Module 2
Instruction set of Intel 8085

4.1:

- An instruction is a command given to the computer to perform specified operation on given data.
- The instruction set of a microprocessor is the collection of the instructions that the microprocessor is designed to execute.
- Instruction has been classified into following groups:

1. Data transfer group
2. Arithmetic group
3. Logical group
4. Branch control group
5. I/O and Machine Control group

1.Data transfer group

-Instructions which are used to transfer data from one point to another point

(copy data)

ie from m/y to reg, reg to m/y, reg to reg,

Ex:Mov A,B

LDA 2500

2.Arithmetic group

The instructions of this gp perform arithmetic operation such as addition,subtraction,increment/decrement of the contents of reg/memory

Ex:ADD ,SUB,INR,DAD

3.Logical group

Instructions under this gp perform logical operation such as AND,OR,XOR;compare ,rotate

Ex: ANA,XRA,ORA,CMP,RAL

4.Branch control group

-decide pgm control

-includes the instructions for conditional and unconditional jump,subroutine call and return,and restart

EX:JMP,JC,JZ,CALL,CZ,RST

5.I/O and Machine control group

-special instructions

-contain I/O &machine related content

-includes the instructions for input/output ports,stack and machine control

Ex:IN,OUT,PUSH,POP,HLT

4.2: Instruction and Data Formats

-Intel 8085 is an 8bit microprocessor.

-it handles 8 bit data at a time

1 byte-8bit

-intel 8085 designed to accommodate 8 bit data.

-if 16 bit data are to be stored, they are stored in consecutive m/y location

-address of a m/y location 16bits ie 2 bytes

- Operand format(data format)

1.Direct data

8 bit /16 bit data directly given in instruction itself

ADI 06 ….8 bit data

LXI H,2500 … 16 bit data

2)Address given in instruction

Address of m/y location or I/O device , where the data resides

IN 02….8 bit address

STA 2400 -----16 bit address

3)One reg specified

EX:ADD B

4)2 reg specified

EX:MOV A,B

5)Data is implied

It operate on content of accumulator

EX:CMA

RAL

RAR

- 8085 instruction set classified according to the size of instructions
- Instructions divided into

1)1 byte instructions

8 bit of data

Include opcode and operand(reg/variable in same byte)

ADD B

Mov M

2)2 byte instruction

First byte-opcode

2nd byte-operand

MVI A,08

ADI 05

1st byte (opcode), 2nd byte(operand)

According to how many bits in operand, either 2 byte/3 byte instruction

3)3 byte instruction

1st byte-opcode

2nd&3rd byte-operand

STA 4600

1st byte(opcode) , 2nd byte(operand's MSB 46), 3rd byte(operand's LSB 00)

LXI 2500

1st byte(opcode) , 2nd byte(operand's MSB 25), 3rd byte(operand's LSB 00)

4.3: ADDRESSING MODES OF 8085

- There are various techniques to specify data(operand)for instructions. These techniques are called addressing modes.
- 5 types of addressing mode

1. Direct Addressing
2. Register Addressing
3. Register Indirect Addressing
4. Immediate Addressing
5. Implicit Addressing

1.Direct Addressing

-address of the operand(data)is given in the instruction itself

[address in instruction]

-it specifies 16 bit address of the operand within the instruction itself.

Ex:

1)LDA 2000H, data is directly copied from the given address to register(Accumulator register)

2)STA 2400H,store the content of the accumulator in the memory location 2400H

2.Register Addressing

-operands are in the general purpose registers

-data transfer between register.

[data in register]

Ex:

1)MOV A,B, Move the contents of reg B into reg A

2)ADD B, Add contents of Register A and register B and store the result in Register A

3.Register Indirect Addressing

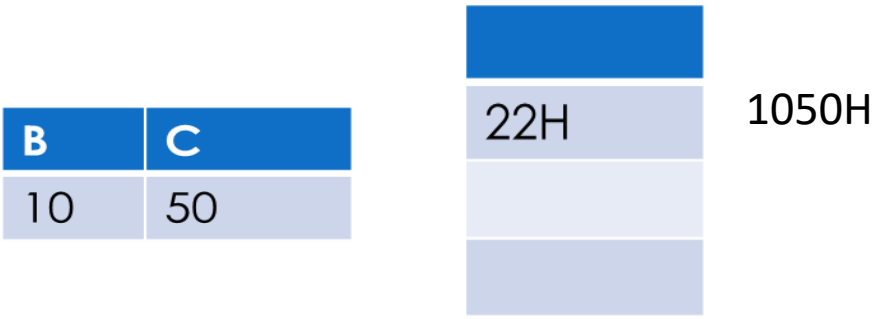
-address of operand is specified by Register pair.

