Syeda Reeha Quasar

14114802719

4C7

Aim

To study and understand GnuSim8085 – 8085 microprocessor stimulator interface and its structure.

Experiment - 1

Computer Organization and Architecture

# **EXPERIMENT – 1**

## **Aim:**

## To study and understand GNUSim8085 – 8085 microprocessor stimulator interface and its structure.

## **Theory:**

Microprocessor is a controlling unit of a micro-computer, fabricated on a small chip capable of performing ALU (Arithmetic Logical Unit) operations and communicating with the other devices connected to it.

Microprocessor consists of an ALU, register array, and a control unit. ALU performs arithmetical and logical operations on the data received from the memory or an input device. Register array consists of registers identified by letters like B, C, D, E, H, L and accumulator. The control unit controls the flow of data and instructions within the computer.

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 up to 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

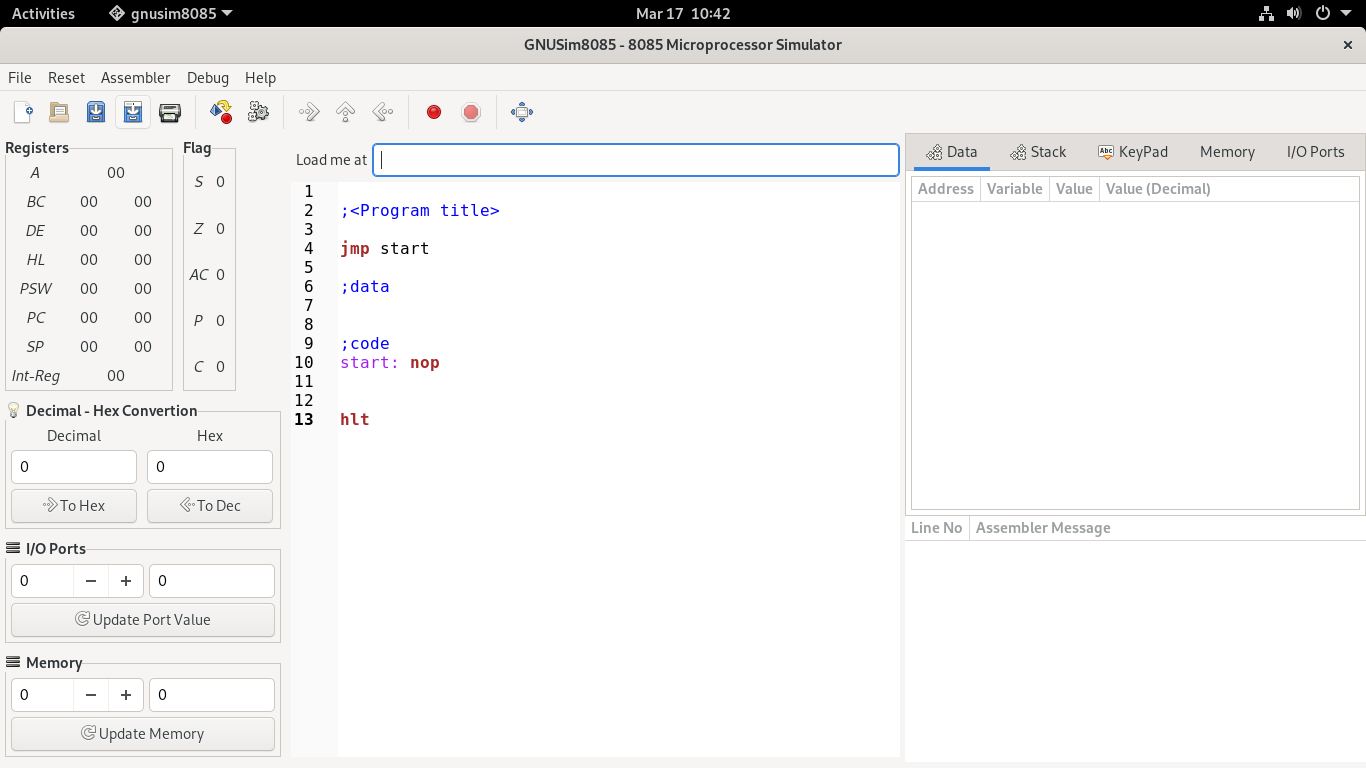
It is used in washing machines, microwave ovens, mobile phones, etc.

