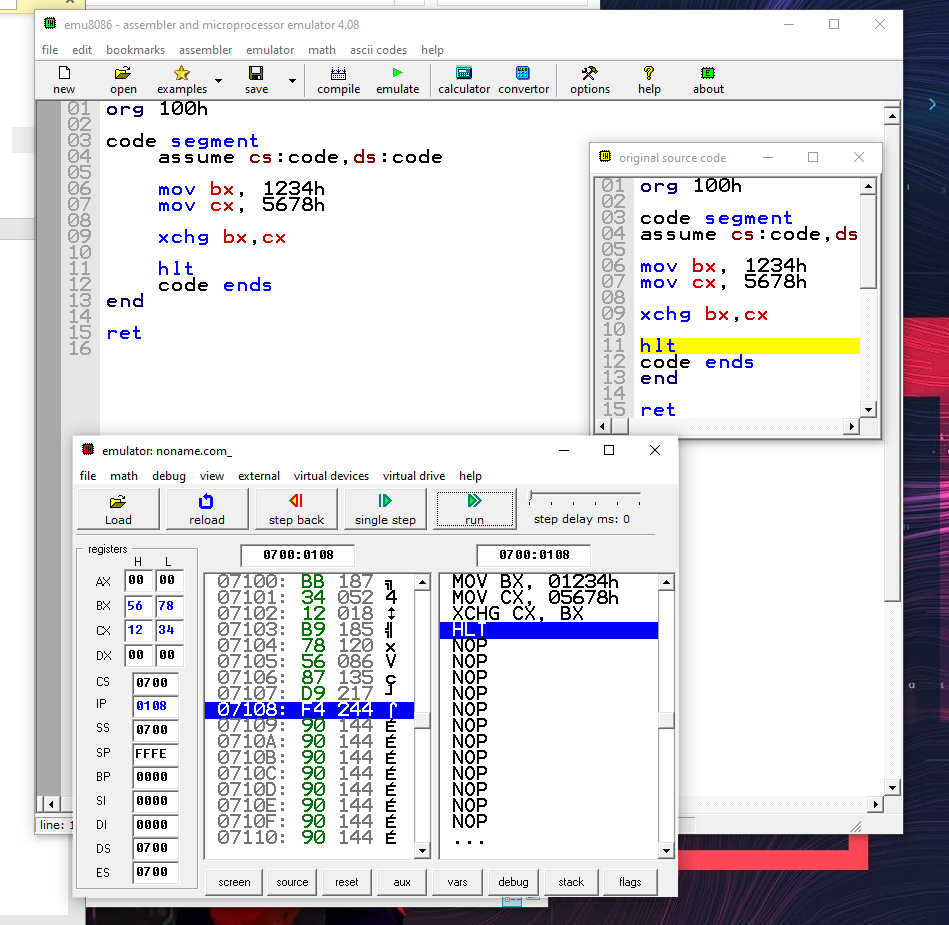


Figure 2 Exchange Program 2

In Fig:1 and Fig:2 we can see the screenshot of the program where the results we got are ax = 5678 and bx=1234 which were expected from the programming.

