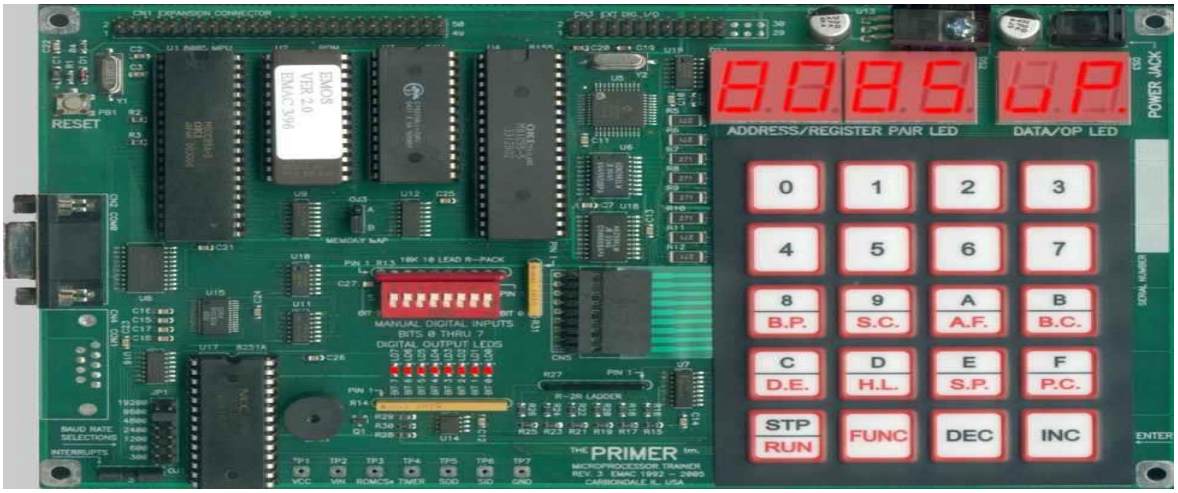
BARODA UNIVERSITY

Vadodara | Gujarat | India

A WORKSHOP MANNUAL OF MICROPROCESSOR AND INTERFACING



Prepared by: Assistant professor Maan Peshavaria
: Assistant professor Palak Vashi

8085

1. INTRODUCTION

- Microprocessor (MP) is a programmable logic device that is capable of data handling and performing data processing operations.
- More elaborately, it is –
  - ❖ Digital device: understands binary.
  - ❖ Programmable device: It can perform multiple tasks and can be instructed (i.e. programmed) to perform specific task (within its capability)
  - ❖ Clock driven: requires clock for its operation.
  - ❖ Capable of data handling: Storing data, communicating data with other devices, etc.
  - ❖ Capable of data processing: Performing various arithmetic and logic operations like addition, subtraction, magnitude comparison, ANDing, ORing, etc.
  - ❖ Available in the form of an Integrated circuit (IC).

2. COMPONENTS OF MICROPROCESSOR

- Primary component of any programmable system, e.g. computer, and is also referred as Central Processing Unit (CPU). CPU popularly consist of three main units viz. Arithmetic and logic unit (ALU), Register array (memory limited in size) and Control unit (CU).

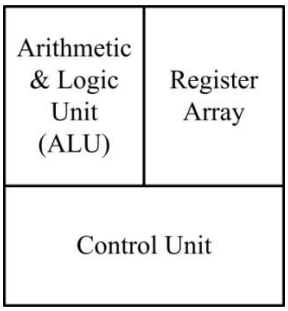


Figure 2.1: Central Processing Unit

ARITHMETIC AND LOGIC UNIT (ALU)

- This unit performs all the logical and arithmetic operations.
- Various arithmetic operations are: addition, subtraction, increment and decrement etc.
- Various logical operations are: AND, OR, NOT, XOR, etc.

TIMING AND CONTROL UNIT

- This unit controls the entire operations being performed by the system.
- It controls the operations of ALU, input/output devices and memory unit.
- This unit interprets the instructions and generates various timing and control signals.

REGISTERS

- A register is a very small amount of very fast memory that is built into the CPU in order to store the current data and instructions which are being executed by the CPU.