## **About Interface:**

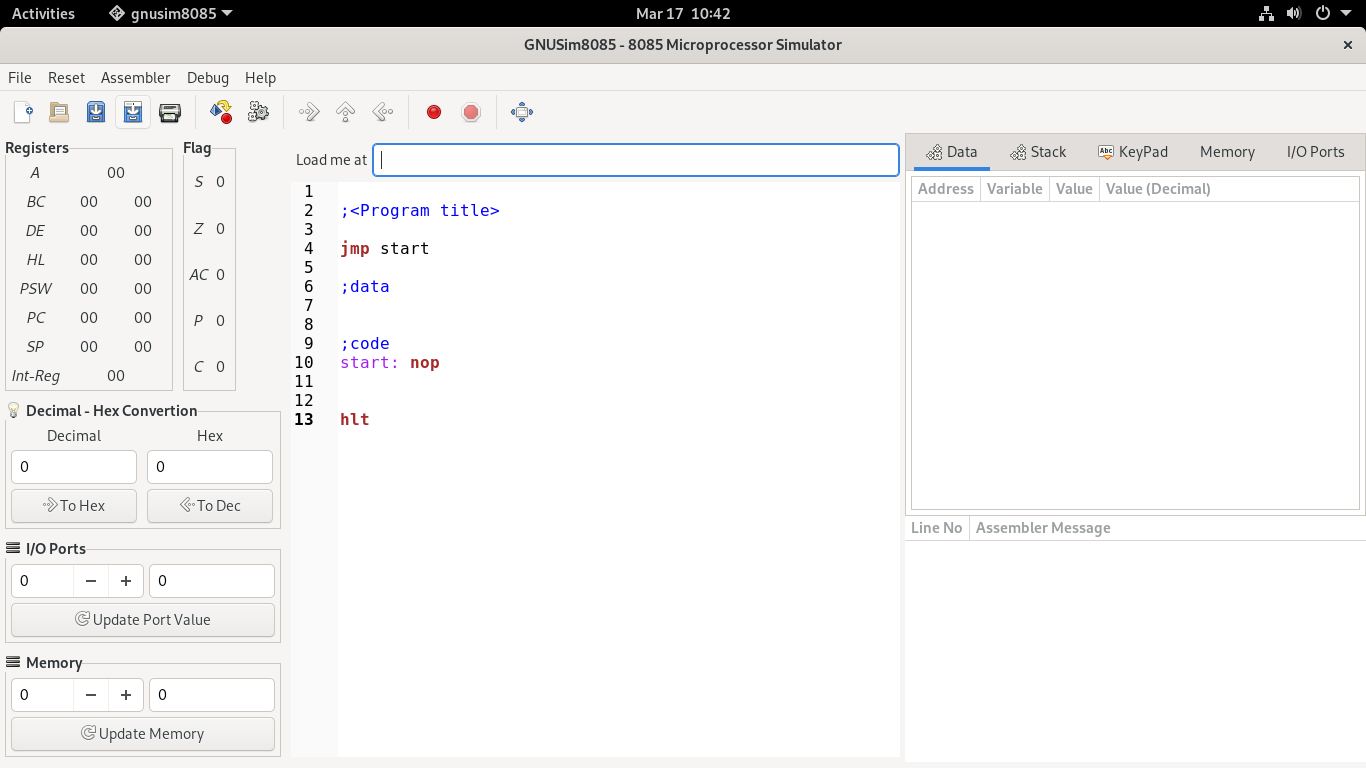
GNUSim 8085 interface is majorly divided into 4 sections:

* Top section - menu bar
* Left section
* Right section
* Center Area for writing instructions

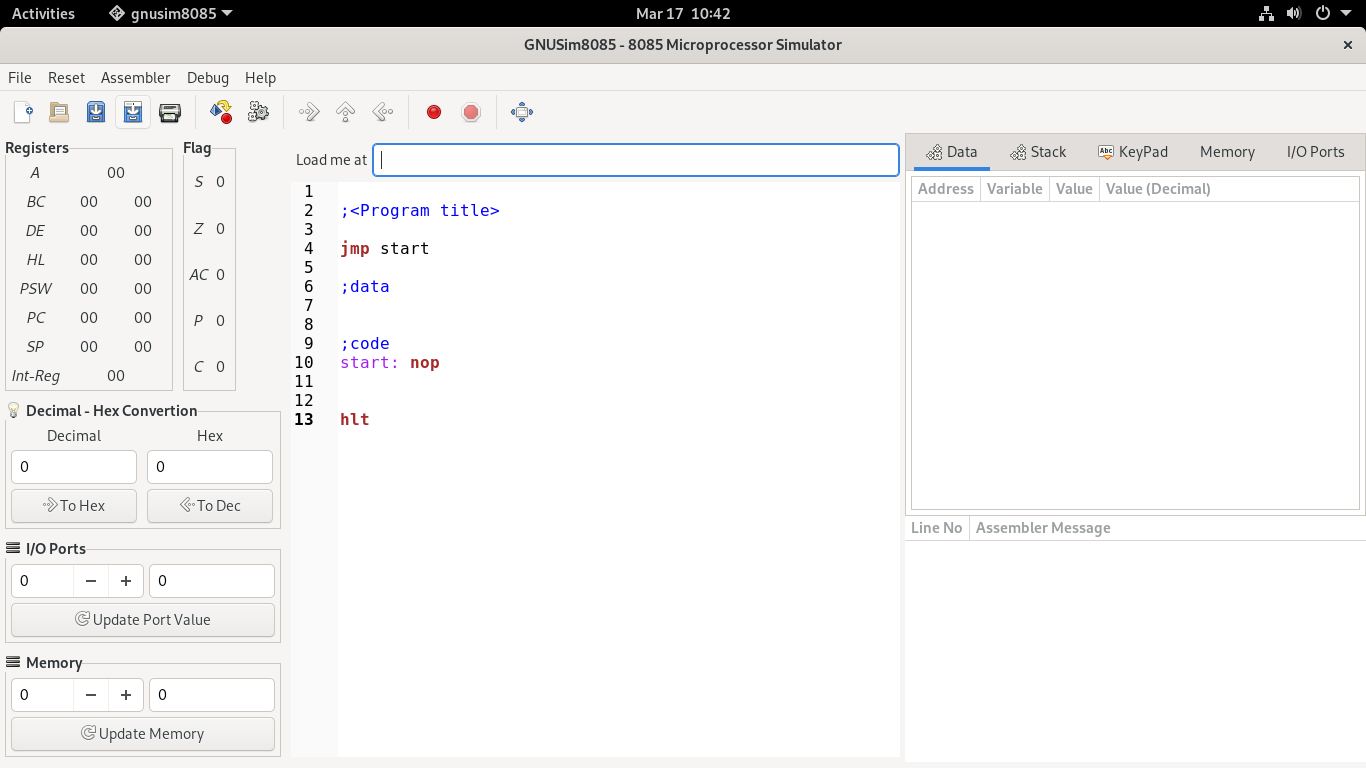
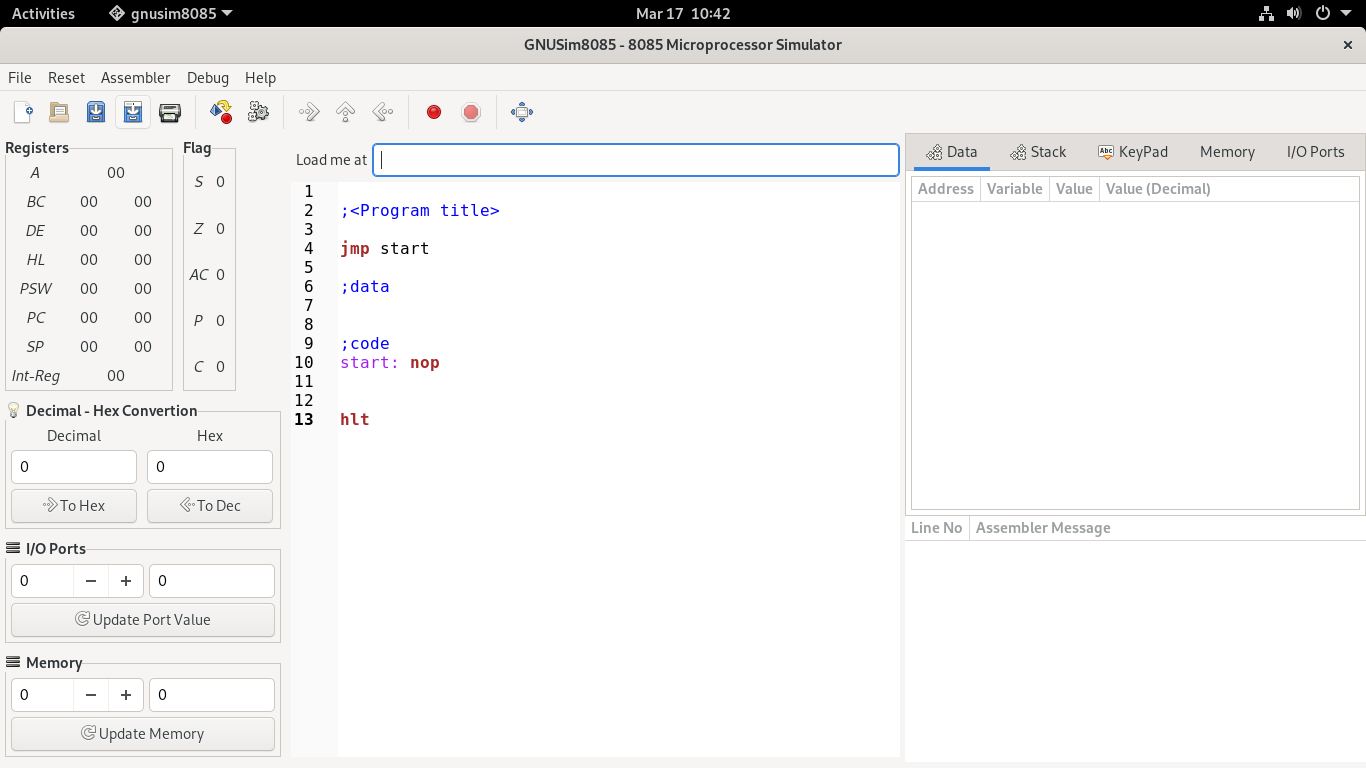
## **Interface:**

