EXPERIMENT NO.1

Aim: To study the introduction to 8085 architecture and basic instruction set of 8085.

Part A: ARCHITECTURE OF 8085:

8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration −

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

It is used in washing machines, microwave ovens, mobile phones, etc.

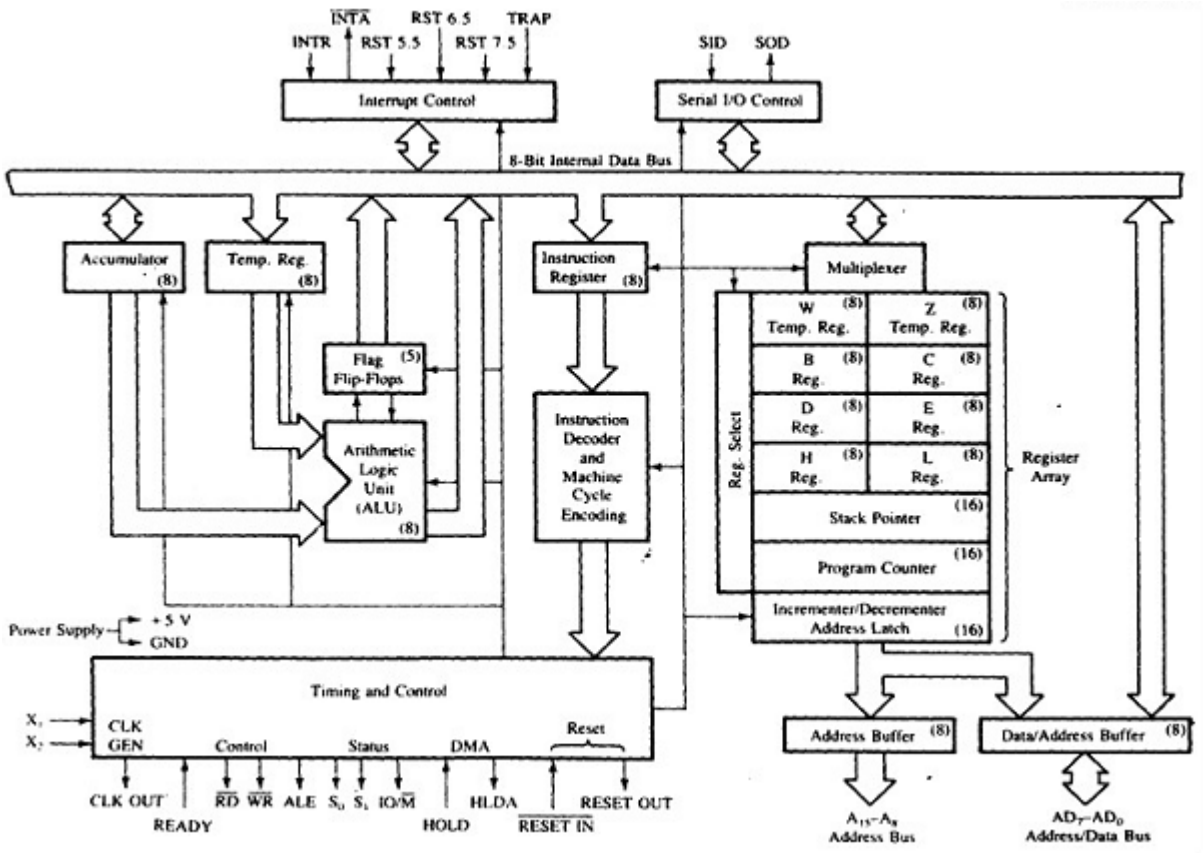


Fig. 1. 8085 consists of the following functional units

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops −

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits −

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA

• RESET Signals: RESET IN, RESET OUT

Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

Serial Input/output control

It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address buffer and address-data buffer

The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

Address bus and data bus

Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

The architecture of the 8085 microprocessor mainly includes the timing & control unit, Arithmetic and logic unit, decoder, instruction register, interrupt control, a register array, serial input/output control. The most important part of the microprocessor is the central processing unit.

Part B: INSTRUCTION SET OF 8085:

Instruction sets are instruction codes to perform some task. It is classified into five categories**:**

1) CONTROL INSTRUCTIONS:

Control Instructions are the machine code that are used by machine or in assembly language by user to command the processor act accordingly. These instructions are of various types. These are used in assembly language by user also. But in level language, user code is translated into machine code and thus instructions are passed to instruct the processor do the task.

