Chapter 3
Instruction Set and addressing modes

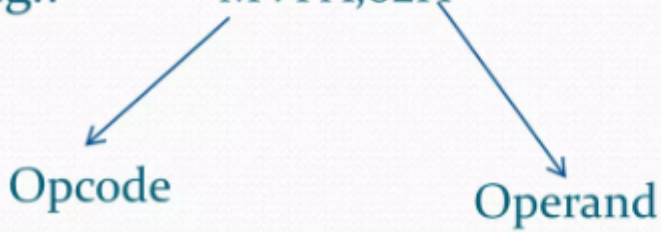
8085 Instruction

- Since the 8085 is an 8-bit device it can have up to 28 (256) instructions.
  - However, the 8085 only uses 246 combinations that represent a total of 74 instructions.
  - Most of the instructions have more than one format.

- These instructions can be grouped into five different groups
  - Data Transfer Operations
  - Arithmetic Operations
  - Logic Operations
  - Branch Operations
  - Machine Control Operations

Instruction Formats

- Each instruction has two parts.
- **Opcode (operation code)**- The first part is the task or operation to be performed.
- **Operand** - The second part is the data to be operated on. Data can be given in various form.
  - It can specify in various ways: may include 8 bit/16 bit data, an internal register, memory location or 8 bit /16 bit address.
- For Eg.: MVI A,02H



Instruction word Size

- An instruction is assembled in binary form(0,1) known as machine code or opcode. Due to different ways of specifying data or operand the machine code are not same for all the instruction
- The size of an instruction signifies how much memory space is required to load an instruction in the memory.8085 instructions are of following sizes:

✓**One-byte or one word Instructions:** opcode and operand in 8 bits only i.e. one byte. Operand(s) are internal register and are coded into the instruction.

e.g. MOV, ADD, ANA, SUB, ORA etc.

MOV A,B Move the content of B in A

01 111 000 = 78H 01 for MOV operation, 111 binary code for register A and last 000 binary code for register B

Task	Op- code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

Instruction Format or Size....Cont...

✓ **Two-byte instructions:** first byte is opcode in 8 bits and second byte is operand either 8 bit data or 8 bit address.

e.g. MVI, ADI, ANI, ORI, XRI etc.

MVIB, 05H Move 05 in register B

06,05 MVIB, 05H in the code for m

Task	Op- code	Operand	Binary Code	Hex Code
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110 DATA	3E (First Byte) Data(Second Byte)

Mnemonics	Hex code
MVI A, 32H	3E 32H

Instruction Format or Size....Cont...

✓ **Three-byte instructions:** first byte is opcode in 8 bits and second and third byte are operand either 16 bit data or 16 bit address.

Operand 1 = lower 8 bit data/address

Operand 2 = Higher 8 bit data/address

opcode + data byte + data byte

e.g. LXI, LDA, STA, LHLD, SHLD etc.

LXIH , 2500H — load HL pair with 2500H

21,00,25 — LXIH , 2500H in code form

Task	Op-code	Operand	Binary Code	Hex Code	Hex Code
Transfer the program sequence to the memory Location 2085H.	JMP	2085H	1100 0011 1000 0101 0010 0000	C3 85 20	First Byte Second Byte Third Byte

Opcode Format

✓ Opcode contains the information regarding about operation, register used, memory to be used etc.

✓ Fixed for each instruction.

Register	Code
B	000
C	001
D	010
E	011
H	100
L	101
M	110
A	111

Register pair	Code
BC	00
DE	01
HL	10
A, F, SP	11

Addressing Modes

The various formats for specifying operands/data are called the ADDRESSING MODES. 8085 instructions can be classified in following addressing modes

❑ Register Addressing mode

- Data is provided through the registers .e.g. MOV, ADD, SUB, ANA, ORA, XRA etc.

For Ex.	MOV A, B	move the content of register B to register A.
	78H	opcode of MOV A,B

❑ Immediate Addressing mode

- Data is present in the instruction. Load the immediate data to the destination provided. e.g. MVI, LXI, ADI, SUI, ANI, ORI etc.

For Ex.	MVI A, 05H	move the 05H in register A.
	3E, 05	opcode of MVI A, 05H

❑ Direct Addressing mode

- Instructions have their operands in memory and the 16-bit memory address is specified in the instruction
- The address of the operand is given in the instruction. e.g. LDA, STA, LHLD, SHLD etc.

For Ex.

STA 7500	store the content of accumulator in the memory location 7500 H
32 , 00, 75	opcode of STA 7500
IN 01	Read data from port 01.
DB,01	opcode of IN 01

Addressing Modes.....Cont...

❑ Indirect Addressing mode

- In this address of the operand is specified by the register pair. The address stored in the register pair points to memory location e.g. LDAX, STAX, PUSH, POP etc.