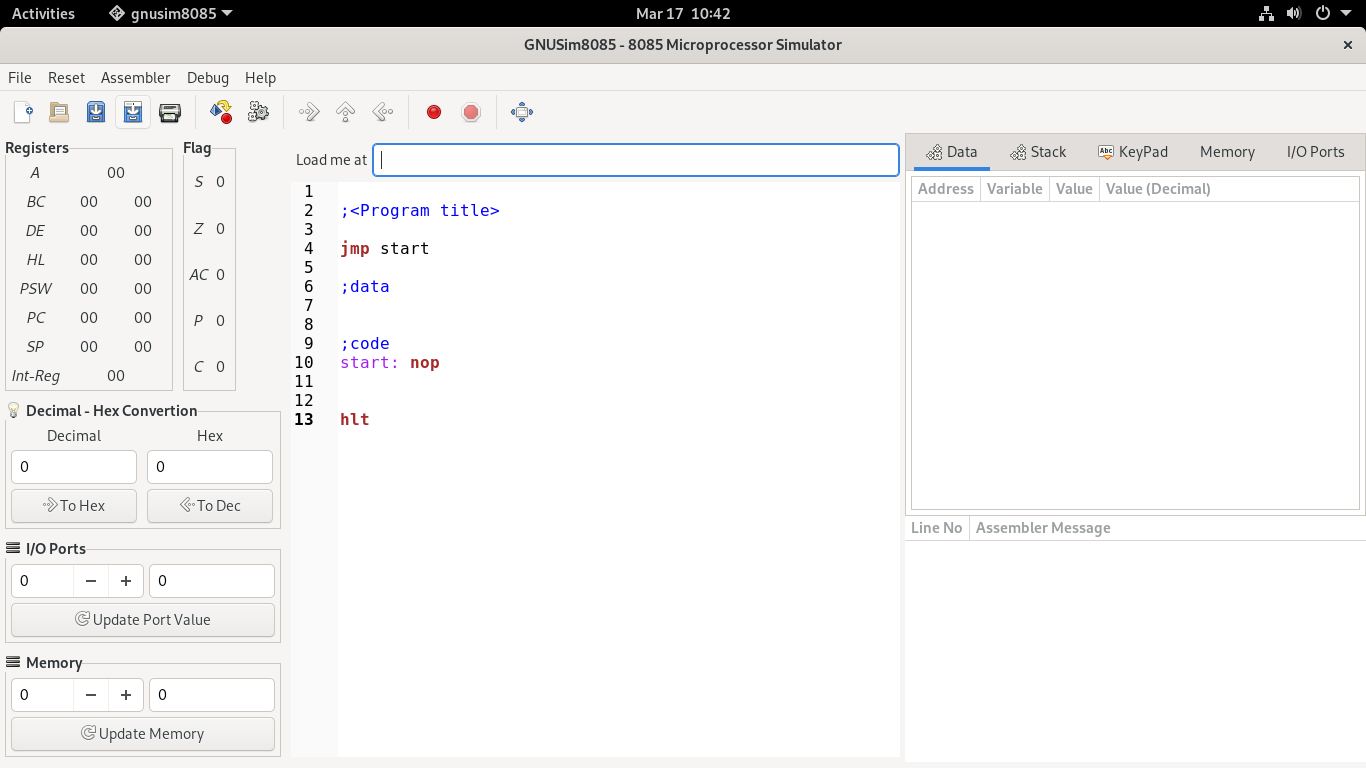
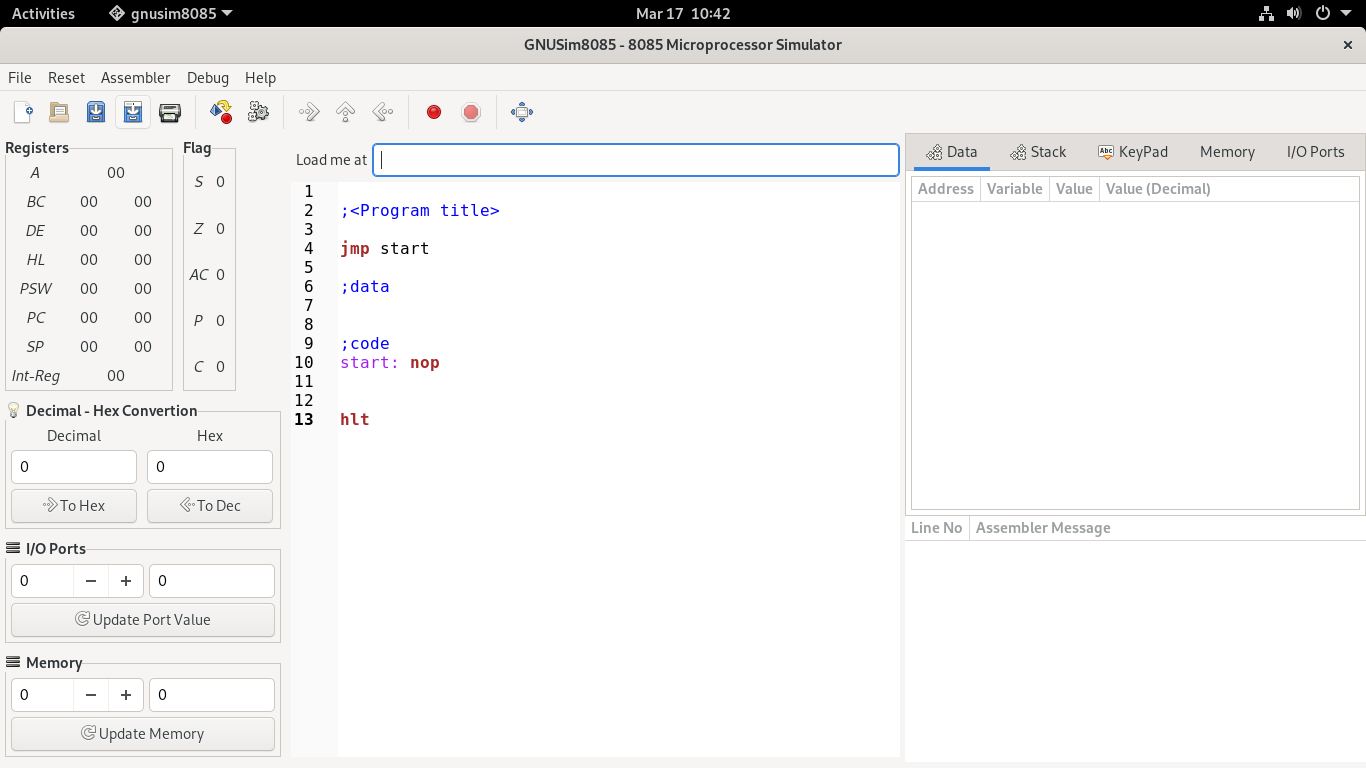


