Lab 01

SUB: CSA

NAME : JARIWALA SAHIL YOGESHKUMAR

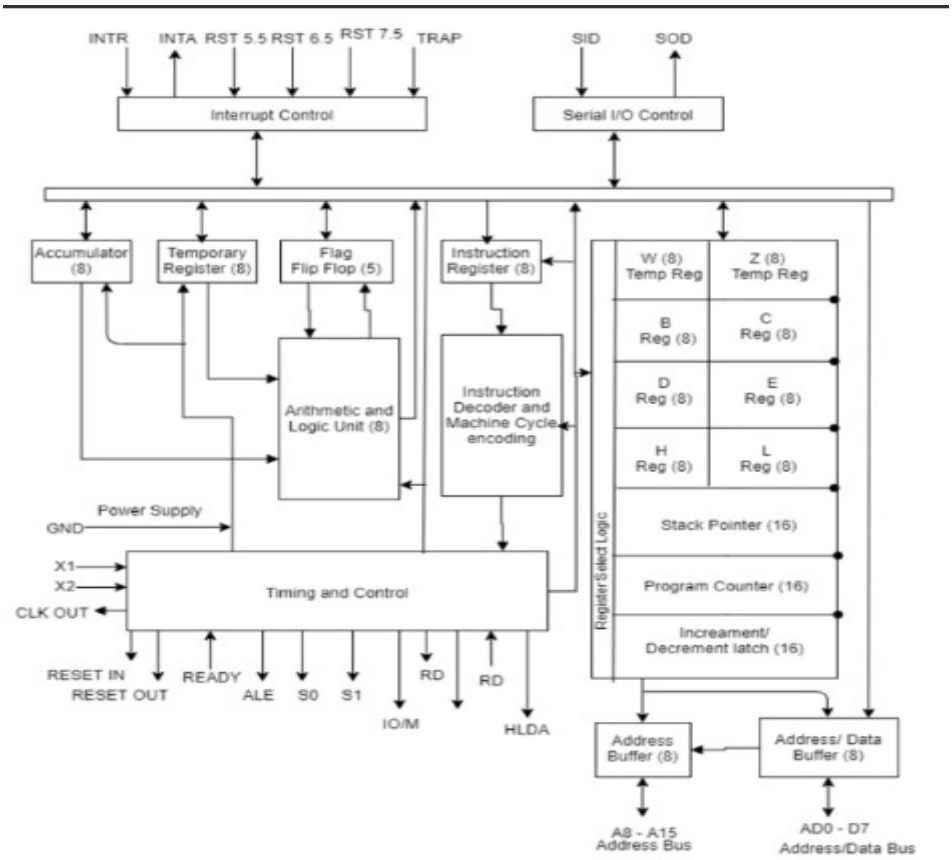
ROLL NO: CE049

AIM: STUDY OF 8085 MICROPROCESSOR

- ***Study of the 8085 Architecture :***

Generally, the 8085 is an 8-bit microprocessor and it was launched by the Intel team in the year of 1976 with the help of NMOS technology. This processor is the updated version of the microprocessor. The configurations of 8085 microprocessor mainly include data bus-8-bit, address bus-16 bit, program counter-16-bit, stack pointer-16 bit, registers 8-bit, +5V voltage supply, and operates at 3.2 MHz single segment CLK. The applications of 8085 microprocessor are involved in microwave ovens, washing machines, gadgets, etc.

Architecture of 8085 microprocessor is as follow:



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.

It can perform operations that are given below:

1.Operates on and stores 8-bit data.
2.It executes arithmetic and logic operations.
3.8085 also sequences the instructions to be executed.
4.Stores data temporarily.

Functional Units of 8085:

1. Registers: These are nothing but set of flip flops. These are basically used to hold (store) the data.

General purpose registers– 8085 microprocessors contain 6 general purpose registers that are present inside the microprocessor and stores 8-bit data in order to execute a program.
These general purpose registers are B, C, D, E, H and L. These registers can be combined to form pairs – BC, DE and HL in order to execute the 16-bit operation.
Temporary registers: These registers are used by the ALU to store the data on temporary basis and these are not accessed by the programmer. These are of 2 types:
1.Temporary data register – It is an 8-bit register that holds the operand and provides it to the ALU for program execution.
2.W and Z register – These registers are also used to hold the temporary values

2.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.

