**AMEIRCAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)**

**FACULTY OF ENGINEERING**

**MICROPROCESSOR AND EMBEDDED SYSTEMS [F]**

**Spring 2021-2022**

Lab report on

# Experiment 01: Introduction to Microprocessor 8086, 8086 instructions and programming with 8086

Supervised by

DR. NADIA ANAM (ID: 1305-1442-2)

Assistant Professor, EEE

## Group – 05

Submitted by:



| | Name | ID | Department |
|---|---|---|---|
| 01. | AKTER, ALAM MORIUM | 19-39957-1 | CSE |
| 02. | MOBARAK, BHUIYAN SAAD BIN | 19-41059-2 | EEE |
| 03. | NABIL, MD. JOBAIR AHMAD | 18-38837-1 | CSE |
| 04. | SAKIB, A.B.M. NAZMUS | 19-41582-3 | EEE |
| 05. | BHUIYAN, FAHIM MAHMUD | 20-42970-1 | CSE |



Date of submission: FEBRUARY 9, 2022

## Title:

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

## Introduction:

A microprocessor is a programmable, clock-driven, register-based electrical device that reads binary instructions from memory, accepts binary data as input, processes data according to those instructions, and outputs the results.

The experiment's major goals are-

• To learn about the internal design and structure of the 8086 microprocessor, as well as how it executes numerous instructions.

• To become acquainted with the emu8086 emulator.

• To understand how resisters exchange values, add and subtract, and so on.

• To learn how to use 8086 instructions to construct structured, intelligible assembly programs.

## Theory and Methodology:

**The 8086 Microprocessor;**

The Intel 8086 is a 16-bit microprocessor chip that was released between early 1976 and mid-1978. The 8086 became the foundation for Intel's future x86 architecture.