## Addition Program:

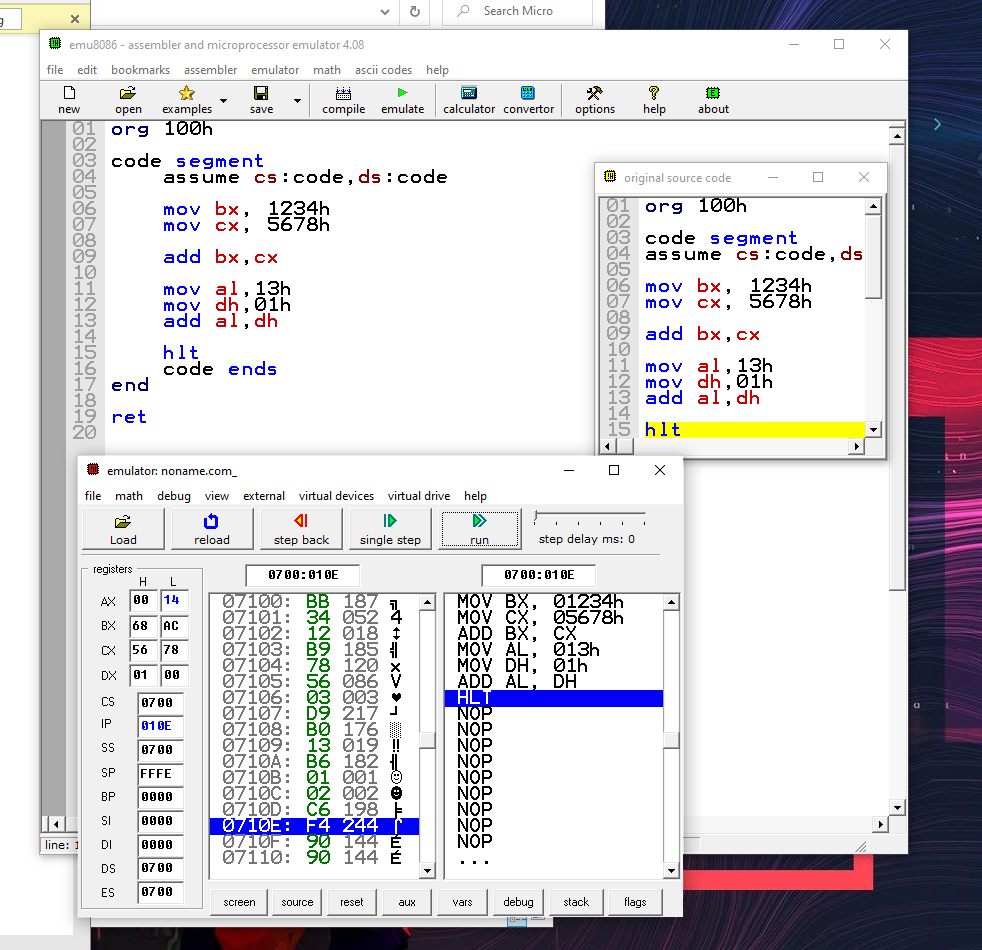


Figure 3 Addition Program

Here we can see the program of addition where the result is as same as we expected.

