

# American International University-Bangladesh

Dept. of Computer Science and Engineering

Course Name: MICROPROCESSOR AND EMBEDDED SYSTEMS

Section: M Session: Spring 21-22

Course Instructor: Dr. Ferdous Jahan Shaun

Submission Date: 03-02-2022

# Title: Introduction to Microprocessor 8086, 8086 instructions and programming with 8086.

**Group Members**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Mohammad Sakib Islam Mazumder | 19-39290-1 |
| Munira Zebin | 19-41014-2 |
| Samir Faisal | 19-41037-2 |
| Nasim Reza Hridoy | 19-41045-2 |
| Shafiul Islam Mazumder | 19-41370-3 |
| Arpita Saha | 19-41363-3 |
| Puravi Debnath Nitu | 19-41371-3 |
| S. M. Rizwanul Hassan | 19-41386-3 |

Contents

[Title: Introduction to Microprocessor 8086, 8086 instructions and programming with 8086. 1](#_Toc94728651)

[1. Abstract 3](#_Toc94728652)

[2. Objectives 3](#_Toc94728653)

[3. Results 3](#_Toc94728654)

[(1) Simulation Environment 3](#_Toc94728655)

[(2) Simulation Results 5](#_Toc94728657)

[(3) Discussion 8](#_Toc94728658)

[4. Lab Tasks 8](#_Toc94728659)

[5. Question for report writing 11](#_Toc94728660)

[6. Conclusion 13](#_Toc94728661)

[7. References 13](#_Toc94728662)

# 

# Abstract

In this experiment we get to learn about the basic architectures of MDA 8086 microprocessor and it’s working principles. We need to familiarize with the emulator by using arithmetic program (Addition, Subtraction, Exchange, Series of Summation), test its different uses and get introduced to segmented memory technology used by microprocessor 8086.

# Objectives

* 1. To get familiar with the 8086 microprocessor and its component.
  2. To understand the functions of the component and their correlation.
  3. To perform reading and writing operands for memory and calculating the address of the memory operands.

# Results

## Simulation Environment

Emulator emu8086 software is used for assembly programming. Open the emu8086 from start menu.