SYSTEM BUS

- A bus in a microprocessor-based system is defined as a group of separate wires which work together to perform a particular task.
- A microprocessor-based system, or microcomputer, has three types of buses which combine to transfer information between the microprocessor and other parts of the system, such as memory or input/output devices.
- Typical tasks performed by these buses include selecting the source or destination location address for a data transfer, actually moving the data from one part of the system to another, and finally, controlling and synchronizing the electronic devices involved in the data transfer process.
- The three microprocessor buses are called the address bus, data bus and control bus, and these names give a strong clue as to their function:

1) Address Bus

- The address bus contains the connections between the microprocessor and memory that carry the addresses at which the CPU is processing at that time, such as the locations that the CPU is reading from or writing to.
- The width of the address bus corresponds to the maximum addressing capacity of the bus, or the largest address within memory that the bus can work with.
- The addresses are transferred in binary format, with each line of the address bus carrying a single binary digit. Therefore the maximum address capacity is equal to two to the power of the number of lines present (2^lines).

2) Data Bus

- This is used for the exchange of data between the processor, memory and peripherals, and is bi-directional so that it allows data flow in both directions along the wires.
- The number of wires used in the data bus (sometimes known as the 'width') can differ.
- Each wire is used for the transfer of signals corresponding to a single bit of binary data.
- As such, a greater width allows greater amounts of data to be transferred at the same time.

3) Control Bus

- The control bus carries the signals relating to the control and co-ordination of the various activities across the computer, which can be sent from the control unit within the CPU.
- Each line is used to perform a specific task. For instance, different, specific lines are used for each of read, write and reset requests.

3. MICROPROCESSOR SYSTEM WITH BUS ORGANIZATION

- To design any meaningful application microprocessor requires support of other auxiliary devices.
- In most simplified form a microprocessor based system consist of a microprocessor, I/O (input/output) devices and memory.
- These components are interfaced (connected) with microprocessor over a common communication path called system bus. Typical structure of a microprocessor based system is shown in Figure 3.1.

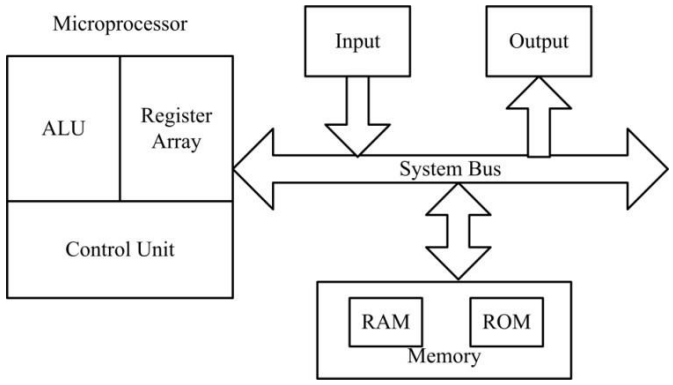
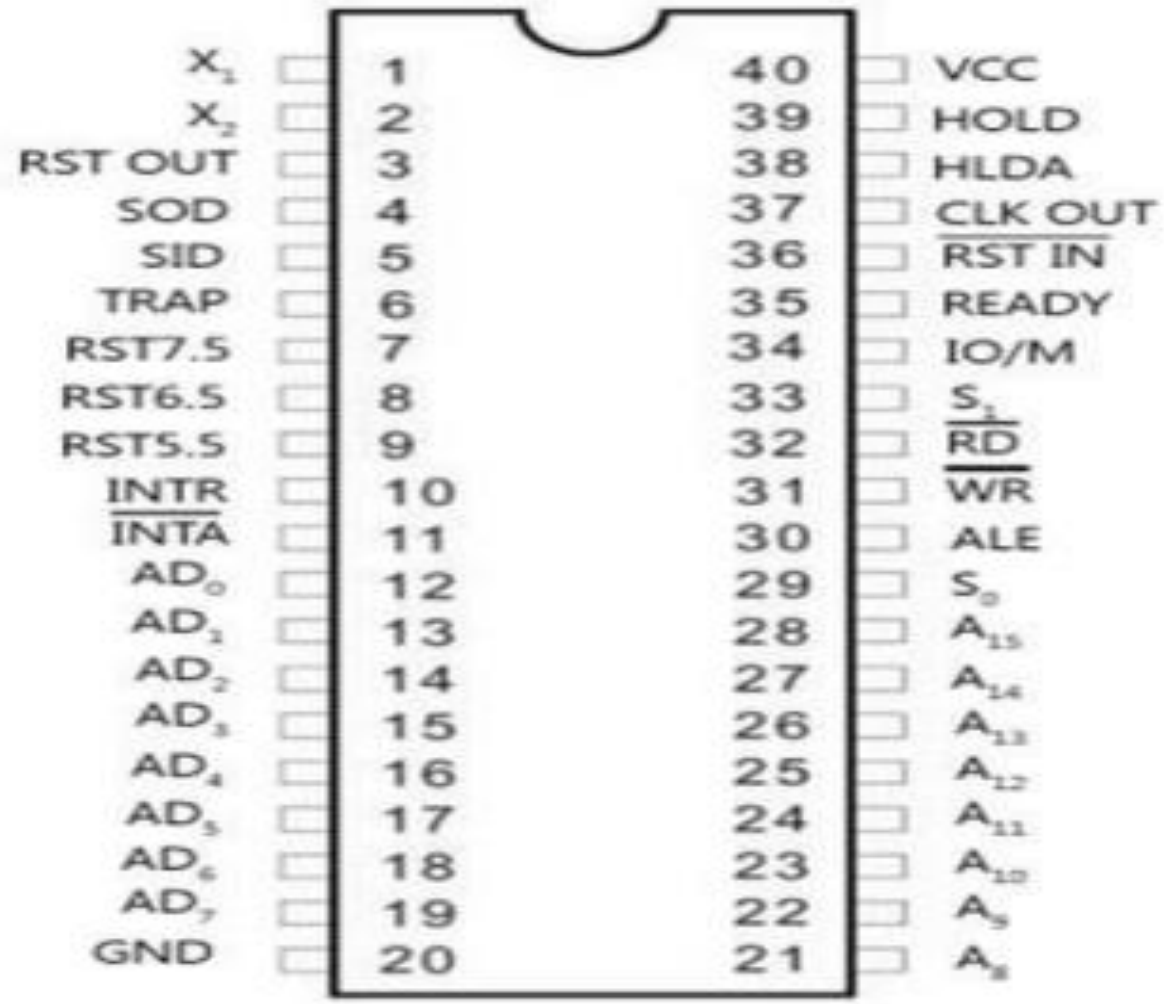