## Subtraction Program:

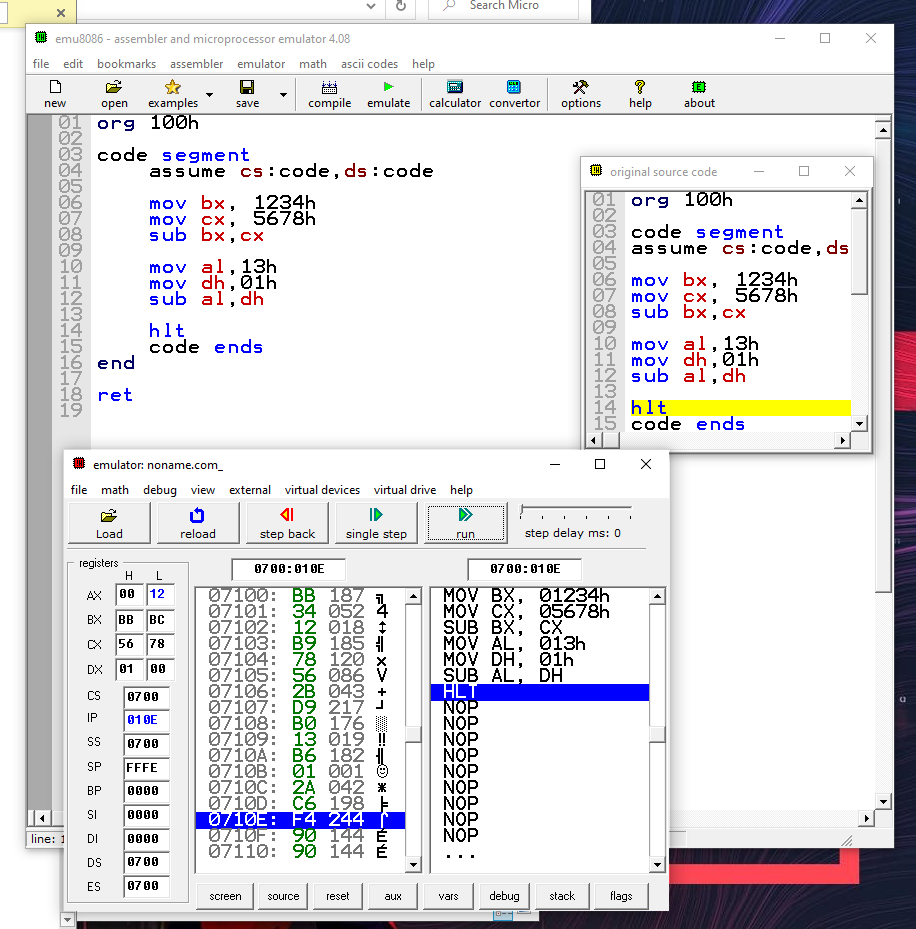


Figure 4 Subtraction Program

Here we can see the program of subtraction where the result in as same as we expected.

## Loop B1:

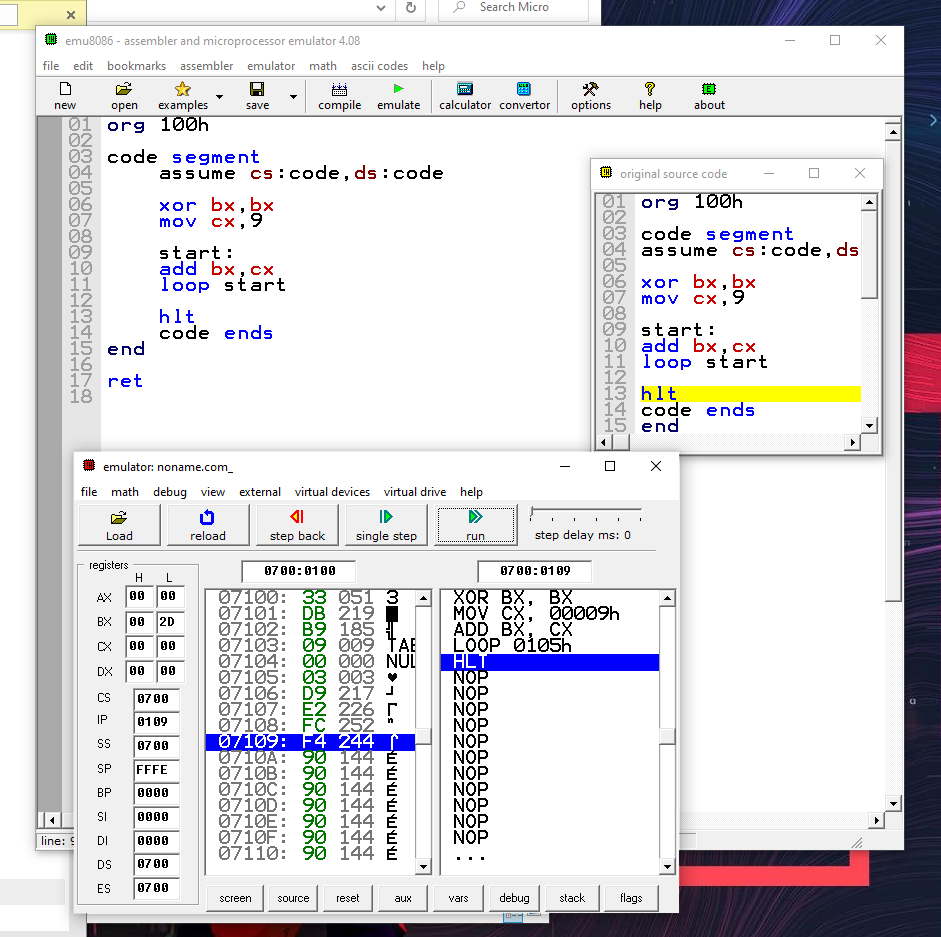


Figure 5 Loop B1

Here we can see the program of loop where the result in as the same as we expected.

