# Project Description

### **What is a Microprocessor?**

The word Microprocessor comes from the combination of micro and processor. In the late 1960’s, processors were built using discrete elements. These devices performed the required operation, but were too large and too slow.In the early 1970’s the microchip was invented.

All of the components that made up the processor were now placed on a single piece of silicon. The size became several thousand times smaller and the speed became several hundred times faster. The “Micro”Processor was born.

Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0’s and 1’s.To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor’s design.

So, the microprocessor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

**Internally, the microprocessor is made up of 3 main units.**

**– The Arithmetic/Logic Unit (ALU).**

– **The Control Unit.**

– **An array of registers for holding data while it is being manipulated.**

### **What is a 8085 Microprocessor?**

The 8085 (from Intel) is an 8-bit microprocessor. The 8085 uses a total of 246 bit patterns to form its instruction set. These 246 patterns represent only 74 instructions. The reason for the difference is that some (actually most) instructions have multiple different formats.

Because it is very difficult to enter the bit patterns correctly, they are usually entered in hexadecimal instead of binary. For example, the combination 0011 1100 which translates into “increment the number in the register called the accumulator”, is usually entered as 3C.

# Instructions Implemented

### **DATA TRANSFER INSTRUCTIONS**

### **MOV (Rd, Rs / M, Rs / Rd, M):**

This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.

Example: MOV B, C or MOV B, M

### **MVI (Rd, data / M, data):**

The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: MVI B, 57H or MVI M, 57H

### **LDA 16-bit address:**

The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered.

Example: LDA 2034H

### **LDAX B/D Reg. pair:**

The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered.

Example: LDAX B

### **LXI Reg. pair, 16-bit data:**

The instruction loads 16-bit data in the register pair designated in the operand.

Example: LXI H, 2034H or LXI H, XYZ

### **LHLD 16-bit address:**

The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered.

Example: LHLD 2040H

### **STA 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 4350H

### **STAX Reg. pair:**

The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.

Example: STAX B

### **SHLD 16-bit address:**

The contents of register L are stored into 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. The contents of registers HL are not altered. 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 2470H

### **SPHL none:**

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. The contents of the H and L registers are not altered.

Example: SPHL

### **XCHG none:**

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

### **PUSH Reg. pair:**

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 highorder 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 B or PUSH A

### **POP Reg. 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: POP H or POP A

### **ARITHMETIC INSTRUCTIONS**

### **ADD R/M:**

The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.

Example: ADD B or ADD M

### **ADC R/M:**

The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.

Example: ADC B or ADC M

### **ADI 8-bit data:**

The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ADI 45H

### **ACI 8-bit data:**

The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ACI 45H

### **DAD Reg. pair:**

The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.

Example: DAD H

### **SUB R/M:**

The contents of the operand (register or memory ) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.

Example: SUB B or SUB M

### **SBB R/M:**

The contents of the operand (register or memory ) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SBB B or SBB M

### **SUI 8-bit data:**

The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.

Example: SUI 45H

### **SBI 8-bit data:**

The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.

Example: SBI 45H

### **DCR R/M:**

The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: DCR B or DCR M

### **DCX Reg. pair:**

The contents of the designated register pair are decremented by 1 and the result is stored in the same place.

Example: DCX H

### **INR R/M:**

The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: INR B or INR M

### **INX Reg. pair:**

The contents of the designated register pair are incremented by 1 and the result is stored in the same place.

Example: INX H

### **LOGICAL INSTRUCTIONS**

### **CMP R/M:**

The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < (reg/mem): carry flag is set

if (A) = (reg/mem): zero flag is set

if (A) > (reg/mem): carry and zero flags are reset

Example: CMP B or CMP M

### **CPI 8-bit data:**

The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < data: carry flag is set

if (A) = data: zero flag is set

if (A) > data: carry and zero flags are reset

Example: CPI 89H

### **ANA R/M:**

The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANA B or ANA M

### **ANI 8-bit data:**

The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANI 86H

### **XRA R/M:**

The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M

### **XRI 8-bit data:**

The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRI 86H

### **ORA R/M:**

The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORA B or ORA M

### **ORI 8-bit data:**

The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORI 86H

### **CMA none:**

The contents of the accumulator are complemented. No flags are affected.

Example: CMA

### **CMC none:**

The Carry flag is complemented. No other flags are affected. Example: CMC

### **STC none:**

The Carry flag is set to 1. No other flags are affected.

