

LAB MANUAL
COMPUTER ORGANIZATION
AND
ARCHITECTURE
ETCS 254



**Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085
(Affiliated to Guru Gobind Singh Indraprastha University,
New Delhi)**

INDEX OF THE CONTENTS

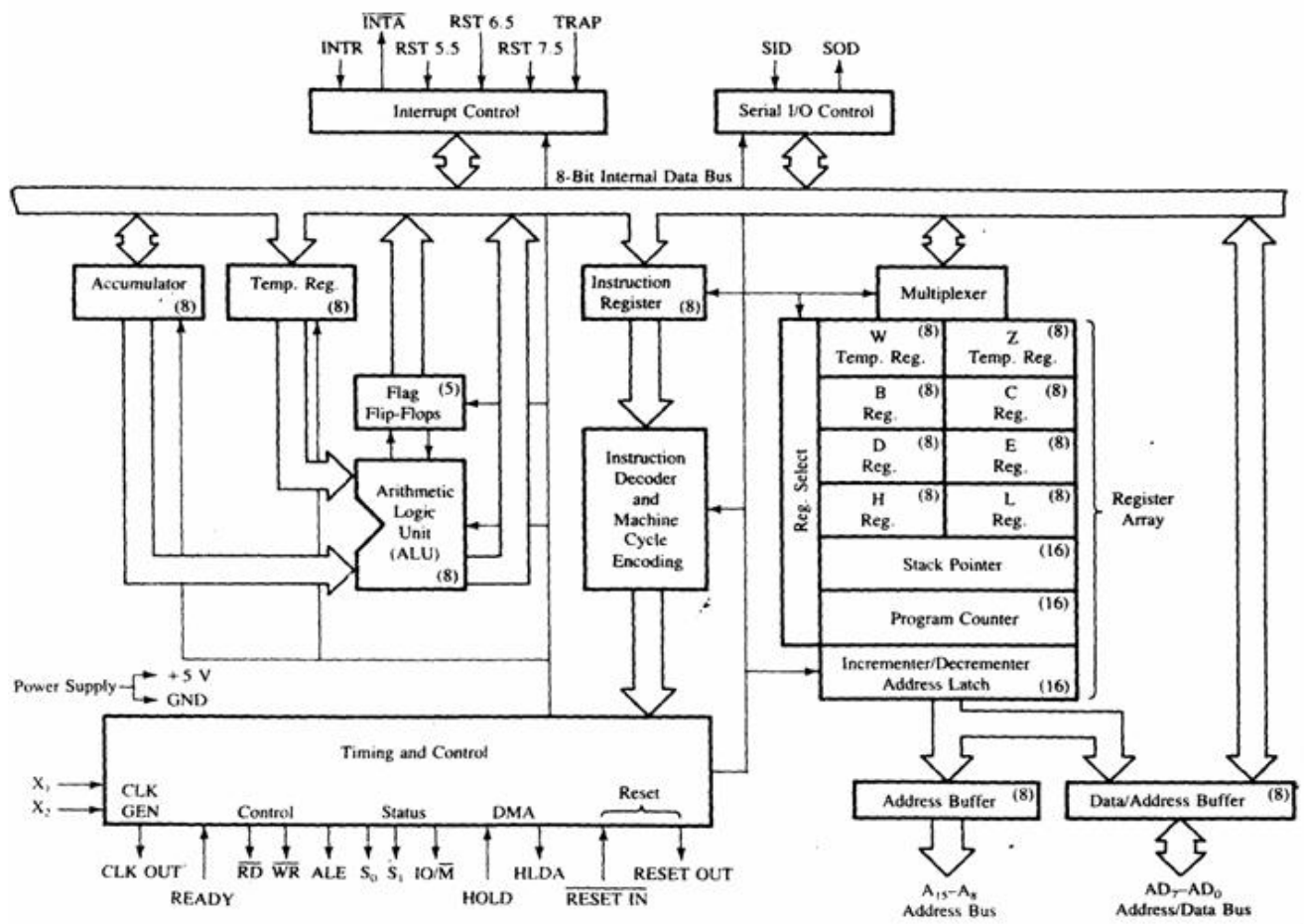
- 1. Introduction to the lab manual**
- 2. Lab requirements (details of H/W & S/W to be used)**
- 3. List of experiments**
- 4. Sample codes**
- 5. Format of lab record to be prepared by the students.**
- 6. Marking scheme for the practical exam**
- 7. Details of the each section of the lab along with the examples, exercises & expected viva questions.**

1. INTRODUCTION TO THE LAB

Computer Organization and architecture lab consist of performing various experiments in GNU Sim (A simulator for 8085 microprocessor).