## 

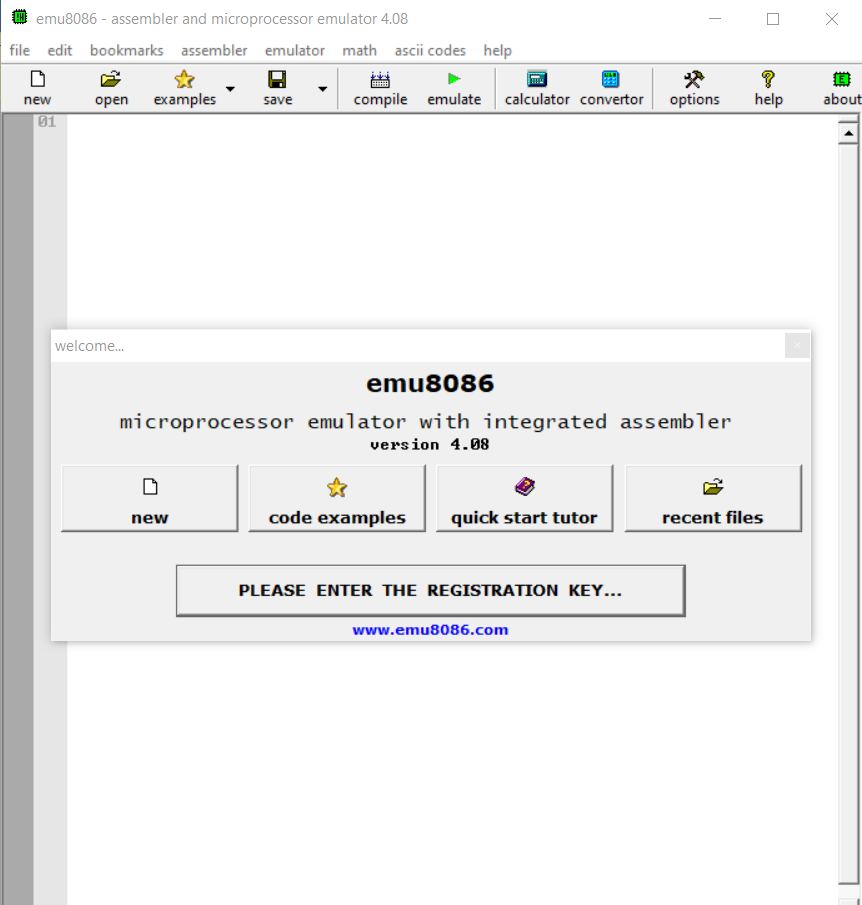


Figure 1: emu8086 Phase-1

Select 1st option and press ok button.

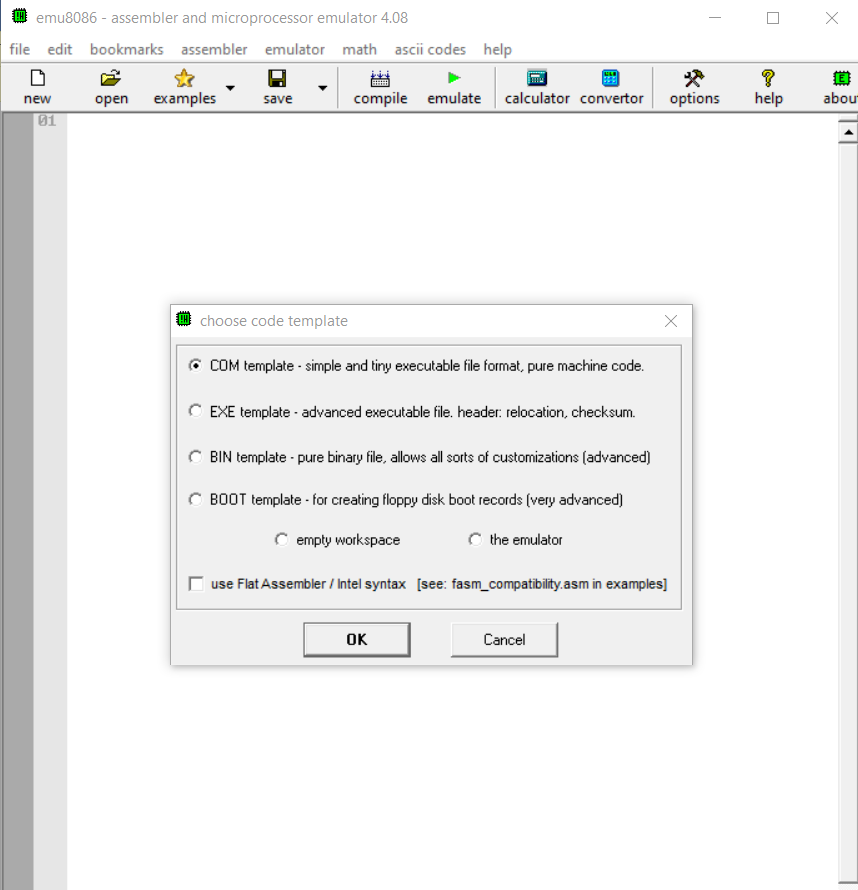


Figure 2: emu8089 choose code template

Write the code in this empty area.