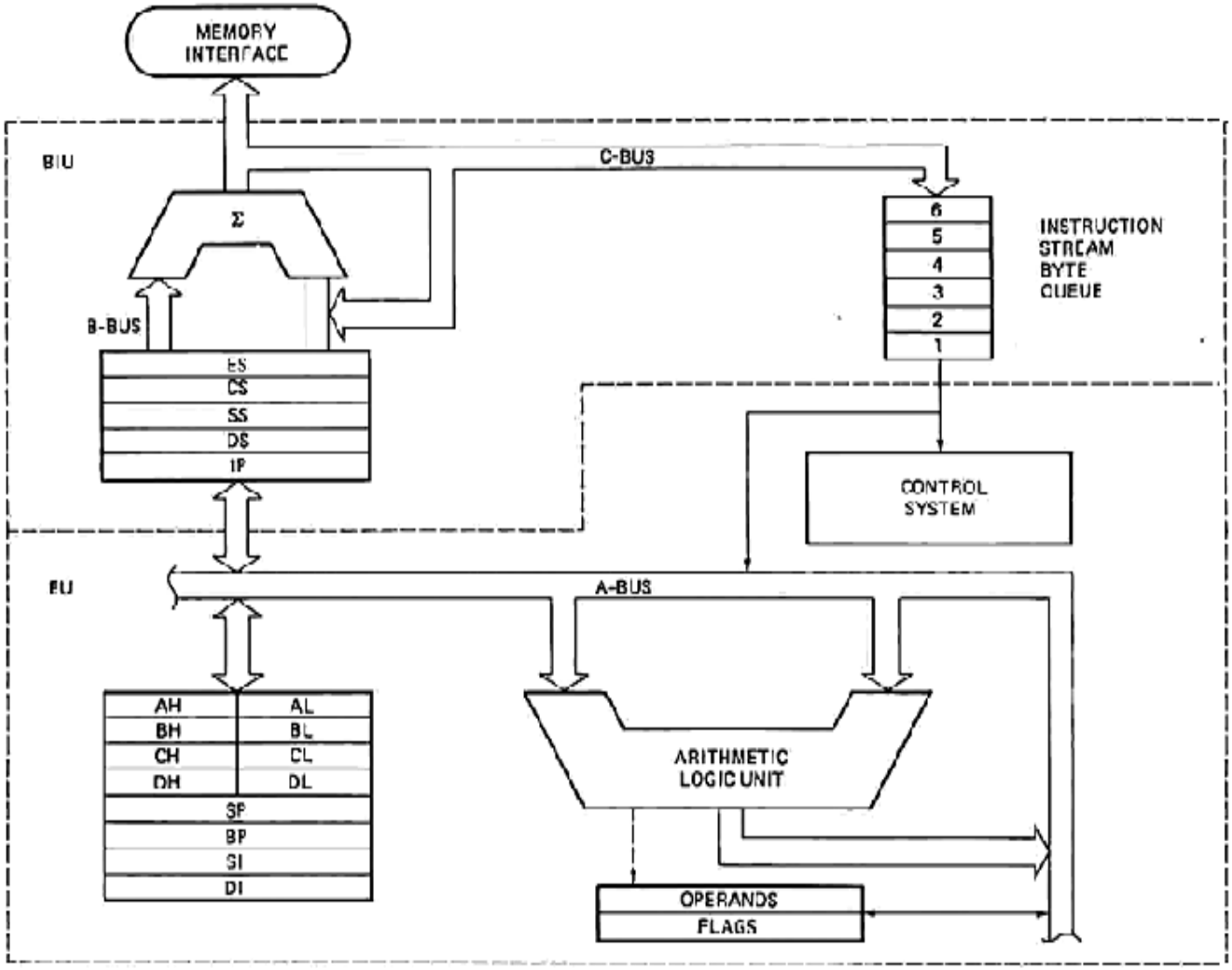



Fig 1: Intel 8086 internal architecture

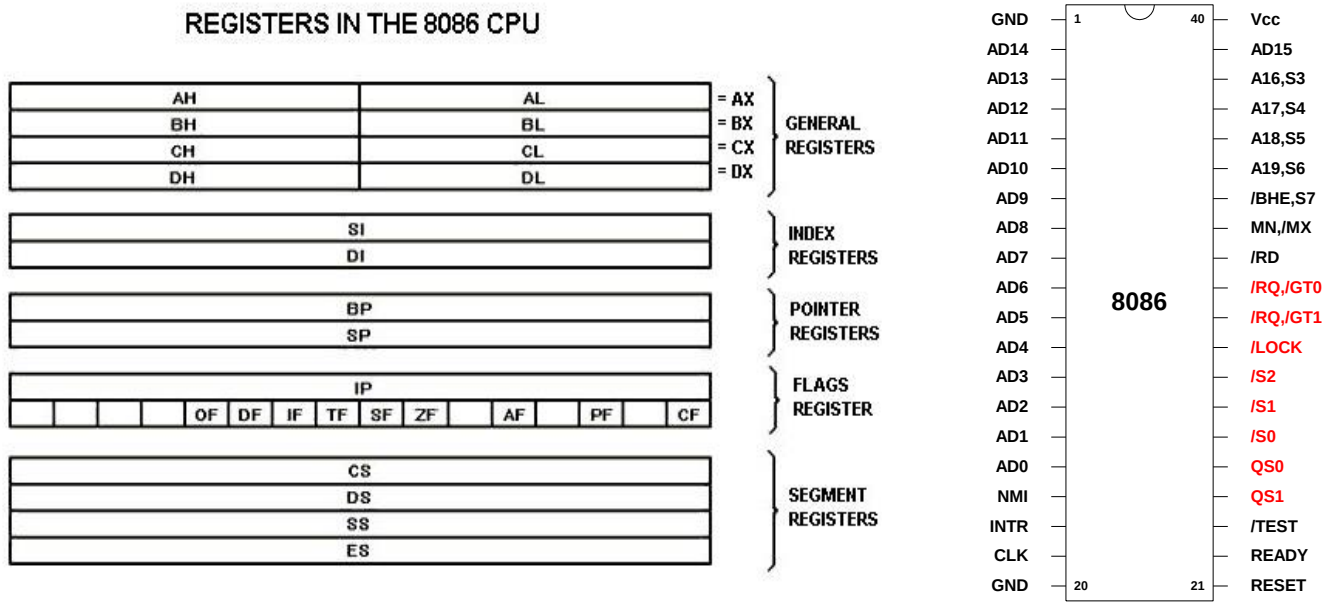



Fig 2: Internal Diagram, Registers and PIN diagram of the 8086 microprocessor

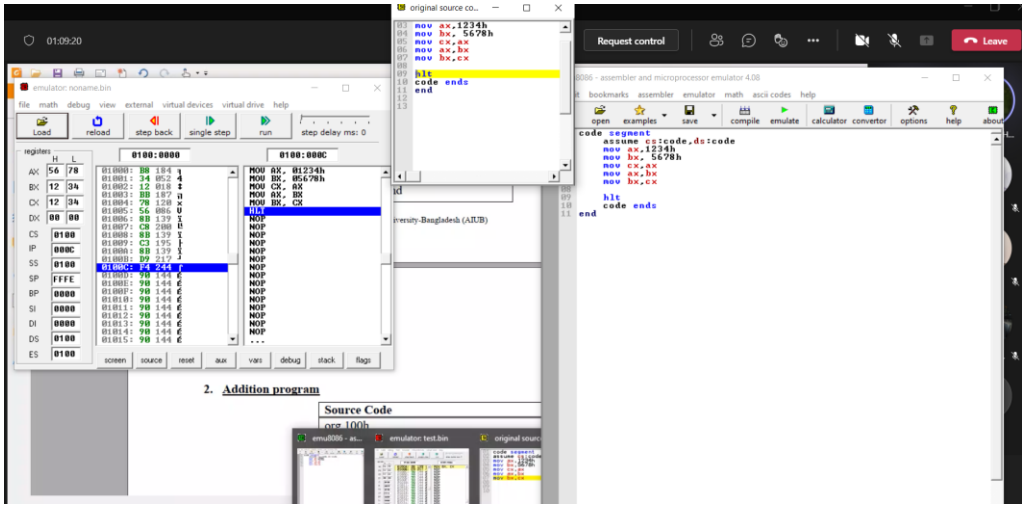
**8086 Segmented Memory;**

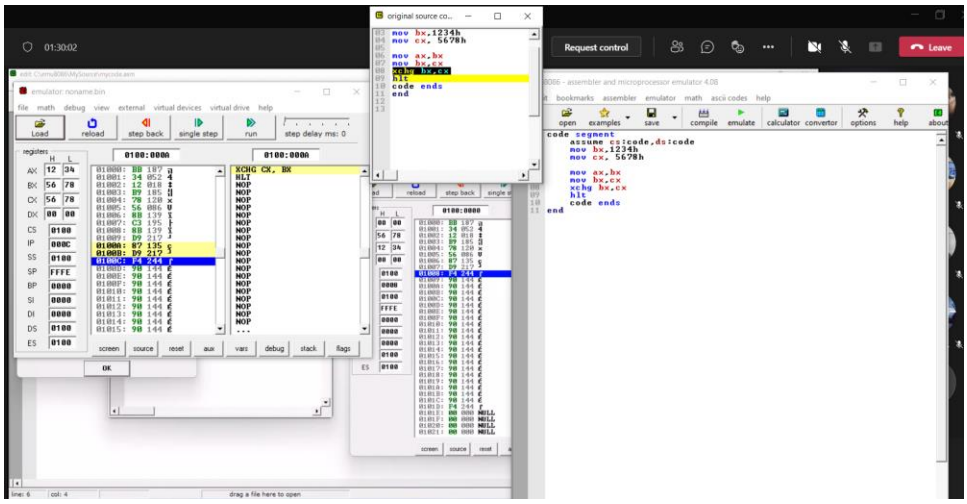
The architecture of 8086 is divided into two units and these are Bus Interface Unit (BIU) and Execution Unit (EU). BIU fetch the instructions from memory, calculate physical address and manage the queue. CS, DS, SS, ES are special registers in the BIU that are used to calculate physical addresses and the corresponding offset segments are SI, DI, SP,BP,IP provided by these registers. In memory, address registers store the address of instructions and data. The CPU uses these values to access memory locations. Memory regions in 8086 are given a 20-bit physical address. The physical addresses are represented as: 00000h 00001h 00002h …… …… FFFFFh.

The use of a 20-bit address in a 16-bit CPU results in segmented memory. The addresses are too large for a 16-bit register or memory word to hold. The 8086 overcomes this limitation by dividing its memory into parts.