## 4.3. Discussion/Comments on Simulation Results:

For doing this program the spacing of the codes and right commands is necessary otherwise the code will not run properly. And also all the register's values must be input correctly otherwise the value of the results for the program will be different and we will not get an expected result.

# 5. Lab Task:

## Lab Task 1:

All the programs of lab task 1 are given above where we discussed the simulation results.

## Lab Task 2:

**Write the assembly language** DX= AX + BX – CX

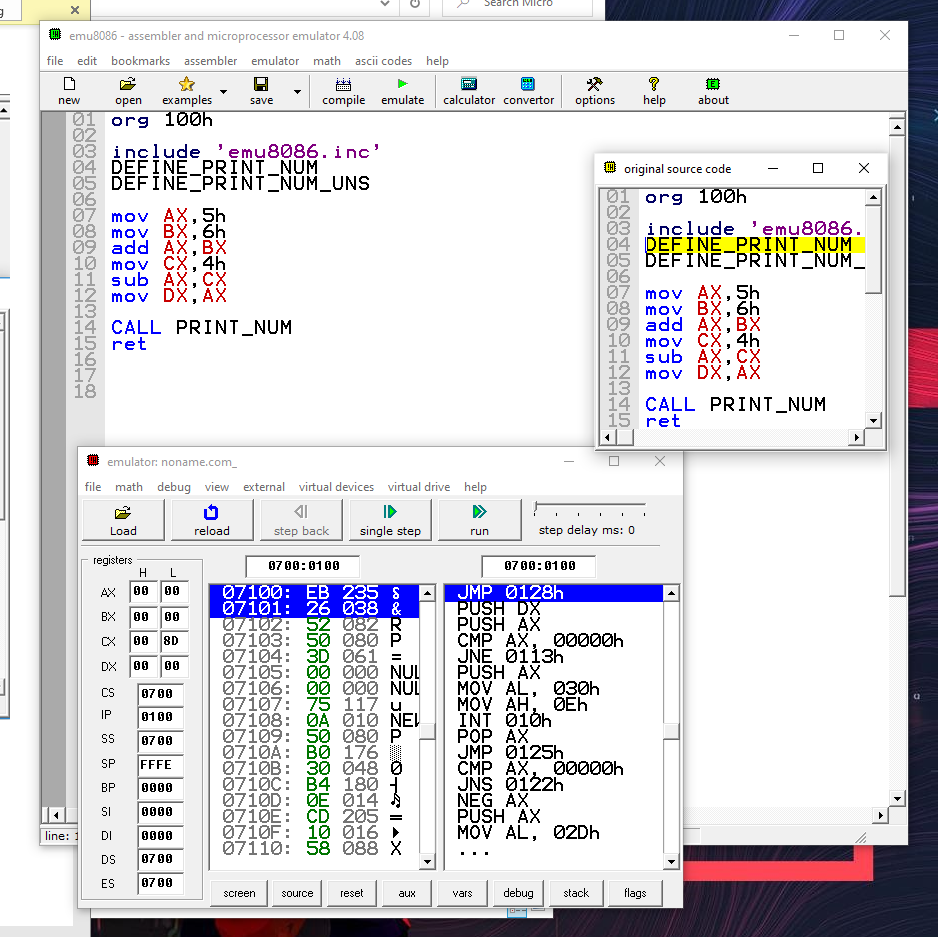


Figure 6 Lab Task 2

## Lab Task 3:

Write a program that displays two characters at column#12 and row#7 at emulator screen