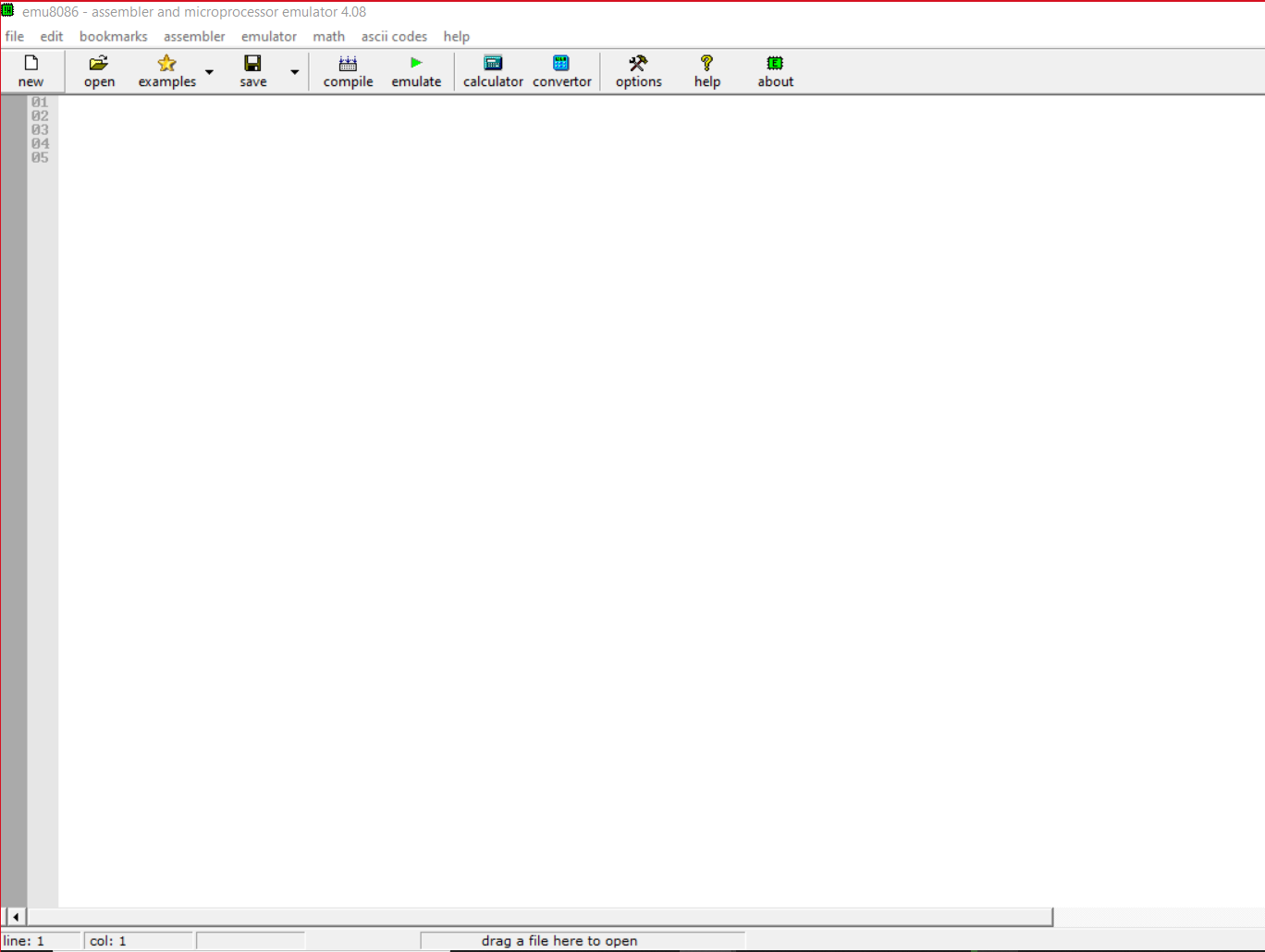


Figure 3: code write area

## Simulation Results

|  |  |
| --- | --- |
| Figure 4: Addition Phase-1 | *Figure 5: Addition Phase-2* |
| Figure 6: Addition Phase-3 | *Figure 7: Addition Phase-4* |

**Addition:** In this simple program-1, we see 1234h and 5630h have located in ax and bx register. Then the value of ax resister allocates to cx register. Then both ax register and cx register does the addition program and shows the output in phase-4 image.

|  |  |
| --- | --- |
| Figure 8: Subtraction Phase-1 | Figure 9: Subtraction Phase-2 |
| Figure 10: Subtraction Phase-3 | Figure 11: Subtraction Phase-4 |

**Subtraction:** In this simple program-2, we see 5634h and 1232h have located in bx and cx register. Then both bx and cx register does the subtraction operation and stores the result in bx register. Another subtraction program is done between al and dh which is shown in Phase-4 picture and the result is stored in al register.