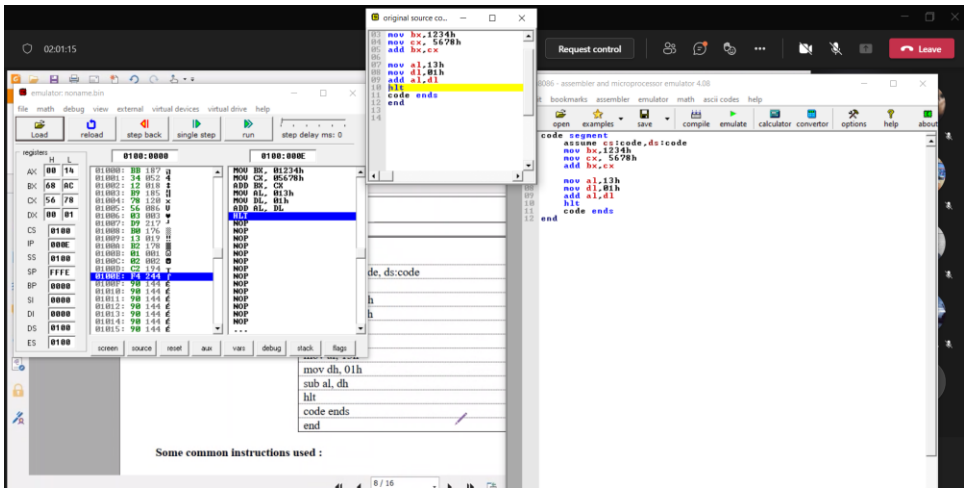
## Sample Programs:

1. Exchange Program-

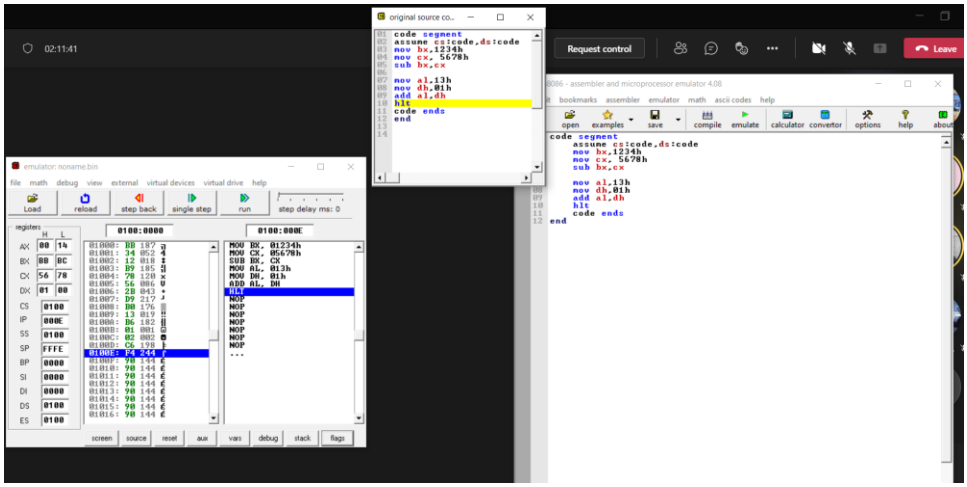




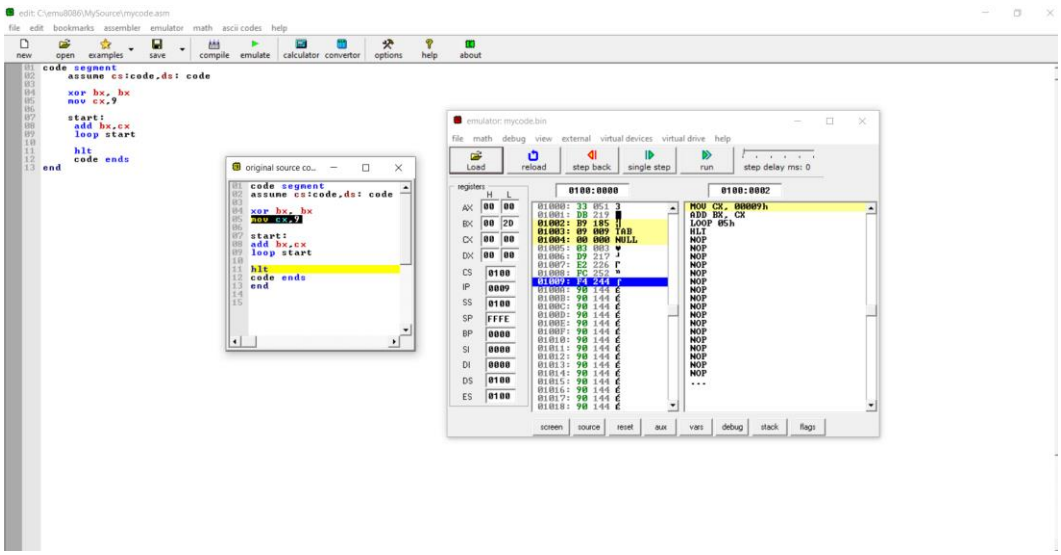
## 2. Addition Program-



## 3. Subtraction Program;

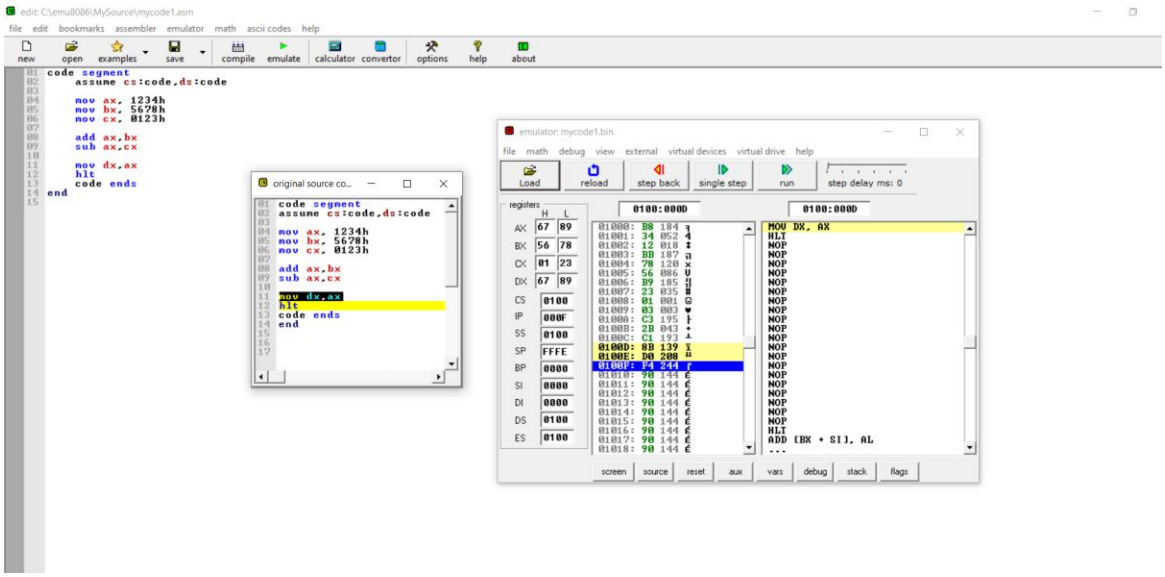



## 4. Summation of a series Program;

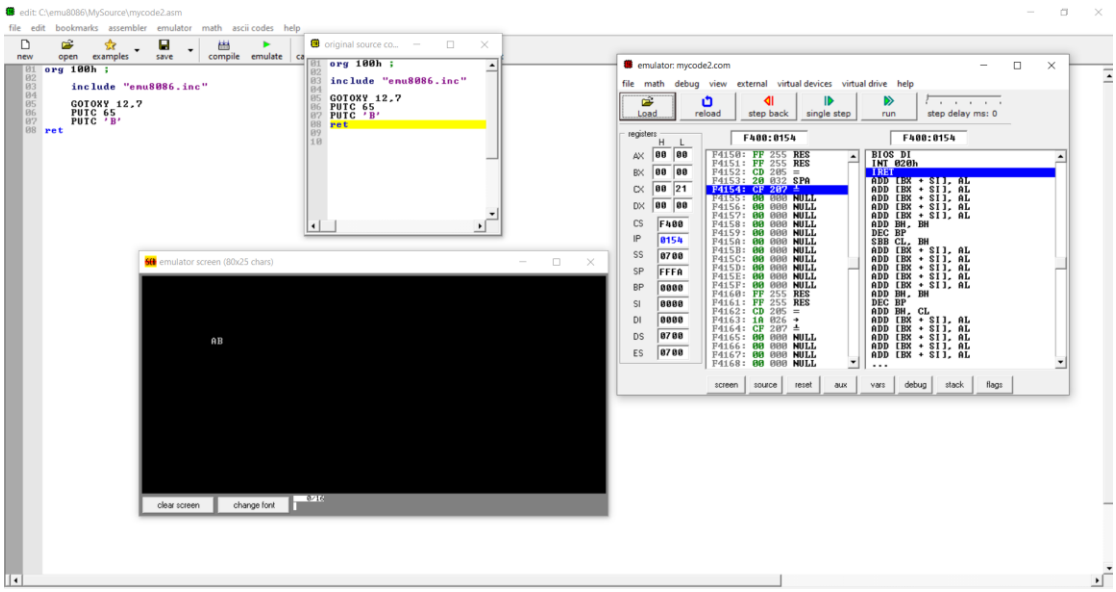


## Lab Task:

2. Write the assembly language program for DX= AX + BX – CX Show the result on emulator screen of DX.



3. Write a program which display two characters at column#12 and row#7 at emulator screen.



## Questions for Report writing:

### 2. What is the advantage of having overlapping segments in 8086 memory system?

A segment begins at a certain location and can have a maximum size of 64 kilobytes. However, if another segment begins at the same 64kilobyte place as the first, the two are considered to be Overlapping Segments. The advantage of having overlapping segments in 8086 memory system are following:-

- Larger addresses can be partitioned into overlapping parts, allowing them to fit inside 16-bit registers.
- Memory segmentation allows a user to deal with registers that are only 16 bits long.
- Only two bytes must be empty for the following instructions to be sent from the memory location.
- Allow the program code, data, and stack portions of the program to be stored in different memory locations.

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

The different ways in which a source operand is denoted in instruction are known as addressing modes. There are eight different addressing modes in 8086 programming; -

• **Immediate addressing mode:** The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

Example: MOV CX, 4929 H, ADD AX, 2387 H, MOV AL, FFH

• **Register addressing mode:** It means that the register is the source of an operand for an instruction.

Example: MOV CX, AX; (copies the contents of the 16-bit AX register into the 16-bit CX register), ADD BX, AX.