LXI H, 7500	load HL pair with 7500H
MOV A, M	move the content of memory location, whose address is in HL pair to the accumulator

❑ Implicit or Implied Addressing mode

- The operand is not specified in the instruction, specified within the opcode itself.
- Operand is supposed to be present in accumulator. e.g. CMA, CMC, STC etc.

RAL	Rotate the content of accumulator towards left.

Instruction Set

- ✓ Since the 8085 is an 8-bit device it can have up to 28 (256) instructions.
- ✓ However, the 8085 only uses 246 combinations that represent a total of 74 instructions.
- ✓ Most of the instructions have more than one format.

These instructions can be grouped into five different groups:

- ➢ Data Transfer Operations
- ➢ Arithmetic Operations
- ➢ Logic Operations
- ➢ Branch Operations
- ➢ Machine Control Operations

Instruction Set.....cont....

(A) Data Transfer operation

- These operations simply **COPY** the data from the source to the destination.
- They transfer:
  - Data between registers.
  - Data Byte to a register or memory location.
  - Data between a memory location and a register.
  - Data between an I/O Device and the accumulator.
- The data in the source is not changed.
- Data transfer instructions never affect the flag bits.

  e.g. LDA, STA, MOV, LDAX, STAX, MVI, LXI etc.

Instruction Set.....cont....

(B) Arithmetic Operation

- These instruction perform addition, subtraction and compare operations.
- These operations are always performed with accumulator as one of the operands.
- The status of the result can be verified by the contents of the flag register.

Addition: Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. The instruction DAD is an exception; it adds 16-bit data directly in register pairs. Ex ADD, ADI

Subtraction - Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. Subtraction is done by 2's compliment method and set carry flag to indicate borrow. SUB. SBI

Increment/Decrement - The 8-bit contents of a register or a memory location can be incremented or decrement by 1. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decrement by 1. INR,DCR.

Instruction Set... Cont....

(C) Logical Operation

- Perform 8-bit basic logical operations with the content of the accumulator
- Logical instructions also modify the flag bits.
- Op-codes for logical instructions include ANA, ANI, ORA, ORI, XRA, XRI, CMA, CMC, RAL, RLC, RAR, RRC, CMP, CPI etc.

AND, OR Exclusive-OR - Any 8-bit number, or the contents of a register, or of a memory location can be logically AND, Or, or Exclusive-OR with the contents of the accumulator. The results are stored in the accumulator.

Rotate- Each bit in the accumulator can be shifted either left or right to the next position.

Compare- Any 8-bit number, or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement - The contents of the accumulator can be complemented.

Instruction Set... Cont....

(D) Branch Operation

These instructions are used to transfer the program control:

- To jump from one memory location to any other memory location within a program
- From one program to another program called as a subroutine.
- Alters the sequence of program execution either conditionally or unconditionally

- **Unconditional branch instructions-** Transfer the program to the specified label or address JMP unconditionally i.e. without satisfying any condition.

  Unconditional Program control instructions are

  - Call & RET

- **Conditional branch instructions -**Transfer the program to the specified label or address when certain condition is satisfied.
  - JNC, JC, JNZ, JZ, JP, JM, JPE, JPO
  - CNC, CC, CNZ, CZ, CP, CM, CPE, CPO
  - RNC, RC, RNZ, RZ, RP, RM, RPE, RPO

Instruction Set... Cont....

(E) Machine Control Operation

These instructions include special instructions such as I/O data transfer, perform machine related operation

- HLT – To halt the CPU
- NOP – To perform no operation
- SIM – To set the masking of hardware interrupts and serial output data
- RIM – To read the status of interrupt mask and serial input data
- EI – Enable Interrupt
- DI – Disable Interrupt

How to Write the Assemble Program

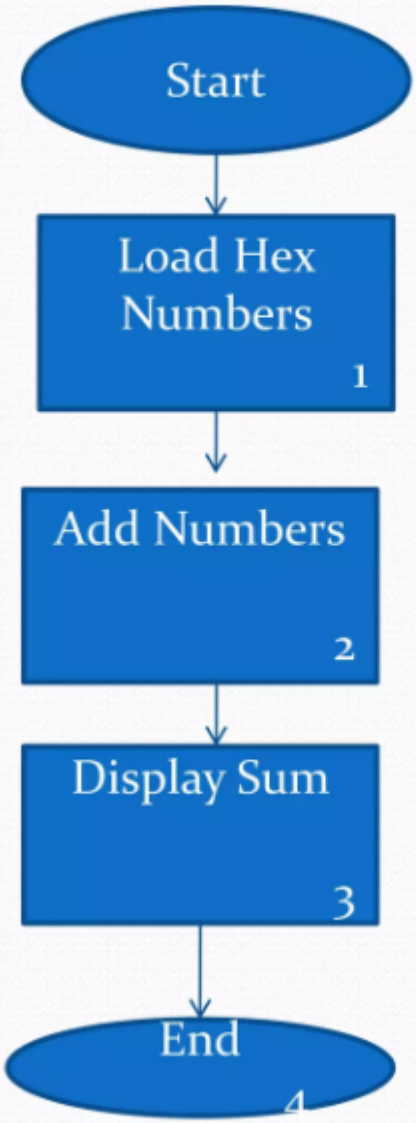
Write the instructions to load two hexadecimal number 32H and 48H in registers A and B respectively. Add two numbers and display the sum at the LED output port PORT 1.