Before doing the coding on the simulator it's necessary to study the complete architecture of 8085 microprocessor along with its instruction set.

1.1 ARCHITECTURE OF MICROPROCESSOR 8085



List of registers used in 8085 to perform various operations:

Accumulator:-It is a 8-bit register which is used to perform arithmetical and logical operation. It stores the output of any operation. It also works as registers for i/o accesses. It can be one of the operand in the instruction.

Temporary Register:-It is a 8-bit register which is used to hold the data on which the accumulator is computing operation. It is also called as operand register because it provides operands to ALU.

Registers:-These are general purposes registers. Microprocessor consists 6 general purpose registers of 8-bit each named as B,C,D,E,H and L. Generally these registers are not used for storing the data permanently. It carries the 8-bits data. These are used only during the execution of the instructions.

These registers can also be used to carry the 16 bits data by making the pair of 2 registers. The valid register pairs available are BC,DE HL. We cannot use other pairs except BC,DE and HL. These registers are programmed by user.

Flag Registers:-It consists of 5 flip flop which changes its status according to the result stored in an accumulator. It is also known as status registers. It is connected to the ALU.

There are five flip-flops in the flag register are as follows:

The bit position of the flip flop in flag register is:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S	Z		AC		P		CY

All of the three flip flop set and reset according to the stored result in the accumulator.

1.**Sign-** If D₇ of the result is 1 then sign flag is set otherwise reset. As we know that a number on the D₇ always decides the sign of the number.

if D₇ is 1: the number is negative.

if D₇ is 0: the number is positive.

2.**Zeros(Z)**-If the result stored in an accumulator is zero then this flip flop is set as 1 otherwise it is reset and also if the result of any arithmetic or logical operation is zero its set as 1(The result of this operation can be in any registers).

3.**Auxiliary carry(AC)**-If any carry goes from D₃ to D₄ in the output then it is set otherwise it is reset.

4.**Parity(P)**-If the no of 1's is even in the output stored in the accumulator then it is set otherwise it is reset for the odd.

5.**Carry(C)**-If the result stored in an accumulator generates a carry in its final output then it is set

otherwise it is reset.

Instruction registers(IR):-It is a 8-bit register. When an instruction is fetched from memory then it is stored in this register.

Description of other components of 8085 to:

Instruction Decoder: - Instruction decoder identifies the instructions. It takes the information from instruction register and decodes the instruction to be performed.

Program Counter:-It is a 16 bit register used as memory pointer. It stores the memory address of the next instruction to be executed. So we can say that this register is used to sequencing the program. Generally the memory have 16 bit addresses so that it has 16 bit memory. The program counter is set to 0000H.

Stack Pointer:-It is also a 16 bit register used as memory pointer. It points to the memory location called stack. Generally stack is a reserved portion of memory where information can be stores or taken back together.

Timing and Control Unit:-It provides timing and control signal to the microprocessor to perform the various operation. It has three control signals. It controls all external and internal circuits. It operates with reference to clock signal. It synchronizes all the data transfers.

There are three control signal:

- 1.ALE-Airthmetic Latch Enable, It provides control signal to synchronize the components of microprocessor.
- 2.RD- This is active low used for reading operation.
- 3.WR-This is active low used for writing operation.

There are three status signal used in microprocessor S0, S1 and IO/M. It changes its status according the provided input to these pins.

Serial Input Output Control-There are two pins in this unit. This unit is used for serial data communication.

Interrupt Unit-There are 6 interrupt pins in this unit. Generally an external hardware is connected to these pins. These pins provide interrupt signal sent by external hardware to microprocessor and microprocessor sends acknowledgement for receiving the interrupt signal. Generally INTA is used for acknowledgement.

1.2 INTRODUCTION TO GNU Simulator 8085

8085 simulator is software on which instructions are executed by writing the programs in assembly language.

GNUSim8085 is a 8085 microprocessor simulator with following features.