|  |  |
| --- | --- |
| Figure 12: Summation of a series Phase-1 | Figure 13: Summation of a series Phase-2 |
| Figure 14: Summation of a series Phase-3 | Figure 15: Summation of a series Phase-4 |

**Summation of a series:** In this simple program-3, we see the value 9 has located in cx register. An addition program has occurred in this register from 9 to 0. From 9, the operation is decreased to 0 in each circular which result has been stored in bx register. The result value is 2D.

|  |  |
| --- | --- |
| Figure 16: Exchange Phase-1 | Figure 17: Exchange Phase-2 |

**Exchange:** In this simple program-4, we see 5678h and 4567h are located in bx and cx register. Then the value of ax register allocates to cx register. Then both bx register and cx register does the exchange program and changed their position to one another. From the output of phase-4, it can be said that that cx register’s new value is 5678h and bx resister’s new value is 4567.

## Discussion

The “EMU8086” is very new to all of our group member. For this reason, we are not accustomed to its interface and usability. So, we require further practice and research.

Then we discovered that the use of Comma (,) is essential. If we missed it in our code, we received syntax error. Example: In summation of a series code, if we did not put colon (:) after the “Start” loop scope then we found a syntax error and if alignment position did not place properly in “Start” scope then the output became different and erroneous.

# 4. Lab Tasks

**a) Implement all the example codes given above in emulator EMU8086 and take note of all general register values.**

=> All the example codes of lab report are provided in ‘Simulation Results’ section.

**(b) Write the assembly language program for DX= AX + BX - CX. Show the result on emulator screen of DX.**

=> Two operations have occurred in this program. One is Addition and another one is Subtraction. Firstly, AX and BX register do Addition operation and stored the result in AX register. Then AX register and CX register does subtraction operation and stores the result in AX register.