Figure 3.1: Microprocessor based system

- Here, microprocessor is master of the system and responsible for executing the program and coordinating with connected peripherals as required.
- Memory is responsible for storing program as well as data. System generally consists of two types of memories ROM (Read only and non-volatile) and RAM (Read/Write and volatile).
- I/O devices are used to communicate with the environment. Keyboard can be example of input devices and LED, LCD or monitor can be example of output device.
- Depending on the application level of sophistication varies in a microprocessor based systems. For example: Washing machine, Computer.



Pin diagram of 8085

4. ADDRESSING MODES

❖ Implied Addressing:

The addressing mode of certain instructions is implied by the instruction's function. For example, the STC (set carry flag) instruction deals only with the carry flag, the DAA (decimal adjust accumulator) instruction deals with the accumulator.

❖ Register Addressing:

Quite a large set of instructions call for register addressing. With these instructions, you must specify one of the registers A through E, H or L as well as the operation code. With these instructions, the accumulator is implied as a second operand. For example, the instruction CMP E may be interpreted as 'compare the contents of the E register with the contents of the accumulator.

Most of the instructions that use register addressing deal with8-bit values. However, a few of these instructions deal with 16-bit register pairs. For example, the PCHL instruction exchanges the contents of the program counter with the contents of the H and L registers.

❖ Immediate Addressing:

Instructions that use immediate addressing have data assembled as a part of the instruction itself. For example, the instruction CPI 41H. When this instruction is executed, the processor fetches the first instruction byte and determines that it must fetch one more byte. The processor fetches the next byte into one of its internal registers and then performs the compare operation.

Notice that the names of the immediate instructions indicate that they use immediate data. Thus, the name of an add instruction is ADD; the name of an add immediate instruction is ADI.

All but two of the immediate instructions uses the accumulator as an implied operand, as in the CPI instruction shown previously. The MVI (move immediate) instruction can move its immediate data to any of the working registers including the accumulator or to memory. Thus, the instruction MVI D, OFFH moves the hexadecimal
Value FF to the D register.

The LXI instruction (load register pair immediate) is even more unusual in that its immediate data is a 16-bit value. This instruction is commonly used to load addresses into a register pair. As mentioned previously, your program must initialize the stack pointer; LXI is the instruction most commonly used for this purpose. For example, the instruction LXI SP, 3OFFH loads the stack pointer with the hexadecimal value 30FF.

❖ Direct Addressing:

Jump instructions include a 16-bit address as part of the instruction. For example, the instruction JMP 1000H causes a jump to the hexadecimal address 1000 by replacing the current contents of the program counter with the new value 1000H.

Instructions that include a direct address require three bytes of storage: one for the instruction code, and two for the 16-bit address

❖ Register Indirect Addressing:

Register indirect instructions reference memory via a register pair. Thus, the instruction MOV M,C moves the contents of the C register into the memory address stored in the H and L register pair. The instruction LDAX B loads the accumulator with the byte of data specified by the address inthe B and C register pair.