- A simple editor component with syntax highlighting.
- A keypad to input assembly language instructions with appropriate arguments.
- Easy view of register contents.
- Easy view of flag contents.
- Hexadecimal <--> Decimal converter.
- View of stack, memory and I/O contents.
- Support for breakpoints for programming debugging.
- Stepwise program execution.
- One click conversion of assembly program to opcode listing.
- Printing support (known not to work well on Windows).
- UI translated in various languages.

Writing a program in assembly language:-

Format of the instruction is as follows:-

Label	Operation	Operands	Comments
Its optional	Necessary	Necessary	Its optional

A basic assembly program consists of 4 parts.

1. Labels
2. Operations :- these operations can be specified as

Machine operations (mnemonics):- used to define operations in the form of opcode as mention in the instruction set of microprocessor 8085.

Pseudo operations (like preprocessor in C):- these are assembly directives.

3. Operands
4. Comments

In addition, you have **constants** in an assembly program. Unless otherwise specified, a constant which is always numeric is in decimal form. If appended with a character **h** it is assumed to be in hexadecimal form. If a hex constant starts with an alpha-char don't forget to include the

number 0 in the beginning, since that will help the assembler to differentiate between a label and a constant.

Labels:-

When given to any particular instruction/data in a program, takes the address of that instruction or data as its value. But it has different meaning when given to EQU directive. Then it takes the operand of EQU as its value. Labels must always be placed in the first column and must be followed by an instruction (no empty line). Labels must be followed by a : (colon), to differentiate it from other tokens.

Operations:-

As mentioned above the operations can be specified in two ways that are **mnemonics** and **pseudo operation**.

Pseudo operations can be defined by using following directives:-

There are only 3 directives currently available in our assembly language.

1. **DB** - define byte (8 bits)
2. **DS** - define size (no. of bytes)
3. **EQU** - like minimalistic **#define** in C

DB is used to define space for an array of values specified by comma separated list. And the label (if given to the beginning of **DB**) is assigned the address of the first data item.

DS is used to define the specified number of bytes to be assigned and initialize them to zero. To access each byte you can use the + or -operator along with label.

EQU behaves similar to **#define** in C. But it is simple. It can be used to give names only to numeric constants. Nesting of **EQU** is not allowed. You can use **EQU** only in operands for pseudo ops and mnemonics.