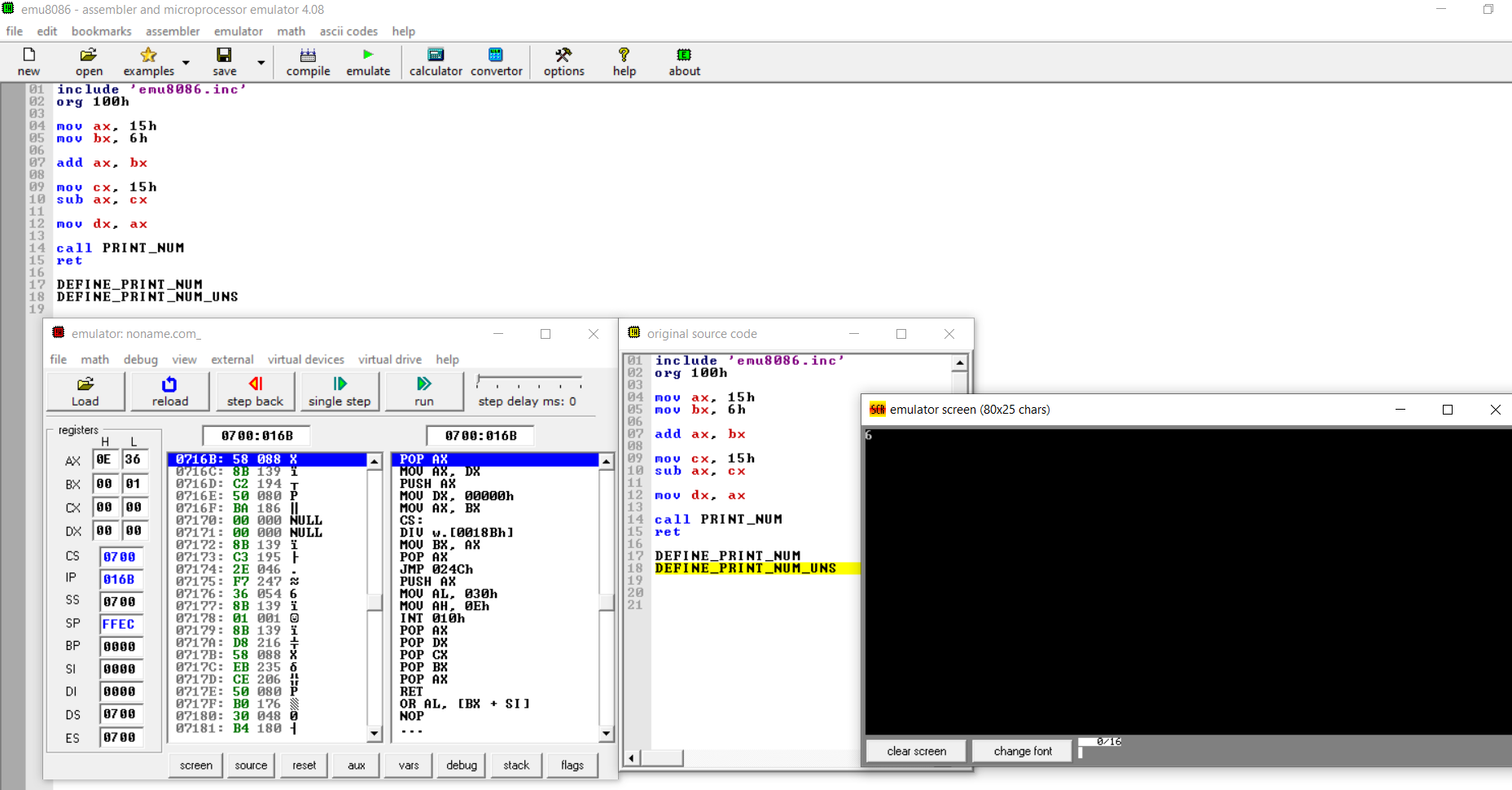


Figure 18: Show dx value in emulator screen

**(c) Write a program which display two characters at column#12 and row#7 at emulator screen.**

=> We have to put two character in column#12 and row #7. For this reason, we put 'A’,'B' character in this emulator. To complete this operation, we added a built-in method named "emu8086.inc".