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

2) LOGICAL INSTRUCTIONS:

Opcode	Operand	Meaning	Explanation

CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
CPI	8-bit data	Compare immediate with the accumulator	The second byte data is compared with the contents of the accumulator.
ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
XRA	R M	Exclusive OR register or memory with the accumulator	The contents of the accumulator are Exclusive OR with M the contents of the register or memory, and the result is placed in the accumulator.
XRI	8-bit data	Exclusive OR immediate with the accumulator	The contents of the accumulator are Exclusive OR with the 8-bit data and the result is placed in the accumulator.
ORA	R M	Logical OR register or memory with the accumulator	The contents of the accumulator are logically OR with M the contents of the register or memory, and result is placed in the accumulator.
ORI	8-bit data	Logical OR immediate with the accumulator	The contents of the accumulator are logically OR with the 8-bit data and the result is placed in the accumulator.
RLC	None	Rotate the accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.
RRC	None	Rotate the accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.
RAL	None	Rotate the	Each binary bit of the accumulator is rotated

		accumulator left through carry	left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.
RAR	None	Rotate the accumulator right through carry	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0.
CMA	None	Complement accumulator	The contents of the accumulator are complemented. No flags are affected.
CMC	None	Complement carry	The Carry flag is complemented. No other flags are affected.
STC	None	Set Carry	Set Carry

3) BRANCHING INSTRUCTIONS:

Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction.

The three types of branching instructions are:

1. Jump (unconditional and conditional)
2. Call (unconditional and conditional)
3. Return (unconditional and conditional)

Opcode	**Operand**	**Meaning**	**Explanation**
JMP	**16-bit addre**	Jump uncondition	The program sequence is transferred to the memory

<table>
<tr><td></td><td>ss</td><td>ally</td><td>address given in the operand.</td></tr>
<tr><td>
<table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>JC</td><td>Jump on Carry</td><td>CY=1</td></tr>
<tr><td>JNC</td><td>Jump on no Carry</td><td>CY=0</td></tr>
<tr><td>JP</td><td>Jump on positive</td><td>S=0</td></tr>
<tr><td>JM</td><td>Jump on minus</td><td>S=1</td></tr>
<tr><td>JZ</td><td>Jump on zero</td><td>Z=1</td></tr>
<tr><td>JNZ</td><td>Jump on no zero</td><td>Z=0</td></tr>
<tr><td>JPE</td><td>Jump on parity even</td><td>P=1</td></tr>
<tr><td>JPO</td><td>Jump on parity odd</td><td>P=0</td></tr>
</table>
</td><td>16-bit address</td><td>Jump conditionally</td><td>The program sequence is transferred to the memory address given in the operand based on the specified flag of the PSW.</td></tr>
<tr><td>
<table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>CC</td><td>Call on Carry</td><td>CY=1</td></tr>
<tr><td>CNC</td><td>Call on no Carry</td><td>CY=0</td></tr>
<tr><td>CP</td><td>Call on positive</td><td>S=0</td></tr>
<tr><td>CM</td><td>Call on minus</td><td>S=1</td></tr>
<tr><td>CZ</td><td>Call on zero</td><td>Z=1</td></tr>
</table>
</td><td>16-bit address</td><td>Unconditional subroutine call</td><td>The program sequence is transferred to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.</td></tr>
</table>

<table>
<tr>
<td>
<table>
<tr><td>CNZ</td><td>Call on no zero</td><td>Z=0</td></tr>
<tr><td>CPE</td><td>Call on parity even</td><td>P=1</td></tr>
<tr><td>CPO</td><td>Call on parity odd</td><td>P=0</td></tr>
</table>
</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td>None</td>
<td>Return from subroutine unconditionally</td>
<td>The program sequence is transferred from the subroutine to the calling program.</td>
</tr>
<tr>
<td>
<table>
<tr><td>Opcode</td><td>Description</td><td>Flag Status</td></tr>
<tr><td>RC</td><td>Return on Carry</td><td>CY=1</td></tr>
<tr><td>RNC</td><td>Return on no Carry</td><td>CY=0</td></tr>
<tr><td>RP</td><td>Return on positive</td><td>S=0</td></tr>
<tr><td>RM</td><td>Return on minus</td><td>S=1</td></tr>
<tr><td>RZ</td><td>Return on zero</td><td>Z=1</td></tr>
<tr><td>RNZ</td><td>Return on no zero</td><td>Z=0</td></tr>
<tr><td>RPE</td><td>Return on parity even</td><td>P=1</td></tr>
<tr><td>RPO</td><td>Return on parity odd</td><td>P=0</td></tr>
</table>
</td>
<td>None</td>
<td>Return from subroutine conditionally</td>
<td>The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW and the program execution begins at the new address.</td>
</tr>
<tr>
<td>PCHL</td>
<td>None</td>
<td>Load the program</td>
<td>The contents of registers H & L are copied into the program</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>counter with HL contents</td><td>counter. The contents of H are placed as the high-order byte and the contents of L as the loworder byte.</td></tr>
<tr><td>RST</td><td>0-7</td><td>Restart</td><td>The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.

