# Experiment 2

August 12, 2020

## AIM:

Addition of two 8 bit numbers and

1. Sum is also 8 bit number
2. Sum is 16 bit number

## SOFTWARE USED :

GNUSim 8085

## THEORY:

The Intel 8085 is an 8-bit microprocessor produced by Intel and introduced in March 1976. It is a software-binary compatible with the more-famous Intel 8080 with only two minor instructions added to support its added interrupt and serial input/output features.

### INSTRUCTIONS IN 8085:

An instruction of a computer is a command given to the computer to perform a specified operation on given data. In microprocessor, the instruction set is the collection of the instructions that the microprocessor is designed to execute.

The programmer writes a program in assembly language using these instructions. These instructions have been classified into the following groups:

1. **Data Transfer Group**
2. **Arithmetic Group**
3. **Logical Group**
4. **Branch Control Group**
5. **I/O and Machine Control Group**

These are briefly explained below

1. **Data Transfer Group**

Instructions, which are used to transfer data from one register to another register, from memory to register or register to memory, come under this group. Examples are: MOV, MVI, LXI, LDA, STA etc. When an instruction of data transfer group is executed, data is transferred from the source to the destination without altering the contents of the source

2. **Arithmetic Group**

The instructions of this group perform arithmetic operations such as addition, subtraction; increment or decrement of the content of a register or memory. Examples are: ADD, SUB, INR, DAD etc**.**

3. **Logical Group**

The Instructions under this group perform logical operation such as AND, OR, compare, rotate etc. Examples are: ANA, XRA, ORA, CMP, and RAL etc.

4. **Branch Control Group**

This group includes the instructions for conditional and unconditional jump, subroutine call and return, and restart. Examples are: JMP, JC, JZ, CALL, CZ, RST etc.

5. **I/O and Machine Control Group**

This group includes the instructions for input/output ports, stack and machine

control. Examples are: IN, OUT, PUSH, POP, and HLT etc

## ALGORITHM

a) Sum is 8 bit

- Get address of first number in H-L pair
- First number in accumulator
- Increment content of H-L pair
- Add first and second number
- Store sum in 2503 H
- Stop

b) Sum is 16 bit

- Address of first number in H-L pair
- MSBs of sum in in register C. Initial value=00
- First number in accumulator
- Address of Second number in H-L pair
- Add the two numbers
- Is carry? If no, go to label AHEAD
- If yes, increment C
- LSB of sum in 2503 H
- MSB of sum in accumulator
- MSB of sum in 2504 H
- Stop

## FLOW CHART:

a) Sum is 8 bit

```
graph TB
st([Start]) --> op1[Load H-L pair with first operands memory address]

op1 --> op2[Move the first operand from memory to accumulator]
op2 --> op3[Increament H-L pair to point next memory location]
op3 --> op4[Add with A]
op4 --> op5[Increment H-L pair]
```

```
op5 --> op6[Move the result from accumulator to memory]
op6 --> Stop
```

b) Sum is 16 bit

```
graph TB
st[Start]-->op1[Load H-L pair with first operands memory address]
op1 --> op2[move the first operand from memory to accumulator]
op2 --> op3[Increament H-L pair to point next memory location]
op3 --> op4[Initialize register  C = 00]
op4 -->op5[Add M with A]
op5 --> cond{If Carry ?}
cond -->|yes|op6[Increment register C]
op6 -->op7[Increment H-L pairt]
op7 -->op8[Move the result to memory]
op8 -->op9[Increment H-L pair and move carry from register C to memory]
op9 --> Stop

cond -->|no|op7
```

---

```
st=>start: Start
e=>end: Stop
op1=>operation: Load H-L pair with first operands memory address
op2=>operation: move the first operand from memory to accumulator
op3=>operation: Increament H-L pair to point next memory location
op4=>operation: Add with A
op5=>operation: Increment H-L pair
op6=>operation: Move the result from accumulator to memory

st->op1->op2->op3->op4->op5->op6->e
```

b) Sum is 16 bit

```
st=>start: Start
e=>end: Stop
op1=>operation: Load H-L pair with first operands memory address
op2=>operation: move the first operand from memory to accumulator
op3=>operation: Increament H-L pair to point next memory location
op4=>operation: Initialize register  C = 00
op5=>operation: Add M with A
cond=>condition: if carry ?
op6=>operation: Increment register C
op7=>operation: Increment H-L pairt
op8=>operation: Move the result to memory
op9=>operation: Increment H-L pair and move carry from register C to memory


st->op1->op2->op3->op4->op5->cond
cond(yes)->op6->op7->op8->op9->e
cond(no)-

op6[Increment register C]
op7[Increment H-L pairt]
op8[Move the result to memory]
op9[Increment H-L pair and move carry from register C to memory]
```

## PROGRAM AND OUTPUT:

### a) Sum is 8 bit

| Add | Mnemo | Op | Comments | Addressing Mode | Machine Cycle | T-States |
|---|---|---|---|---|---|---|
| 2000 | LXI H | 2501H | Load H-L pairs with address 2501H | Immediate | Opcode fetch + 2 Memory read | 10 |
| 2003 | MOV A, M | | Move first operand from memory to accumulator | Indirect | Opcode fetch + Memory read | 7 |
| 2004 | INX H | | Increment H-L pair | Register | Opcode fetch | 6 |
| 2005 | ADD M | | Add M with A | Register | Opcode fetch + Memory read | 7 |
| 2006 | INX H | | Increment H-L pair | Register | Opcode fetch | 6 |
| 2007 | MOV M, A | | Move contents of Accumulator to memory add sotred in H-L Pair | Indirect | Opcode fetch + memory write | 7 |
| 2008 | HLT | | HALT | - | Opcode fetch | 4 |

### Output

**Data:**

```
2501- 04 H

2502- 02 H
```

**Result:**

```
2503- 06 H
```

## b) Sum is 16 bit

| Add | Mn | Op | Comments | Addressing Mode | Machine Cycle | T-States |
|---|---|---|---|---|---|---|
| 2000 | LXI H | 2501 H | Load H-L pair with Address 300H | Immediate | Opcode fetch and 2 memory read | 10 |
| 2003 | MVI C | 00 | Initialize C register with zero value | Immediate | Opcode fetch + memory read | 7 |
| 2005 | MOV A, M | | Move first operand form memory to A | Indirect | Opcode fetch + memory read | 7 |
| 2006 | INX H | | Increment H-L Pair | Register | Opcode fetch | 6 |
| 2007 | ADD M | | Add M with A | Indirect | Opcode fetch + memory read | 7 |
| 2008 | JNC | AHEAD | Jump to label if no carry | Immediate | Opcode fetch + 2 memory read | 10 |
| 2008 | INR | C | Increment reg C | Register | Opcode fetch | 4 |
| 2000 | STA | 2503H | Load data of accumulator to memory | Direct | Opcode fetch + 2 memory read + 2 memory write | 13 |
| 200F | MOV A, C | | Move content of C register to accumulator | Register | Opcode fetch | 4 |
| 2010 | STA | 2504H | Load data of accumulator to memory | Direct | Opcode fetch + 2 memory read + 2 memory write | 13 |
| 2013 | HLT | | HALT | - | Opcode | 4 |

## Output

**Data:**

```
2501- 98 H

2502- 9A H
```

**Result:**

```
2503- 32 H, LSBs of sum

2504- 01 H, MSBs of sum
```

## Result:

The sum of two 8 bit numbers in 8085 is obtained for both the cases when

1) Sum of the numbers is 8 bit
2) Sum of the numbers is 16 bit.