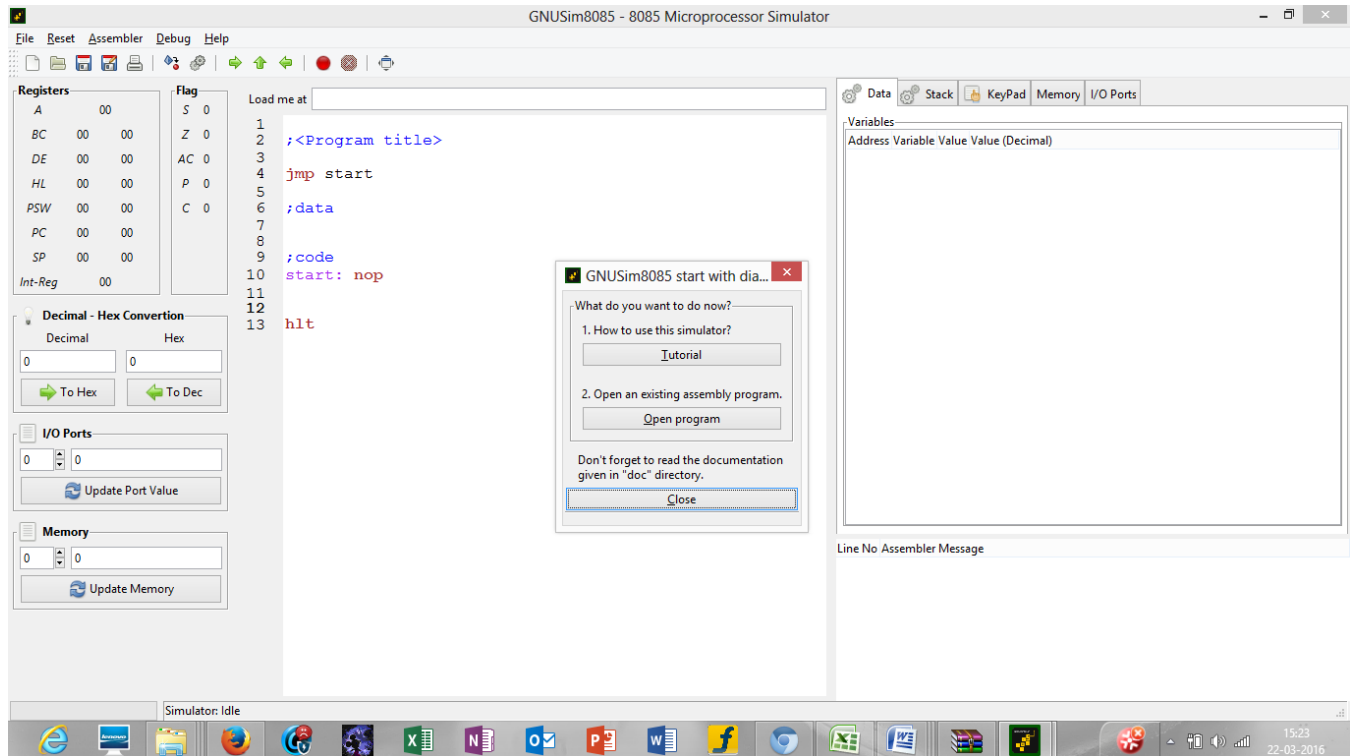
Operands:-

Operands are specified according to the user. The register set specified in the architecture of 8085 (A, B, C, D, H and L) are used to access and store data. These registers are specified as operand. In case of accessing data or storing data in the memory 'm' is specified as an operand and the address of this memory location is taken from the HL pair (data in HL pair).

INSTRUCTION SET OF MICROPROCESSOR 8085

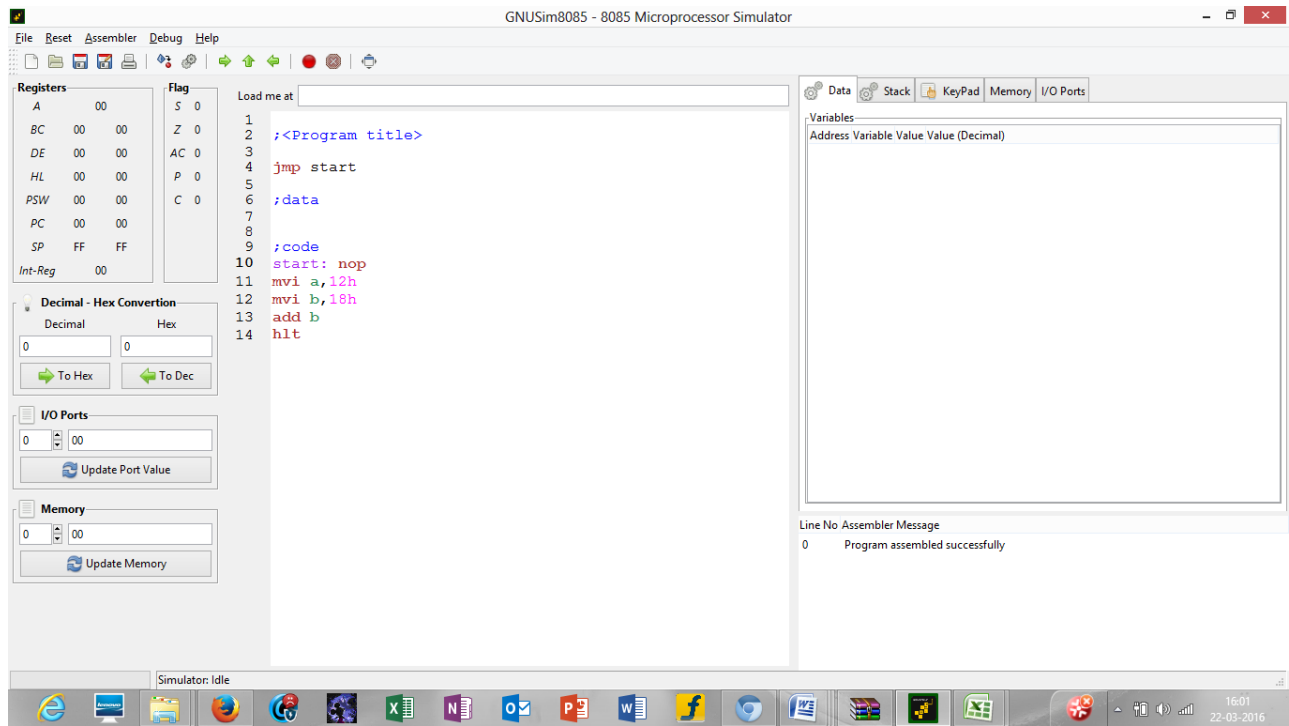
[See the appendix 1](#)