Example: STC

### **LOGICAL INSTRUCTIONS**

### **NOP none:**

No operation is performed. The instruction is fetched and decoded. However no operation is executed.

Example: NOP

### **HLT none:**

The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.

Example: HLT

### TIMING DIAGRAM

Representation of Various Control signals generated during Execution of an Instruction.

Following Buses and Control Signals must be shown in a Timing Diagram:

**– Higher Order Address Bus.**

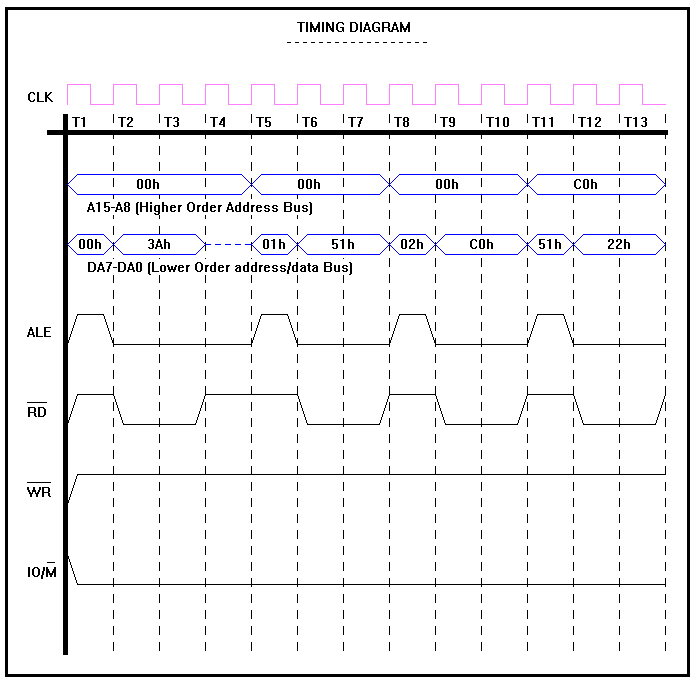
– **Lower Address/Data bus**

– **ALE**

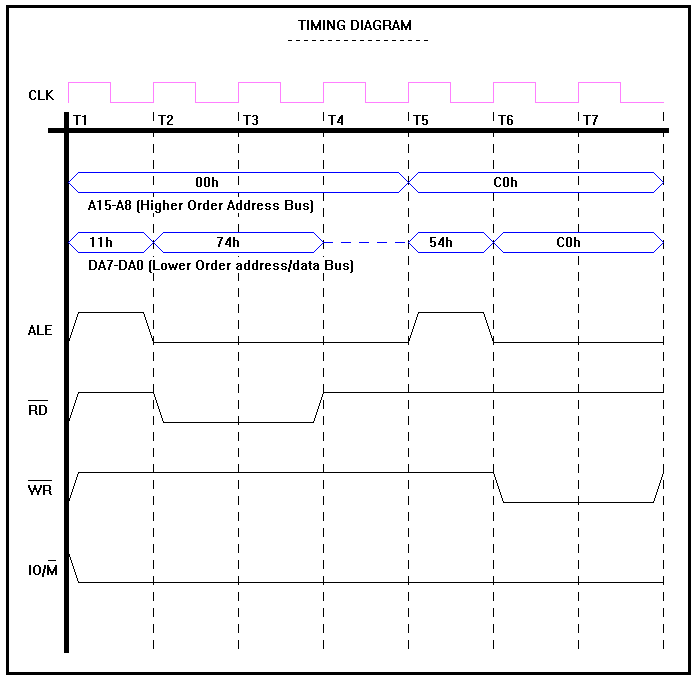
**– RD**

– **WR**

– **IO/M**

****

**Timing Diagram of LDA C051**

****

**Timing Diagram of MOV M, H**

### CONCLUSION

Through this project an attempt has been undertaken by us to simplify the working of 8085 Microprocessor.

This has been done by using C++ logic for different instructions, and creating the Timing Diagram for better understanding of the program.

### REFERENCES

1. Websites:-
2. <http://www.cplusplus.com/>
3. [https://www.youtube.com/](https://www.youtube.com/watch?v=uMgYwDokXi4) ( For timing diagram )
4. Books / Slides:-
5. 8085 Microprocessor by Ramesh S Gaonkar
6. opcodes-table-of-intel-8085
7. Softwares:-
8. Jubin's 8085 Simulator ver 2