❖ Combined Addressing Modes:

Some instructions use a combination of addressing modes. A CALL instruction, for example, combines direct addressing and register indirect addressing. The direct address in a CALL instruction specifies the address of the desired subroutine; the register indirect address is the stack pointer. The CALL instruction pushes the current contents of the program counter into the memory location specified by the stack pointer.

5. INSTRUCTION TYPES

Instruction Naming Conventions: The mnemonics assigned to the instructions are designed to indicate the function of the instruction.

❖ Data Transfer Croup:

The data transfer instructions move data between registers or between memory and registers.

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from
Memory STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from
Memory SHLD	Store H & L Registers Directly in
Memory	

An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register
Pair STAX	Store Accumulator in Address in Register
Pair XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

❖ Arithmetic Group:

The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add immediate data to Accumulator Using
Carry SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow/Carry Flag
SBI	Subtract Immediate from Accumulator Using Borrow/Carry
Flag INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of
	Register Pair to H & L Register Pair

❖ Logical Group:

This group performs logical (Boolean) operations on data in registers and memory and on condition flags.

The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate
Data ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate
Data XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

CMP	Compare
CPI	Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left through Carry
RAR	Rotate Right through Carry

Complement and carry flag instructions:

CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

❖ Branch Group:

The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

JMP	Jump
CALL	Call
RET	Return

Conditional branching instructions examine the status of one of four condition flags to determine whether the specified branch is to be executed. The conditions that may be specified are as follows:

NZ	Not Zero (Z = 0)
Z	Zero (Z = 1)
NC	No Carry (C = 0)
C	Carry (C = 1)
PO	Parity Odd (P = 0)
PE	Parity Even (P = 1)
P	Plus (S = 0)
M	Minus (S = 1)

Thus, the conditional branching instructions are specified as follows:

Jumps	Calls	Returns	
C	CC	RC	(Carry)
INC	CNC	RNC	(No Carry)
JZ	CZ	RZ	(Zero)
JNZ	CNZ	RNZ	(Not Zero)
JP	CP	RP	(Plus)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)

Two other instructions can affect a branch by replacing the contents or the program counter:

PCHL	Move H & L to Program Counter
RST	Special Restart Instruction Used with Interrupts

❖ Stack I/O, and Machine Control Instructions:

The following instructions affect the Stack and/or Stack Pointer:

PUSH	Push Two bytes of Data onto the
Stack POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

The I/0 instructions are as follows:

IN	Initiate Input Operation
OUT	Initiate Output Operation

The Machine Control instructions are as follows:

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Practical-1

Write the size of the following instructions

1. SUB B 1byte
2. INR B 1byte
3. MVI B,25H 2byte
4. LXI D, 2500H 3byte
5. MOV M, A 1byte
6. SBI 38H 2byte
7. INX H 1byte
8. DCR D 1byte
9. CMA 1byte
10. RAL 1byte
11. LHLD 3000H 1byte

Practical-2

1. Multiply 1245H by 2 .

```
LXI H, 1245H
DAD H
HLT
```

2. ADD 1245H & 1345H using DAD instruction.

```
LXI H, 1245H
LXI B, 1345H
DAD B
HLT
```

3. SUBTRACT 1245H & 1345H using SBB instruction.

```
LXI H, 1245H
LXI B, 1345H
SBB B
HLT
```

4. WRITE A PROGRAM to find one's complement of an 8-bit number present in memory location 2501H and store the result in 2502H.

```
LDA 2501H
CMA
STA 2502H
HLT
```

5. WRITE A PROGRAM to complement the higher nibble without affecting the lower nibble.

```
LDA 2504H
XRI F0H
STA 2505H
HLT
```

Practical-3

Write a program to add two numbers present in memory locations 2000H and 2001H and add 25H to the result and store the final result in memory location 4000H.

Address	Data
2000H	20H
2001H	30H
…	
…	
4000H	……..

Solution:

```
LDA 2000H : Load Acc with the content of mem location 2000H (A = 20H)
MOV B,A (B=A=20H)
LDA 2001H (A=30H)
ADD B (A+B) AND RESULT WILL BE STORED IN ACC
ADI, 25H (A+25)
STA 4000H ( 4000H= A)
HLT
```

Practical-4

Write an assembly language program to perform addition of two hexadecimal numbers (99H and 98H) by immediate addressing mode. Store the result and carry in memory locations 2503H and 2504H respectively