## Top Section – Menu Section

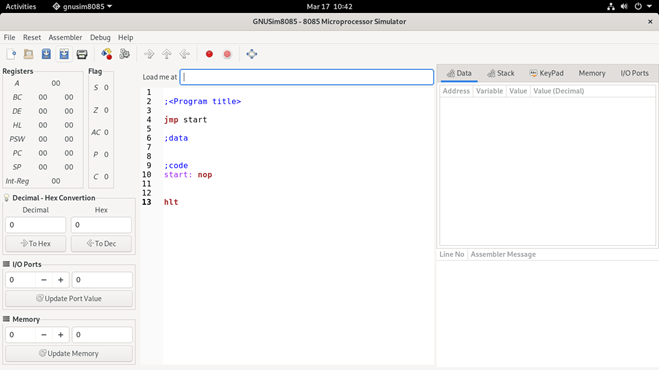


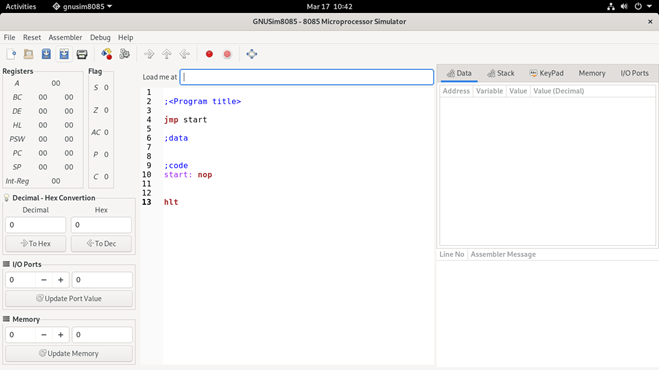
## Left Section:





## Right Section:





## **Top Section:**

This contains various option to create new program, save program, build program, execute, Debug, reset and various other options program execution, saving and printing.