[address in register]

Ex:

1)Mov A,M ,move the content of the memory location,whose address is in H-L pair,to the accumulator.

2)LDAX B



4.Immediate Addressing

-operand(data) is specified within instruction itself.

-ie,8/16 bit data can be specified in instruction as one of its operand.

-we directly transfer data into register.

Ex:

1)MVI A,05, Move 05 to register A

2)ADI 06,Add 06 to content of Accumulator

5.Implicit Addressing

-operate on content of Accumulator.

-do not require address of the operand.

EX:

CMA, complement Accumulator contents

RAL, rotate accumulator content left by 1 position

RAR, rotate accumulator content right by 1 position\

Summary(Addressing mode):

Addressing Mode:manner in which operand given in instruction

1)immediate..data in instruction

2)register….data in reg

3)direct…address in instruction

4)indirect…address in reg

5)implied

4.4. Status flags

Refer module 1.no need to write it again

4.6.Instruction set of Intel 8085(in detail)

I.Data Transfer Group

1.**MOV r1,r2**(Move data:Move the content of one register to another)

[r1]<-[r2]

The content of register r2 moved to register r1.

-No flags affected

Ex:Move A,B, moves the content of register B to register A

MOV r,M(Move the content of memory to register)

[r]<-[[H-L]]

-The content of memory location,whose address is in H-L pair ,is moved to register r

-No flags affected

MOV M,r(Move the content of register to memory)

[[H-L]]<-[r]

-The content of register r is moved to the memory location addressed by H-L pair

-No flags affected.

EX: MOV M,C, Move the content register C to the memory location whose address is in H-L pair.

2.MVI r,data(Move immediate data to register)

[r]<-data

-The data given in instruction moved to register r

-No flags affected.

Ex: MVI A,05, Move 05 to register A

MVI M,data

[[H-L]]<-data

-The data is moved to the memory location whose address is in H-L pair.

-No flags affected.

3.LXI rp,data16 (Load register pair immediate)

[rp]<-data 16 bits, [rh]<-8 MSB ,[rl]<-8 LSB of data.

-This instruction loads 16 bit immediate data into register pair rp.

-In register pair,only high-order register is mentioned after the instruction.

Ex: LXI H,2500H, loads 2500H into H-L pair

H:25 L:00

4.LDA addr(Load Accumulator Direct)

[A]<-[addr]

-The content of the memory location ,whose address is specified in the instruction, is loaded into the accumulator

Ex: LDA 2400H ,load the content of memory location 2400H into the accumulator.

5.STA addr (Store accumulator direct)

[addr]<-[A]

-The content of the accumulator is stored in the memory location whose address is specified in the instruction

EX:STA 2000H, store content of accumulator in the memory location 2000H

II. Arithmetic Group

1.**ADD r**(Add register to accumulator)

[A]<-[A]+[r]

-The content of register r is added to the content of the accumulator, and the sum is placed in the accumulator.

-All flags affected.

ADD M(Add Memory to accumulator)

[A]<-[A]+[[H-L]]

-The content of memory location addressed by H-L pair is added to the content of the accumulator. The sum is placed in the accumulator.

-All flags affected.

2. **ADC r**(Add register with carry to accumulator)

[A]<-[A]+[r]+[CS]

-The contents of register r and carry status are added to the content of the accumulator. The sum is placed in the accumulator.

-All flags affected.

ADC M

[A]<-[A]+[[H-L]]+[CS]

The content of memory location addressed by H-L pair and carry status are added to the content of the accumulator. The sum is placed in the accumulator.

-All flags affected.

3**.ADI data**(Add Immediate data to accumulator)

[A]<-[A]+data

- The immediate data is added to the content of the accumulator .The sum is placed in the accumulator
- EX:ADI 08, add 08 to the content of accumulator and place the result in the accumulator.
- All flags affected.

4. **ACI data**(Add with carry immediate data to accumulator)

[A]<-[A]+data+[CS]

-data given in the instruction and carry status added to the content of accumulator. The sum is placed in the accumulator.

-All flags affected.

5.**DAD rp**(Add register pair to H-L pair)

[H-L]<-[H-L]+[rp]

-The contents of register pair rp are added to the contents of H-L pair and the result is placed in H-L pair.

-only carry flag(CS) is affected

III. Logical Group:

The instructions of this group perform AND ,OR, EXCLUSIVE OR operations;Compare,rotate or take complement of data in register or memory.