HOW TO START WITH GNU Simulator 8085



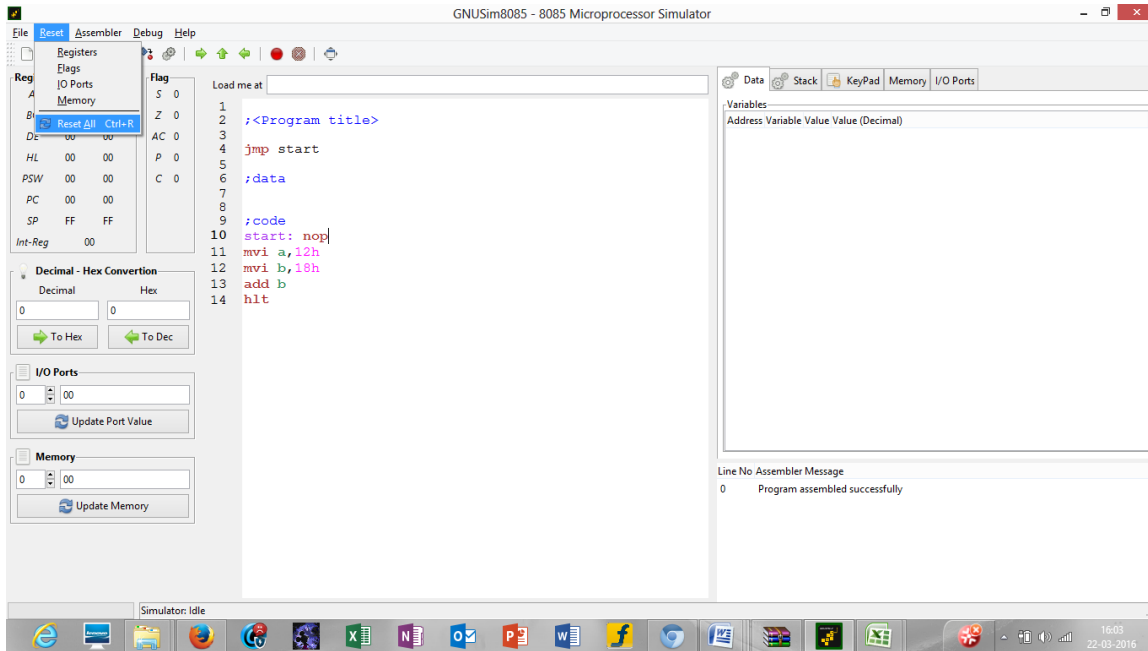
Picture 1

Step1: open GNU Sim 8085 above window will open. Now click on close button highlighted in the above screen shot.