## **Left section:**

This is the display section where we can see various registers and flags and observe the outputs changes and processing.

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

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 - PC**

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 - SC**

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

**Instruction registers 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.

**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)

We also have a decimal - hexadecimal converter for converting decimal to hexadecimal and vice versa for program inputs and calculations.

There is 2 more sections for input output ports value assignment to the ports and for updating the memory ports.

## **Right Section:**

It has 2 major sections the lower section displays error in program if there any and compilation status. These contains message from the assembler.

Upper right section is for entering data for various address position, editable I/O ports, editable memory tabulation and stack Display.

## **Center Section:**

It’s the space where we write various statements to be executed.

Instruction sets are instruction codes to perform some tasks. It is classified into four categories.

* Transfer Instructions
* Arithmetic Instructions
* Logical Instructions
* Branching Instructions

### 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 |

### 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 |
| 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 |
| 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 |

### 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 accumulator left through carry | Each binary bit of the accumulator is rotated 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 |

### Branching Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Operand | Meaning | Explanation |
| JMP | **16-bit address** | Jump unconditionally | The program sequence is transferred to the memory address given in the operand. |
| |  |  |  | | --- | --- | --- | | Opcode | Description | Flag Status | | JC | Jump on Carry | CY=1 | | JNC | Jump on no Carry | CY=0 | | JP | Jump on positive | S=0 | | JM | Jump on minus | S=1 | | JZ | Jump on zero | Z=1 | | JNZ | Jump on no zero | Z=0 | | JPE | Jump on parity even | P=1 | | JPO | Jump on parity odd | P=0 | | **16-bit address** | Jump conditionally | The program sequence is transferred to the memory address given in the operand based on the specified flag of the PSW. |
| |  |  |  | | --- | --- | --- | | Opcode | Description | Flag Status | | CC | Call on Carry | CY=1 | | CNC | Call on no Carry | CY=0 | | CP | Call on positive | S=0 | | CM | Call on minus | S=1 | | CZ | Call on zero | Z=1 | | CNZ | Call on no zero | Z=0 | | CPE | Call on parity even | P=1 | | CPO | Call on parity odd | P=0 | | **16-bit address** | Unconditional subroutine call | 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. |
| RET | **None** | Return from subroutine unconditionally | The program sequence is transferred from the subroutine to the calling program. |
| |  |  |  | | --- | --- | --- | | Opcode | Description | Flag Status | | RC | Return on Carry | CY=1 | | RNC | Return on no Carry | CY=0 | | RP | Return on positive | S=0 | | RM | Return on minus | S=1 | | RZ | Return on zero | Z=1 | | RNZ | Return on no zero | Z=0 | | RPE | Return on parity even | P=1 | | RPO | Return on parity odd | P=0 | | **None** | Return from subroutine conditionally | 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. |
| PCHL | **None** | Load the program counter with HL contents | The contents of registers H & L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the loworder byte. |
| RST | **0-7** | Restart | The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.   |  |  | | --- | --- | | Instruction | Restart Address | | RST 0 | 0000H | | RST 1 | 0008H | | RST 2 | 0010H | | RST 3 | 0018H | | RST 4 | 0020H | | RST 5 | 0028H | | RST 6 | 0030H | | RST 7 | 0038H |   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 −   |  |  | | --- | --- | | Interrupt | Restart Address | | TRAP | 0024H | | RST 5.5 | 002CH | | RST 6.5 | 0034H | | RST 7.5 | 003CH | |

## **Center Section – Instruction Writing Section**

## 