3. Stack Pointer (SP): It is also a 16-bit register and is a part of memory. The data is stored in the stack in serial format and stack pointer generally stores the address of the last data element stored in the stack. Thus the stack is based on LIFO. Whenever a new data is added in the stack, then the stack pointer starts pointing towards the very next memory location. As against, when a data element is removed from the stack, then the stack pointer points to previous occupied memory location.

4. Accumulator: It is an 8-bit register that stores the result of the operation performed by the ALU. It is also known as register A.

5. Flags: Flag register basically holds the status of the current result generated by the ALU and not the actually generated result. Thus we can say it is used to test the data conditions. 8085 has 5 flags that shows 5 different data conditions. These are carry, sign, zero, parity and auxiliary carry flags. However, the mostly used are: sign, carry and zero.

- ***To study the basic data transfer instructions, the opcodes and operands.***

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	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the

		indirect	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
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
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

- ***<u>To write a simple program of addition of two numbers and to check the status of each of the flag</u>***.

MVI B,55H (move immediate to Register B)

MOV A, B (copy data of B into Accumulator)

MVI B,33H (move immediate to Register B)

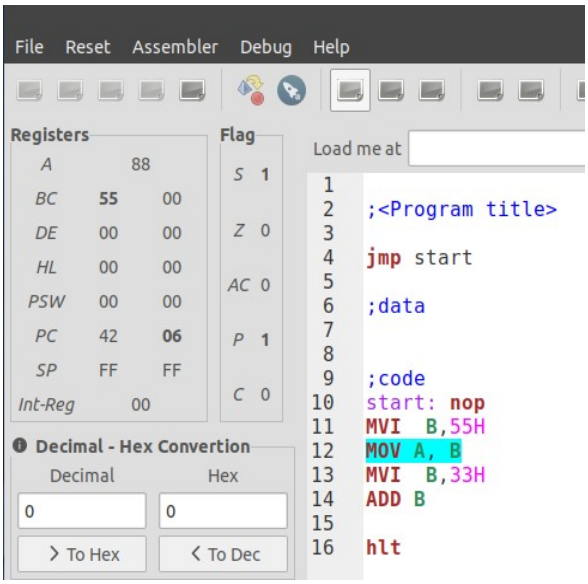
ADD B (add data of B with Accumulator)

HLT

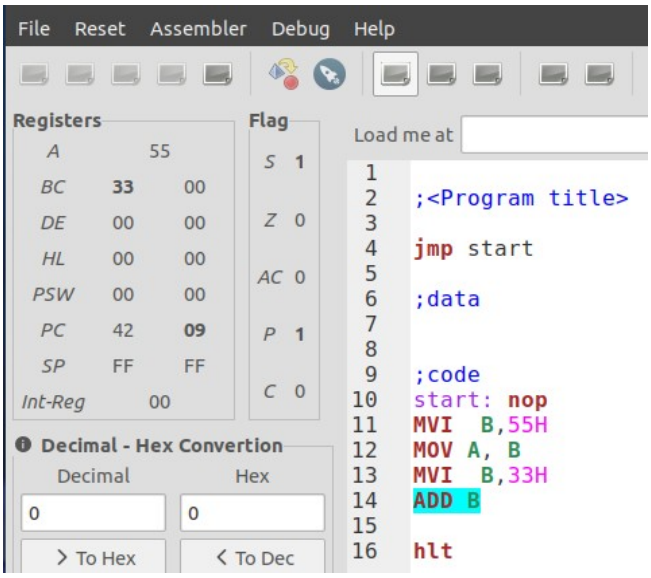
OUTPUT:

sign and parity flag are one . Other all are zero.

1.



2.



3.