Memory Address	Label	Mnemonics/ Opcodes	Comments
2000H		**MVI C, 00H**	Clear Register C
2002H		**MVI A, 99H**	Move 99H in Accumulator
2004H		**MVI B, 98H**	Move 98H in Register B
2006H		**ADD B**	Add the content of A and B and store the result in A
2007H		**JNC AHEAD**	Jump to AHEAD if CY = 0
200AH		**INRC**	If CY=1, increment Reg C by 1, C=1
200BH	**AHEAD:**	**STA 2503H**	Store the result (Accum) in 2503H
200EH		**MOV A,C**	Move content of C (i.e. Carry) in Accum.
200FH		**STA 2504H**	Store the Carry (Accum) in 2504H
2012H		**HLT**	End the program

Practical-5

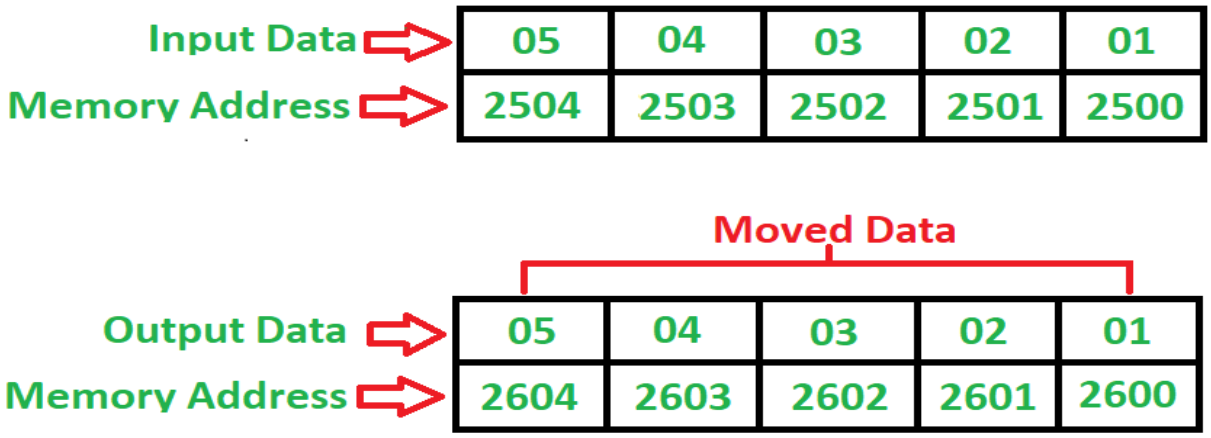
WRITE A PROGRAM to find larger of two numbers and store it in 2503H Assume the first no. 98H is placed in the memory location 2501H And the second no. 87H is placed in the memory location 2502H

Address	Label	Mnemonics & Operands	Comments
2000H		LXI H 2501	Address of 1st no in HL pair
2003H		MOV A,M	1st no in accumulator
2004H		INX H	Address of 2nd no in HL pair
2005H		CMP M	Compare 2nd no with 1st no (Is A> M)
2006H		JNC AHEAD	Yes, large no is in accumulator, goto ahead
2009H		MOV A,M	No, get second no in accumulator
200AH	AHEAD:	STA 2503H	Store large no in 2503H
200DH		HLT	stop

Practical-6

Write a program to move blocks of bits from source location starting at 2500 to destination location starting from 2600 where size of blocks is 05 bytes.

Output:-



Algorithm –

1. Load register pair H-L with the address 2500H
2. Load register pair D-E with the address 2600H
3. Move the content at memory location into accumulator
4. Store the content of accumulator into memory pointed by D-E
5. Increment value of register pair H-L and D-E by 1
6. Decrements value of register C by 1
7. If zero flag not equal to 1, go to step 3
8. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MVI	C, 05	[C] <- 05
2002	LXI	H, 2500	[H-L] <- 2500
2005	LXI	D, 2600	[D-E] <- 2600
2008	MOV	A, M	[A] <- [[H-L]]
2009	STAX	D	[A] -> [[D-E]]
200A	INX	H	[H-L] <- [H-L] + 1
200B	INX	D	[D-E] <- [D-E] + 1
200C	DCR	C	[C] <- [C] – 1
200D	JNZ	2008	Jump if not zero to 2008
2010	HLT		Stop

Explanation – Registers A, D, E, H, L, C are used for general purpose:

1. **MOV** is used to transfer the data from memory to accumulator (1 Byte)
2. **LXI** is used to load register pair immediately using 16-bit address (3 Byte instruction)
3. **MVI** is used to move data immediately into any of registers (2 Byte)
4. **STAX** is used to store accumulator into register pair indirectly (3 Byte instruction)
5. **DCR** is used to decrease register by 1 (1 Byte instruction)
6. **INX** is used to increase register pair by 1 (1 Byte instruction)
7. **JNZ** is used to jump if not zero to given memory location (3 Byte instruction)
8. **HLT** is used to halt the program.