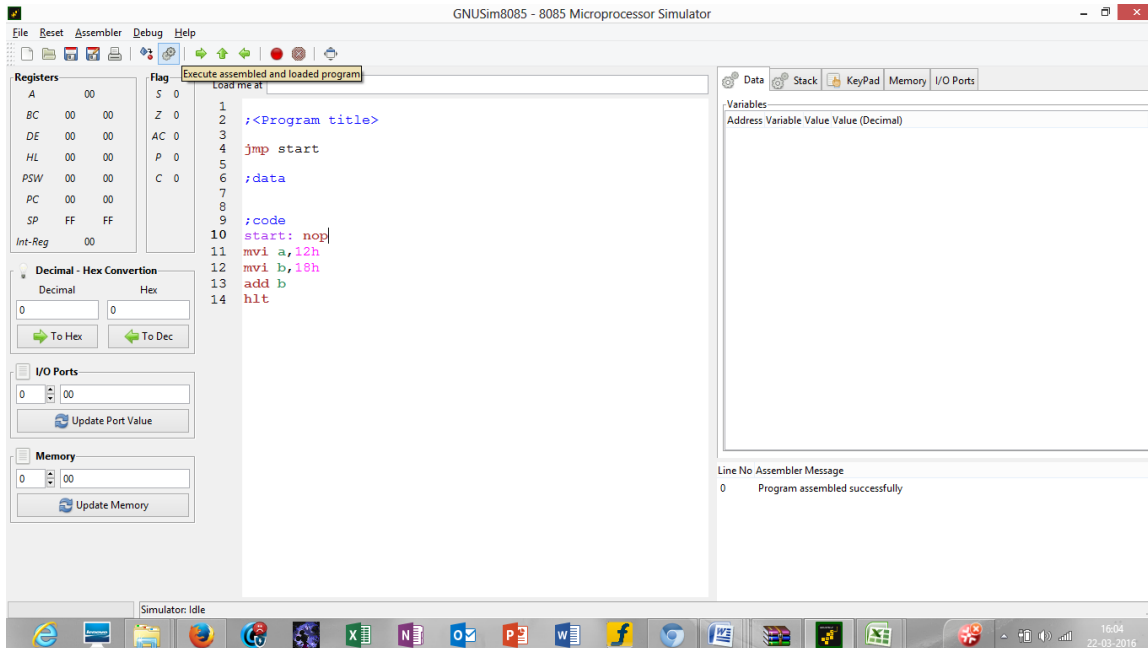
Picture 2

Step2: start writing the code after start: nop in load me at 10 that is at load me at 11.



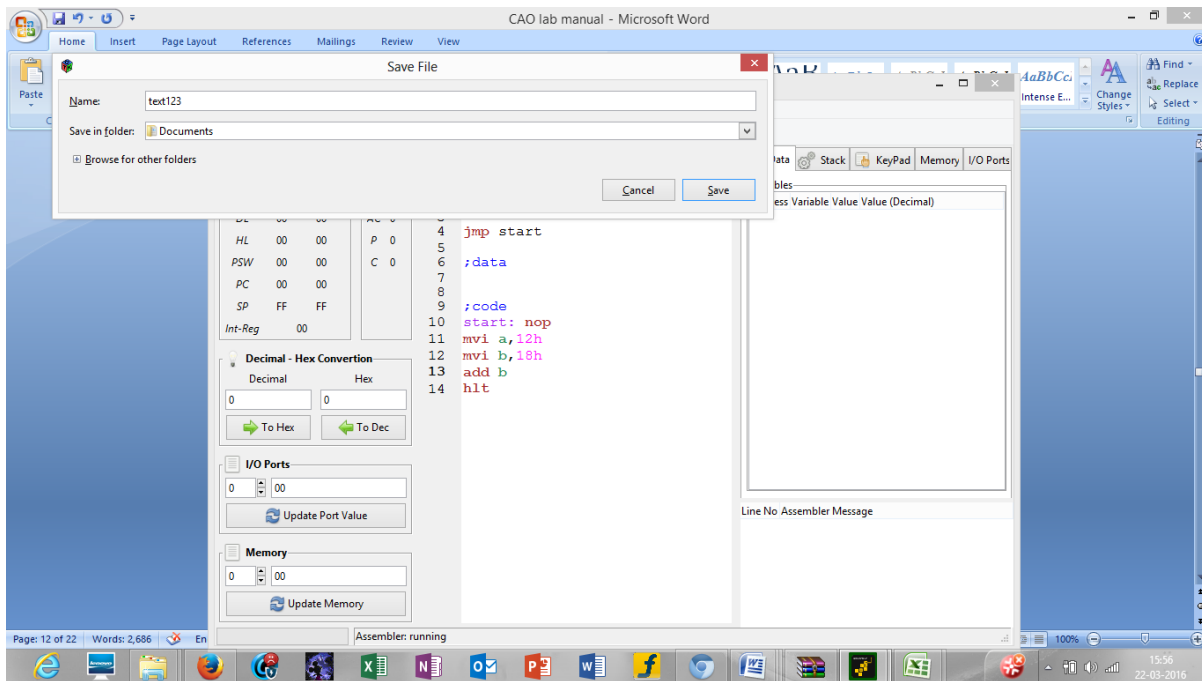
Picture 3

Step 3: click on reset and reset all the registers by clicking on reset all.



