SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

UNIT – II – 8085 Instruction Set and Assembly Language Programming - SECA1404

II. 8085 Instruction Set and Assembly Language Programming

An instruction is a binary pattern designed inside a microprocessor to perform a specific function.

- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instruction

These instructions move data between registers, or between memory and registers.

These instructions copy data from source todestination.

While copying, the contents of source are not modified.

Opcode	Operand	Description
MOV	Rd, Rs Rd, M M, Rs	Copy from source to destination.

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 specifiedby the contents of the HL registers.

Example: MOV B, C

- MOV B, M MOV M, C

Data Transfer Instruction

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- 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 H-Registers.

Example: MVI A, 57H MVI M, 57H

Data Transfer Instruction

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.

- **Example:** LXI H, 2034 H

Data Transfer Instruction

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- 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

Data Transfer Instruction

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory locationinto the accumulator.
- The contents of either the register pair or the memory location are not altered.
- Example: LDAX B

Data Transfer Instruction

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.

- Example: LHLD 2040 H

Data Transfer Instruction

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are copied into the memory location specified by the operand.
- Example: STA 2500 H

Data Transfer Instruction

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.
- Example: STAX B

Data Transfer Instruction

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored intomemory location specified by the 16-

bitaddress.

- The contents of register H are stored into the next memory location.
- Example: SHLD 2550 H

Data Transfer Instruction

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- Example: XCHG

Arithmetic Instructions

These instructions perform the operations like:

- Addition
- Subtract
- Increment
- Decrement

Addition

Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.

- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- **Example:** The contents of register B cannot be added directly to the contents of register C.

Subtract

Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.

- The result is stored in the accumulator.
- Subtraction is performed in 2's complement form.
- If the result is negative, it is stored in 2's complement form.
- No two other 8-bit registers can be subtracted directly.

Increment/Decrement

The 8-bit contents of a register or a memory location can be incremented or decremented by 1.

- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Arithmetic Instructions

Opcode	Operand	Description
ADD	R, M	Add register or memory to accumulator

- The contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADD B or ADD M

Arithmetic Instructions

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADC B or ADC M

Arithmetic Instructions

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit Arithmetic Instructions
- Data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- Example: ADI 45 H

Arithmetic Instructions

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- Example: ACI 45 H

Arithmetic Instructions

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY isset. No other flags are changed.
- Example: DAD B

Arithmetic Instructions

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

Arithmetic Instructions

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair. All flags are modified to reflect the result of subtraction.
- Example: SBB B or SBB M

Arithmetic Instructions

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the accumulator. The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- Example: SUI 45 H

Arithmetic Instructions

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result ofsubtraction.
- Example: SBI 45 H

Arithmetic Instructions

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- Example: INR B or INR M

Arithmetic Instructions

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.

- The result is stored in the same place.
- Example: INX H

Arithmetic Instructions

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- Example: DCR B or DCR M

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- Example: DCX H

LOGICAL INSTRUCTIONS

These instructions perform logical operations on data stored in registers, memory and status flags.

The logical operations are:

- AND
- OR
- XOR
- Rotate
- Compare
- Complement

AND, OR, XOR

Any 8-bit data, or the contents of register, or memory location can logically have

- AND operation
- OR operation
- XOR operation with the contents of accumulator.
- The result is stored in accumulator.

ROTATE

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

COMPARE

Any 8-bit data, or the contents of register, or memory location can be compares for:

- Equality
- Greater Than
- Less Than with the contents of accumulator.
- The result is reflected in status flags.

COMPLEMENT

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by0.

LOGICAL INSTRUCTION

Opcode	**Operand**	**Description**
CMP	R M	Compare register or memory with accumulator

- 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:

Opcode	Operand	Description
CMP	R, M	Compare register or memory with accumulator

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

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

- if (A) > (reg/mem): carry and zero flags arereset.
- **Example:** CMP B or CMP M

LOGICAL INSTRUCTION

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of 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 isset
- if (A) = data: zero flag isset
- if (A) > data: carry and zero flags arereset

- **Example:** CPI 89H

LOGICAL INSTRUCTION

Opcode	Operand	Description
XRA	R M	Exclusive OR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
- **Example:** XRA B or XRA M.