• **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], MOV AL [0300H]

• **Register indirect addressing mode:** This addressing mode allows data to be addresses at any memory location through an offset address held in any of the following register: BP, BX, DI and SI.

Example: MOV AX, [BX]; (Suppose the register BX contains 4895H, then the contents 4895H are moved to AX), ADD CX, {BX}

• **Based addressing mode:** In this addressing mode, the offset address of the operand is given by the sum of the contents of the BX/BP registers and 8-bit/16-bit displacement.

Example: MOV DX, [BX+04], ADD CL, [BX+08]

• **Indexed addressing mode:** In this addressing mode, the operand's offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example: MOV BX, [SI+16] ADD AL, [DI+16]

• **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], MOV AX, [AX+DI]

• **Based indexed with displacement mode:** In this addressing mode, the operands offset is computed by adding the base register contents. An index registers contents and 8 or 16-bit displacement.

Example: MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]

## Conclusion:

In this experiment we familiarized ourselves with the 8086 emulator and we performed different sample programs to understand the functionalities of the emu8086. In addition, some lab tasks were performed to conclude certain results. The commands that have been used to do all these programs are mov, xchg, add, sub and loop factorial, GOTOXY and so on from the above programs. Therefore, these programs changed the value of the registers and indicated the location in memory. So, overall, the experiment went quite well and smoothly.

## References:


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

2. EMU8086 Manual.

3. https://tutorialspoint.dev/computer-science/operating-systems/memory-segmentation-8086-microprocessor

4. Microprocessor - 8086 Addressing Modes (tutorialspoint.com)