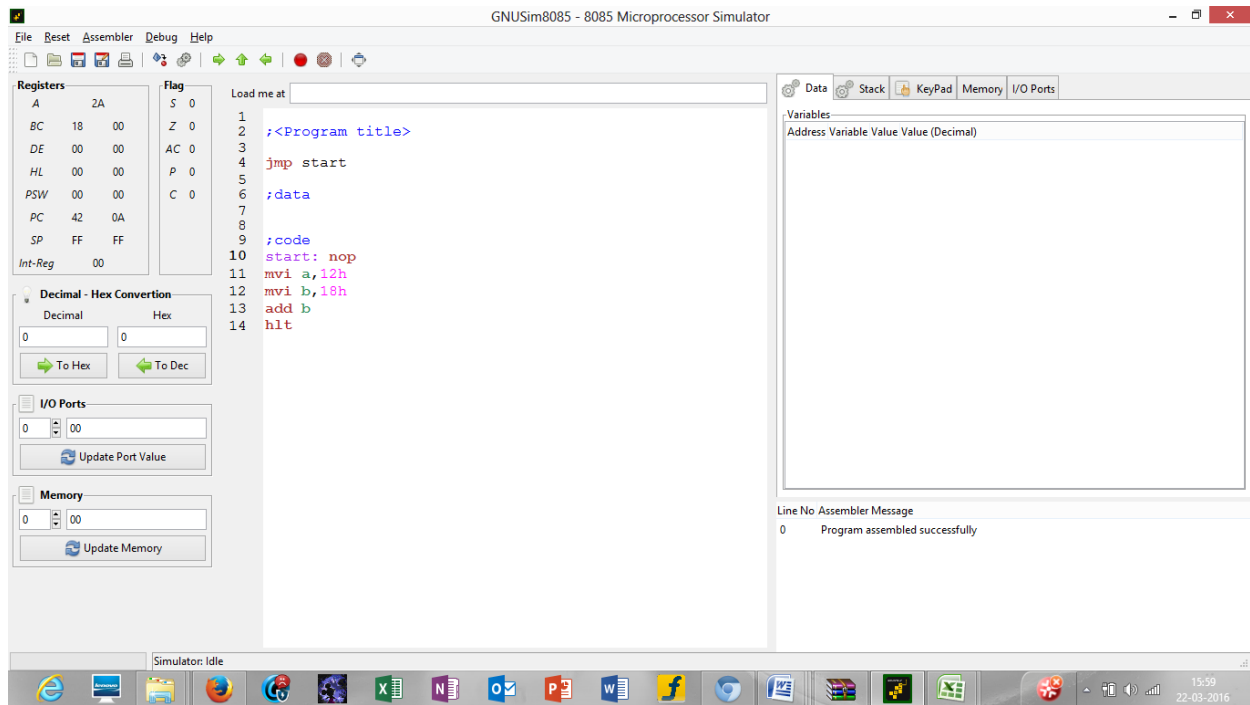
Picture 4

Step 4: click on the highlighted button to execute the code



Picture 5

Step 5: after you execute the code mention the name your program by writing the name in the name section as mentioned in the screen shot in picture 5 and the drive where you want to save it. After that click on save.



Picture 6

Step 6: after this you will see the result of the instructions in the respective registers as seen in the above picture 6.

2. Lab requirements (details of H/W & S/W to be used)

GNU Sim 8085 is an open source and is platform independent.

General requirements are:

Software requirements: GNU Sim 8085

Operating System: Windows 7

Hardware requirements: P-IV C2D 2.9 GHZ
320 GB HDD/2 GB RAM
Cabinet/1.44 FDD
LAN Card

3. LIST OF EXPERIMENTS

Paper Code: ETCS 254

P C

Paper: Computer Organization and Architecture

3 2

List of Experiments

1. Write the working of 8085 simulator GNUsim8085 and basic architecture of 8085 along with small introduction.
2. Study the complete instruction set of 8085 and write the instructions in the instruction set of 8085 along with examples.
3. Write an assembly language code in GNUsim8085 to implement data transfer instruction.
4. Write an assembly language code in GNUsim8085 to store numbers in reverse order in memory location.
5. Write an assembly language code in GNUsim8085 to implement arithmetic instruction.
6. Write an assembly language code in GNUsim8085 to add two numbers using lxi instruction.
7. Write an assembly language code in GNUsim8085 to add two 8 bit numbers stored in memory and also storing the carry.
8. Write an assembly language code in GNUsim8085 to find the factorial of a number.
9. Write an assembly language code in GNUsim8085 to implement logical instructions.

Write an assembly language code in GNUsim8085 to implement stack and branch instructions.