<table><tr><td>Instruction</td><td>Restart Address</td></tr><tr><td>RST 0</td><td>0000H</td></tr><tr><td>RST 1</td><td>0008H</td></tr><tr><td>RST 2</td><td>0010H</td></tr><tr><td>RST 3</td><td>0018H</td></tr><tr><td>RST 4</td><td>0020H</td></tr><tr><td>RST 5</td><td>0028H</td></tr><tr><td>RST 6</td><td>0030H</td></tr><tr><td>RST 7</td><td>0038H</td></tr></table>The 8085 has additionally 4 interrupts, which can generate RST instructions internally and doesn’t require any external hardware. Following are those instructions and their Restart addresses −

<table><tr><td>Interrupt</td><td>Restart Address</td></tr><tr><td>TRAP</td><td>0024H</td></tr><tr><td>RST 5.5</td><td>002CH</td></tr><tr><td>RST 6.5</td><td>0034H</td></tr></table></td></tr>
</table>

			RST 7.5 003CH

4) ARITHMETIC INSTRUCTIONS:

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADD K.
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADC K
ADI	8-bit data	Add the immediate to the accumulator	The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADI 55K
ACI	8-bit data	Add the immediate to the accumulator with carry	The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.

			Example − ACI 55K
LXI	Reg. pair, 16bit data	Load the register pair immediate	The instruction stores 16-bit data into the register pair designated in the operand. **Example** − LXI K, 3025M
DAD	Reg. pair	Add the register pair to H and L registers	The 16-bit data of the specified register pair are added to the contents of the HL register. **Example** − DAD K
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. **Example** − SUB K
SBB	R M	Subtract the source and borrow from the accumulator	The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. **Example** − SBB K
SUI	8-bit data	Subtract the immediate from the accumulator	The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator. **Example** − SUI 55K
SBI	8-bit data	Subtract the immediate from the accumulator with borrow	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. **Example** − XCHG

INR	R M	Increment the register or the memory by 1	The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. **Example** − INR K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. **Example** − INX K
DCR	R M	Decrement the register or the memory by 1	The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. **Example** − DCR K
DCX	R	Decrement the register pair by 1	The contents of the designated register pair are decremented by 1 and their result is stored at the same place. **Example** − DCX K
DAA	None	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit BCD digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

			Example − DAA

5) DATA TRANSFER INSTRUCTIONS:

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. **Example** − MOV K, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. **Example** − MVI K, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. **Example** − LDA 2034K
LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. **Example** − LDAX K
LXI	Reg. pair, 16-bit data	Load the register pair immediate	The instruction loads 16-bit data in the register pair designated in the register or the memory. **Example** − LXI K, 3225L

LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. **Example** − LHLD 3225K
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example** − STA 325K
STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. **Example** − STAX K
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example** − SHLD 3225K
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

			Example − XCHG
SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H register provide the high-order address and the contents of the L register provide the low-order address. **Example** − SPHL
XTHL	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1). **Example** − XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. **Example** − PUSH K
POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory

			location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. **Example** − POPK
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. **Example** − OUT K9L
IN	8-bit port address	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. **Example** − IN5KL

CONCLUSION: In this practical, I studied about the architecture of 8085 microprocessor as well as the instruction set of 8085 along with some examples.