1.ANA r(AND register with accumulator)

[A]<-[A]^[r]

-the content of register r is ANDed with the content of accumulator,and the result is placed in the accumulator.

-All status flags affected.

ANA M(AND memory with accumulator)

[A]<-[A]^[[H-L]]

-The content of memory location addressed by H-L pair is ANDed with the accumulator .The result is placed in the accumulator.

-All flags affected

Cs-0

AC-1

ANI data(AND immediate data with accumulator)

[A]<-[A]^data

-the data given in the instruction is ANDed with the content of accumulator .The result is placed in accumulator.

-All flags affected

CS:0

AC:1

3.ORA r(OR register with accumulator)

[A]<-[A]v[r]

-The content of register r is ORed with the content of accumulator.The result is placed in the accumulator.

-All flags affected

-CS & AC:0

ORA M(OR memory with accumulator)

[A]<-[A]v[[H-L]]

-The content of the memory location addressed by H-L pair is ORed with the content of accumulator.The result is placed in the accumulator.

-All flags affected

-CS & AC:0

4.ORI data(OR immediate data with accumulator)

[a]<-[A] v data

-The data given in the instruction is ORed with the content of the accumulator.The result is placed in the accumulator.

-All flags affected

CS & AC=0

5.XRA r(EXCLUSIVE –OR register with accumulator)

[A]<-[A] v [r]

-The content of register r is EXCLUSIVE ORed with the content of accumulator.The result is placed in the accumulator.

-All flags affected

CS & AC=0

IV. Branch group

- The instructions of this group change the normal sequence of the program.
- There are two types of branch instructions: Conditional and Unconditional.
- The conditional branch instructions transfer the program to the specified label when certain condition is satisfied.

- The unconditional branch instructions transfer the program to the specified label unconditionally.

1.JMP address(label) (Unconditional jump:jump to the instruction specified by the operand)
[PC]<-Label

-No flags affected

- The address of the label is the address of memory location for next instruction to be executed. The program jumps to the instruction specified by the address(label)unconditionally.
- **Conditional jump addr(label)**

-After the execution of a conditional jump instruction the program jumps to the instruction specified by the address(label)if the specified condition is fulfilled.

-The program proceeds further in the normal sequence if the specified condition is not fulfilled.

1.JZ addr(label)(Jump if the result is zero)

[PC]<-address(label),jump if z=1

-no flags affected

-The program jumps to the instruction specified by the address(label)if the result is zero(zero status Z=1).

-Here the result after the execution of preceding instruction is under consideration

-No flags affected

2.JNZ addr(label) (Jump if the result is not zero)

[Pc]<-address(label), Jump if z=0

-No flags affected.

-The program jumps to the instruction specified by the address (label)if the result is non-Zero(ie Zero status Z=0)

3.JC addr(label)(jump if there is a carry)

[PC]<-address(label),jump if CS=1

-No flags affected.

-The program jumps to the instruction specified by the address(label)if there is a carry(ie carry status CS=1)

-Here the carry after the execution of the preceding instruction is under consideration

4.JNC address(label) (Jump if there is no carry)

[PC]<-address(label), jump if CS=0

-No flags affected

-The program jumps to the instruction specified by the address(label)if there is no carry(ie carry status CS=0)

5.JP address(label) (Jump if the result is plus)

[PC]<-address(label), jump if S=0

-No flags affected.

-The program jumps to the instruction specified by the address (label) if the result is plus.

5.Stack,I/o and machine control group

1. **IN port address**(input to accumulator from i/o port)

[A]<-[port]

No flags affected

2.OUT port address(output from accumulator to i/o port)

[port]<-[A]

No flags affected

3.PUSH(push the content of register pair to stack)

[[sp]-1]<-[rh]

[[sp]-2]<-[rl]

[sp]<-[sp]-2

No flags affected

PUSH PSW(push processor status word)

[[sp]-1]<-[A]

[[sp]-2<-PSW

[sp]<-[sp]-2

No flags affected

Ex:

[A]=33H flag reg=25H [sp]=D015

PUSH PSW

[D014]=33H [D013]=25H

[SP]=D013H

4.POP rp(pop the content of reg pair ,which was saved,from stack)

[rl]<-[[SP]]

[rh]<-[[SP]+1]

[sp]<-[sp]+2

No flags affected

POP PSW(pop processor status word)

PSW<-[[sp]]

[A]<-[[SP]+1]

[sp]<-[sp]+2

No flags affected

5.HLT(halt)

Stops the microprocessor

Registers &status flags unaffected

****