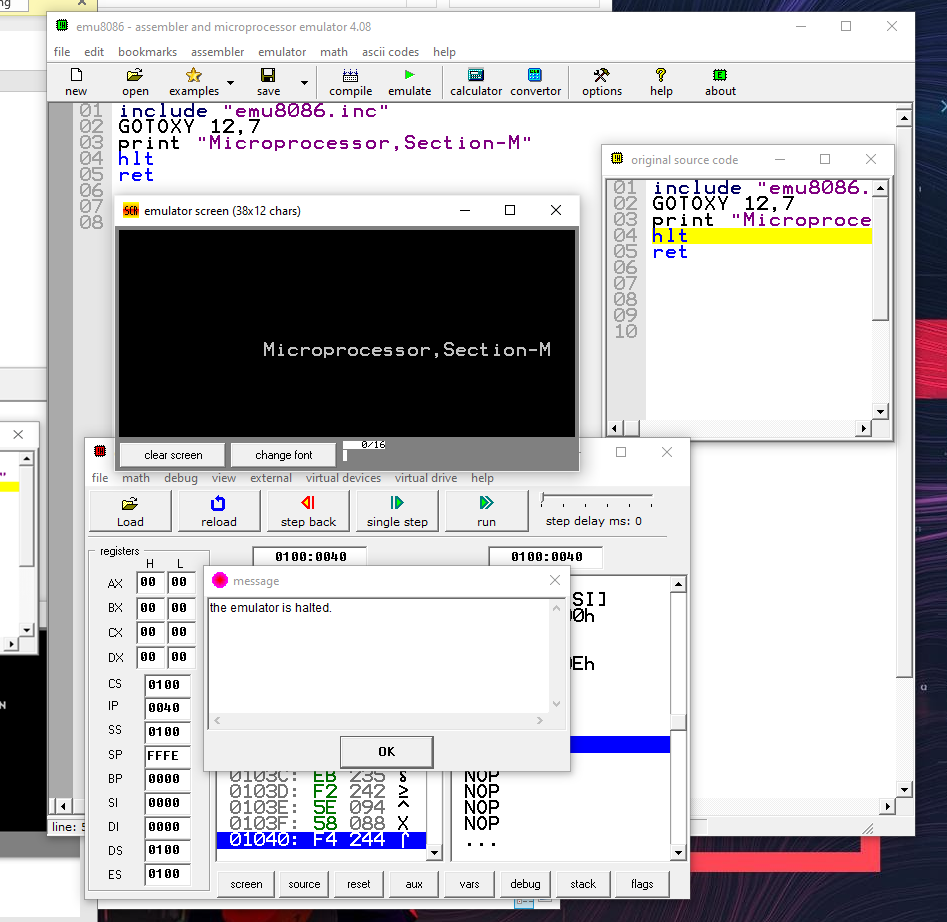


Figure 7 Lab Task 3

**Questions & Answers for Report writing :**

1. **Include all codes’ list file printout following lab report writing template mentioned in appendix A**

Ans:- Enclosed in the Lab Task segment and simulation results.

1. **What is the advantage of having overlapping segments in 8086 memory system?**

Ans:- Advantages of memory overlapping segmentation in the 8086 are as given below :

* With the help of memory segmentation, a user can work with registers having only 16-bits.
* By memory segmentation, the various portions of a program can be more than 68kb.
* The data and the user's code can be stored separately allowing for more flexibility.

1. **For a memory location with a physical address 1256Ah, Calculate the address in the segment: offset form for segments 1256h and 1240h.**

Ans :- Given,

Physical Address = 1256h

Logical address for segment 1256h = Segment: Offset

= 1256h: 000Ah

Logical address for segment 1240h = Segment: Offset = 1240h: 016Ah

1. **What are the different data addressing modes available in 8086? Briefly explain each of them with examples.**

Ans:- The way of specifying data to be operated by an instruction is known as addressing modes. This specifies that the given data is immediate data or an address. It also specifies whether the given operand is register or register pair. There are 8 different addressing modes in 8086 programming –