Divide the program into small steps.

- Load the numbers in register
- Add the number
- Display the sum at the LED output port PORT 1.

The steps listed in the problem, the sequence can be represented in the block diagram, called flow chart

Write the Assemble Program..cont..



Block 1: MVI A, 32H — Load A with 32 H
MVI B, 48H — Load B with 48 H

Block 2: ADD B — Add two bytes and save sum in A

Block 3: OUT 01H — Display accumulator content at port 01H

Block 4: HALT — End

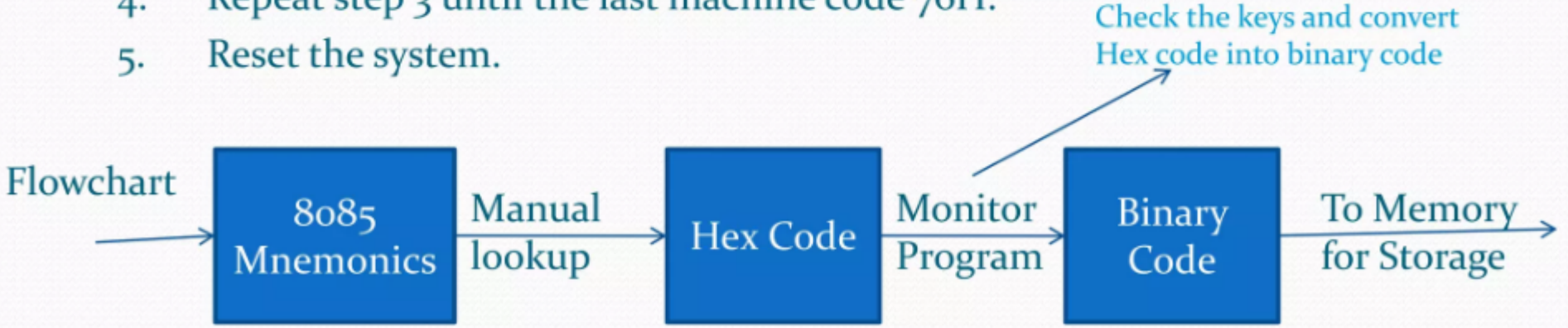
Write the Assemble Program..cont..

Mnemonics	Hex Code	
MVI A, 32H	3E 32	2 Byte Instruction
MVI B, 48H	06 48	2 Byte Instruction
ADD B	80	1 Byte Instruction
OUT 01H	D3 01	2 Byte Instruction
HALT	76	1 Byte Instruction

Storing in Memory and converting from Hex code to binary Code

To store the program in R/W memory, assume the memory ranges from 2000 H to 20FFH. Now to enter the Program

1. Reset the system by pushing the RESET key.
2. Enter the first memory address using the HEX keys, where the program should be written.(2000H)
3. Enter each machine code by pushing Hex keys. For Ex, to enter the first machine code, push 3E and store keys. Store key, store the program in memory location 2000H and upgrade the memory address to 2001.
4. Repeat step 3 until the last machine code 76H.
5. Reset the system.



Program Stored in memory as

Mnemonics	Hex Code	Memory Content	Memory Address
MVI A, 32H	3E	0011 1110	2000
	32	0011 0010	2001
MVI B, 48H	06	0000 0110	2002
	48	0100 1000	2003
ADD B	80	1000 0000	2004
OUT 01H	D3	1101 0011	2005
	01	0000 0001	2006
HALT	76	0111 1110	2007

Eight Machine code, require eight memory location. MPU can only understand binary code; everything else(Mnemonics, Hex code, comments) for the convenience of human being.

Executing the Program

To execute the program, we need to tell the MPU where the program begins by entering the memory address 2000H.

- Push the execute Key, so MPU loads 2000H in PC.
- PC is transferred from the monitor program to our program.
- MPU begin to read the one machine cycle at a time, when it fetches the complete instruction, it execute that instruction.
- For Eg. When it fetches the machine code stored in 2000 and 2001H and execute the instruction MVI A, 32H , thus load 32H in register A.
- Continue to execute the instruction until it fetches the HLT instruction.