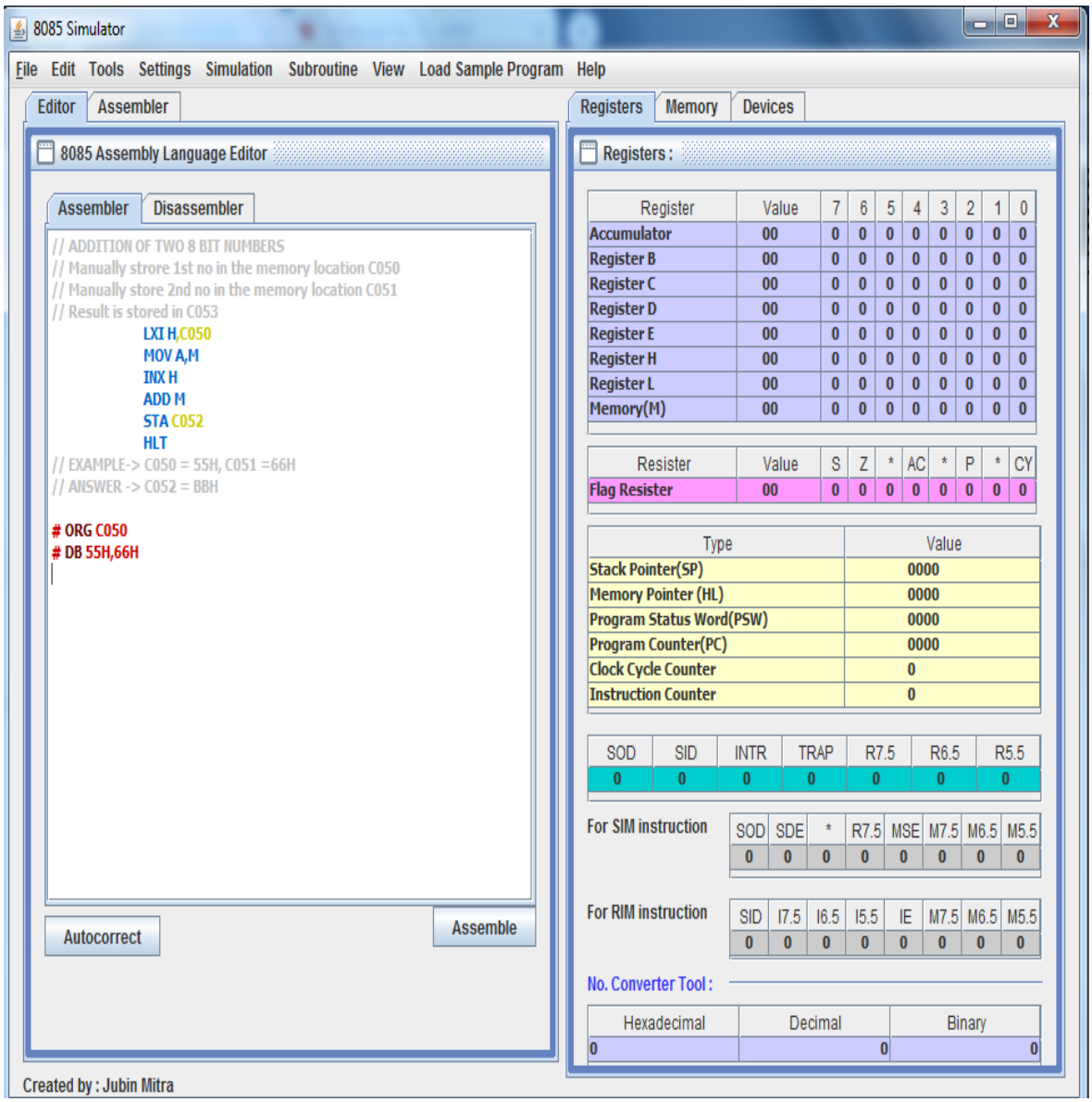
Practical No.2

Aim: Introduction to Jubin's Simulator for Assembly Language Programming of 8085.

8085 Simulator:

8085 Simulator is powerful application that supplies 8085 microprocessor users and educators with user-friendly graphical development environment for Windows with integrated simulator.

8085 simulator is on Java platform, Supports all 8085 interrupts and instructions. User enters 8085 assembly language programs, whose simulation is performed by the software.



Features:-

(a) Assembler Editor

- Can load Programs.
- Supports assembler directives.
- Number parameters can be given in binary, decimal and hexadecimal format.
- Supports writing of comments.
- Supports labeling of instructions, even in macros.
- Has error checking facility.

(b) Disassembler Editor

- Supports loading of Intel specific hex file format.

- It can successfully reverse trace the original program from the assembly code, in most of the cases.

(c) Memory Editor

- Can directly update data in a specified memory location
- It has 3 types of interface; user can choose from it according to his need.
  - Show entire memory content
  - Show only loaded memory location
  - Store directly to specified memory location
- Allows user to choose memory range.

(d) I/O Editor

- It is necessary for peripheral interfacing.
- Enables direct editing of content

(e) Interrupt Editor

- All possible interrupts are supported.
- The simulation can be reset any time by pressing the clear memory in the settings tab.

(f) Program execution/Simulator

There are 2 modes of simulator engine:

Run all at a Time: It takes the current settings from the simulation speed level and starts execution accordingly.

Step by Step: It is manual mode of control of FORWARD and BACKWARD traversal of instruction set.

Conclusion: In this practical, I learned about Jubin's 8085 simulator software and its applications. I also learned about how to use the software.