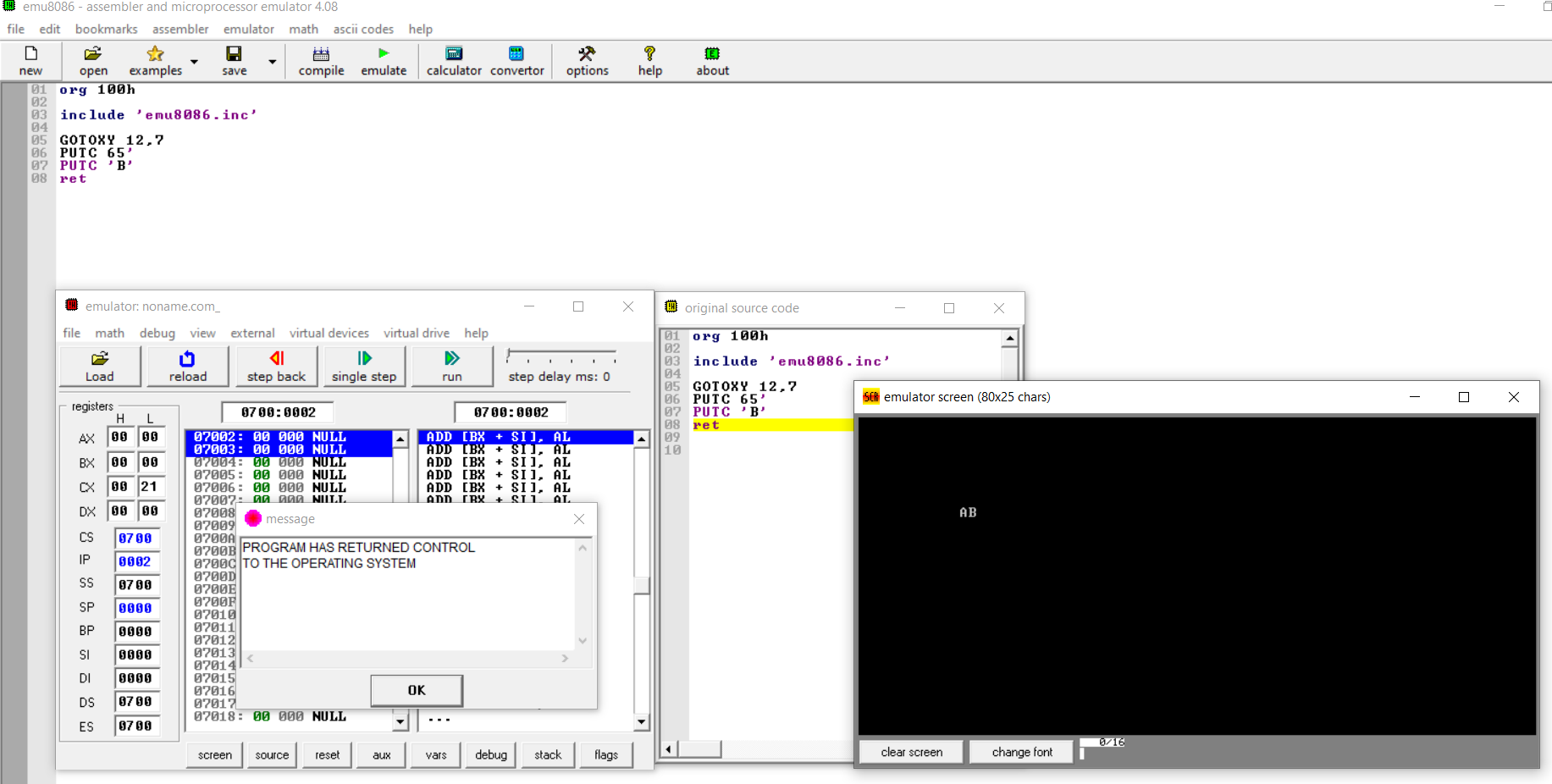


Figure 19: display two characters

**(d) Write the assembly code for the following sequence 1+3+5+7…. +N. Where N = 5 using loop.**

=> In this program, an additional program has occurred in this register from 1 to N. From 1, the operation increases up to N in each circular whose result is stored in ax register.