3. Sample code

Programs in gnusim8085

I. ADDITION OF TWO NUMBERS

```
lda var1
mov b,a
lda var2
add b
sta var3
hlt
var1: db 04h
var2: db 09h
var3: db 00h
```

II. To add n consecutive numbers

```
lxi h,var
mov c,m
mvi b,01h
mvi e,00h
mvi a,00h
back: add b
jnc skip
inr e
skip: inr b
dcr c
jnz back
sta result
mov a,e
sta carry
hlt
var: db 0Ah
result: db 00h
carry: db 00h
```

III. count the number of 1's

```
lxi h,var
mvi c,08h;counter
mov a,m
mvi b,00h;count number of 1's
back: rar
jnc skip
inr b
```

```
skip: dcr c
jnz back
mov a,b
sta result
hlt
var: db 19h
result: db 00h
```

IV. multiply two 8 bit numbers without shifting

```
lxi h,
var; multiplicand
mvi d,00h
mov e,m
inx h
mov c,m; multiplier as counter for repeated addition
mvi h,00h
mvi l,00h
back: dad d
dcr c
jnz back
shld result
hlt
var: db 08h
var2: db 07h
result: db 00h
result2: db 00h
```

V. addition of two numbers using lxi

```
lxi h,var1
mov a,m
inx h
mov b,m
sub b
inx h
mov m,a
hlt
var1: db 08h
var2: db 03h
var3: db 00h
```

VI. division of 8bit number

```
lhld var;dividend
lda var2;divisor
```



```

mov b,a
mvi c,08h
back: dad h
mov a,h
sub b
jc forward
mov h,a
inr l
forward: dcr c
jnz back
shld var3
hlt
var: db 0ch
var1: db 00h
var2: db 05h
var3: db 00h
var4: db 00h

```

VII. To find the smallest and largest number from the given series

```

lxi h,var
mov c,m ;counter
inx h
dcr c
mov b,m;for largest
mov d,m;for smallest
mov a,m
back: cmp b
jc ahead
mov b,a
ahead: cmp d
jnc ahead2
mov d,a
ahead2: inx h
mov a,m
dcr c
jnz back
inx h
mov m,d
inx h
mov m,b
hlt
var: db 05h
var1: db 02h
var2: db 02h
var3: db 07h
var4: db 0Ah
var5: db 0Ah
smallest: db 00h
largest: db 00h

```

VIII. FACTORIAL OF THE NUMBER

```
lxi sp,27ffh
lda var2
cpi 02h
jc last
mvi d,00h
mov e,a
dcr a
    mov c,a
call facto
xchg
shld var
jmp end
last: lxi h,0001h
end: shld var
hlt
facto: lxi h,0000h
mov b,c
back: dad d
dcr b
jnz back
xchg
dcr c
cnz facto
ret
var: db 00h
var2: db 03h ; input the number 3 here, donot give number more than 5
```

6. **FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS**

1. The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Font should be (Size 20", italics bold, Times New Roman)

Faculty name

Font should be (12", Times Roman)

Student name

Roll No.:

Semester:

Group:

Font should be (12", Times Roman)



Maharaja Agrasen Institute of Technology, PSP Area,
Sector – 22, Rohini, New Delhi – 110085

Font should be (18", Times Roman)

2. The second page in the record should be the index as displayed below.

COMPUTER ORGANIZATION AND ARCHITECTURE

PRACTICAL RECORD

PAPER CODE : ETCS - 254

Name of the student :

University Roll No. :

Branch :

Section/ Group :

PRACTICAL DETAILS

a) List of experiments

Exp. no	Experiment Name	Date of performance	Date of checking	Remarks	Marks (10)

3. Each practical which student is performing in the lab should have the following details in the respective sections:
 - a) AIM
 - b) Date
 - c) Procedure
 - d) Output
 - e) Tutorial questions

7. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

- Internal Practical Exam
- External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 40 marks is as follows

1.	Regularity:	25
	<ul style="list-style-type: none">• Performing program in each turn of the lab• Attendance of the lab• File	
2.	Viva Voice:	10
3.	Project:	5

NOTE: For the regularity, marks are awarded to the student out of 10 for each experiment performed in the lab and at the end the average marks are giving out of 25.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student:	20
2. Viva Voice:	15
3. Experiment performance:	15
4. File submitted:	10

NOTE:

- Internal marks + External marks = Total marks given to the students
(40 marks) (60 marks) (100 marks)
- Experiments given to perform can be from any section of the lab.