## a. Immediate addressing mode

* In this mode, the operand is specified in the instruction itself. Instructions are longer but the operands are easily identified.
* Example: MOV CL, 12H
* This instruction moves 12 immediately into the CL register. CL ← 12H

## b. Register addressing mode

* In this mode, operands are specified using registers. This addressing mode is normally preferred because the instructions are compact and the fastest executing of all instruction forms.
* Registers may be used as source operands, destination operands, or both.
* Example: MOV AX, BX
* This instruction copies the contents of the BX register into the AX register. AX ← BX

## c. Direct memory addressing mode

* + In this mode, address of the operand is directly specified in the instruction. Here only the offset address is specified, the segment being indicated by the instruction.
  + Example: MOV CL, [4321H]
  + This instruction moves data from location 4321H in the data segment into CL.
  + The physical address is calculated as DS \* 10H + 4321, Assume DS = 5000H

∴ PA = 50000 + 4321 = 54321H

∴ CL ← [54321H]

## d. Register based indirect addressing mode

* In this mode, the effective address of the memory may be taken directly from one of the base registers or index registers specified by the instruction. If a register is SI, DI, and BX then DS is by default segment register.
* If BP is used, then SS is by default segment register.
* Example: MOV CX, [BX]
* This instruction moves a word from the address pointed by BX and BX + 1 in the data segment into CL and CH respectively. CL ← DS: [BX] and CH ← DS: [BX + 1]
* A physical address can be calculated as DS \* 10H + BX.

## e. Register relative addressing mode

* + In this mode, the operand address is calculated using one of the base registers and an 8 bit or a 16-bit displacement.
  + Example: MOV CL, [BX + 04H]
  + This instruction moves a byte from the address pointed by BX + 4 in the data segment to CL. CL ← DS: [BX + 04H]
  + A physical address can be calculated as DS \* 10H + BX + 4H.

## f. Base indexed addressing mode

* + Here, the operand address is calculated as a base register plus an index register.
  + Example: MOV CL, [BX + SI]
  + This instruction moves a byte from the address pointed by BX + SI in the data segment to CL. CL ← DS: [BX + SI]
  + A physical address can be calculated as DS \* 10H + BX + SI.

## g. Relative based indexed addressing mode

* In this mode, the address of the operand is calculated as the sum of base register, index register, and 8 bit or 16-bit displacement.
* Example: MOV CL, [BX + DI + 20]
* This instruction moves a byte from the address pointed by BX + DI + 20H in the data segment to CL. CL ← DS: [BX + DI + 20H]
* Physical address can be calculated as DS \* 10H + BX + DI + 20H.

## h. Implied addressing mode

* + In this mode, the operands are implied and are hence not specified in the instruction.
  + Example: STC
  + This sets the carry flag.

1. **Write a code for finding the value of 6!**

Ans:- Input: 06H Output: 02DOH

As In decimal: 6\*5\*4\*3\*2\*1= 720 In hexadecimal: 720= 02DOH

MOV CX, [0500]

MOV AX, 0001

MOV DX, 0000 MUL CX LOOP 040A

MOV [0600], AX

MOV [0601], DX HLT

# Conclusion :

In this experiment, we can learn how to write a code and execute it with microprocessor 8086. Microprocessor follows a sequence fetch, decode, and then it executes. We can see how this 8086 chip works like exchange value, addition, and subtraction. We have done some instructions, loops, and arrays from this laboratory work. We simulated and implemented it with microprocessor 8086.

For this experiment, we use a simulation software named “emu8086”. When we did the simulations we were not facing any problem at first. In loop problem when we try to simulate, we have got some error. Then we checked the assembly language and find that we were put the wrong syntax and then we make that syntax correct and get the right value. We make sure that we understand each line of code that we executed in the lab task and using some function to solve the problem.

# References:

1. “Microprocessors and Micro-Computer based System Design”, Second edition – by Dr. M. Rafiquzzaman
2. EMU8086 Manual
3. https://www.geeksforgeeks.org/addressing-modes-8086-microprocessor/