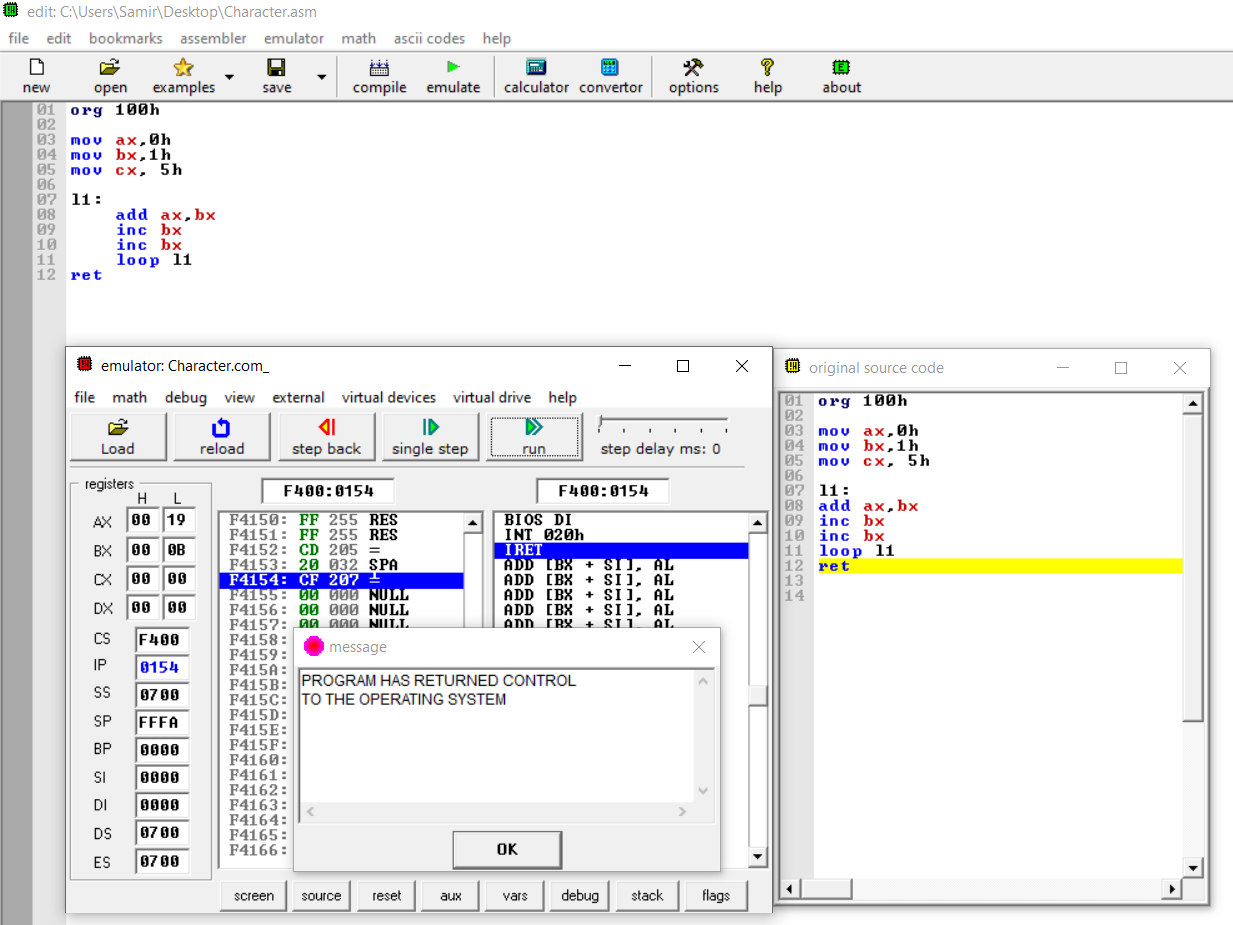


Figure 20: sequence using loop

# 5. Questions for report writing

**(a) Include all codes‟ list file printout following lab report writing template mentioned in appendix A.**

=> All the codes are included in ‘Lab Task’ section.

**(b) What is the advantage of having overlapping segments in 8086 memory system?**

=> Segments may overlap, the segment: offset form of an address is not unique, that is the same physical address can be represented in different segment: offset combinations. There are several advantages of working with the segmented memory. First of all, after initializing the 16 bits segment registers, the 8086 has to deal with only 16-bit effective addresses. That is 8086 has to store and manipulate only 16-bit address components as both segment and offset are 16 bits

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

=>

Here, physical address 1256Ah

Local address for segment 1256h = segment: Offset

= 1256h: {1256Ah-(1256h\*10h)}

= 1256h: 000Ah

Local address for segment 1240h = segment: Offset

= 1240h: {1256Ah-(1240h\*10h)}

= 1240h: 016Ah

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

=> An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere. There are 5 different addressing mode in 8086 microprocessors.

1. Direct addressing mode: The addressing mode in which the effective address of the memory location is written directly in the instruction. Example: MOV AX, [1592H]

2. Register indirect addressing mode: This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI. Example: MOV AX, [BX]

3. Based addressing mode: In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement. Example: ADD CL, [BX+08]

4. Indexed addressing mode: In this addressing mode, the operands’ offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements. Example: MOV BX, [SI+16]

5. Based-index addressing mode: In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register. Example: ADD CX, [AX+SI]

**(e) Write a code for finding the value of 6!**

=>

Input: 06H

Output: 02D0H

As In decimal: 6\*5\*4\*3\*2\*1= 720 In hexadecimal: 02D0H

mov cx, [0500]

mov ax, 0001

mov dx, 0000

MUL cx

Loop 040A

mov [0600], ax

mov [0601], dx

hlt

# 6. Conclusion

In this experiment, we learned how to implement and execute a code in “EMU8086” emulator. We learned some operations like data transfer instruction, logical instruction, arithmetic instruction and branching instruction etc. In this experiment, we simulated and implemented exchange value, addition and subtraction operation.

# 7. References

1. “Microprocessors and Micro-Computer based System Design”, Second edition – by Dr. M. Rafiquzzaman.

2. EMU8086 Manual.