LOGICAL INSTRUCTION

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically OR ed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result. CY and AC are reset.
- **Example:** ORA B or ORA M.

LOGICAL INSTRUCTION

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically Red with the8- bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 86H.

LOGICAL INSTRUCTION

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified bythe contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

LOGICAL INSTRUCTION

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

 The contentsof the accumulator are XORed with the 8-bit data. The result is placed in theaccumulator.

 S, Z, P are modified to reflect theresult. CY and AC arereset.

 Example: XRI 86H.

LOGICAL INSTRUCTION

Opcode	Operand	Description
RLC	None	Rotate accumulator left

 Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.

 S, Z, P,AC are not affected.

 Example: RLC.

LOGICAL INSTRUCTION

Opcode	Operand	Description
RRC	None	Rotate accumulator right

 Each binary bit of the accumulator is rotated right byone position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.

 S, Z, P,AC are not affected.

 Example: RRC.

LOGICAL INSTRUCTION

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.
- S, Z, P,AC are not affected.
- **Example:** RAL.

LOGICAL INSTRUCTION

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right byone position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0. S, Z, P,AC are not affected.
- **Example:** RAR.

LOGICAL INSTRUCTION

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contentsof the accumulator are complemented.
- No flags areaffected.
- **Example:** CMA.

LOGICAL INSTRUCTION

Opcode	Operand	Description
CMC	None	Complement carry

 The Carry flag iscomplemented. No other flags areaffected.

 Example: CMC.

LOGICAL INSTRUCTION

Opcode	Operand	Description
STC	None	Set carry

 The Carry flag isset to1. No other flags areaffected. **Example:** STC.

BRANCH INSTRUCTIONS

The branching instruction alter the normal sequential flow. These instructions alter either unconditionally or conditionally

BRANCH INSTRUCTIONS

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

 The program sequence is transferred to thememory location specified by the 16-bit address given in the operand.

 Example: JMP 2034 H.

BRANCH INSTRUCTIONS

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to thememory location specified by the 16-bit address given in the operand based on the specified flag of thePSW.
- **Example:** JZ 2034 H.
- The program sequence is transferred to thememory location specified by the 16-bit address given in the operand based on the specified flag of thePSW.
- Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto thestack.
- **Example:** CZ 2034 H.

JUMP CONDITIONALLY

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0

JUMP UNCONDITIONALLY

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- The program sequence is transferred to thememory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushedonto the stack.
- **Example:** CALL 2034 H.

RETURN UNCONDITIONALLY

Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred fromthe subroutine to the callingprogram.
- The two bytes from the top of the stack are copied into the program counter, and program execution beginsat the newaddress.
- **Example:** RET.

RETURN CONDITIONALLY

Opcode	Operand	Description
Rx	None	Call conditionally

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
- The two bytes from the top of the stack are copied into the program counter, and program execution beginsat the newaddress.
- **Example:** RZ.

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1

RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

Opcode	**Operand**	**Description**
RST	0 – 7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon thenumber.
- These are used as software instructions in a program to transfer program execution to one of the eight locations.
- **Example:** RST 3.

Opcode	**Operand**	**Description**
RST	0 – 7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon thenumber.
- These are used as software instructions in a program to transfer program execution to one of the eight locations.
- **Example:** RST 3.

RESRART ADDRESSES

Instructions	Restart Address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H

CONTROL INSTRUCTIONS

The control instructions control the operation of microprocessor.

Opcode	Operand	Description
NOP	None	No operation

- No operation isperformed.
- The instruction is fetched and decoded butno operation is executed.
- **Example:** NOP

CONTROL INSTRUCTIONS

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing thecurrent instruction and halts any further execution.
- An interrupt or reset is necessary to exit from the halt state.
- **Example:** HLT

CONTROL INSTRUCTIONS

Opcode	Operand	Description
DI	None	Disable interrupt

- The interrupt enable flip-flop is reset and all the interrupts except the TRAP aredisabled.
- No flags areaffected.
- **Example:** DI

CONTROL INSTRUCTIONS

Opcode	Operand	Description
EI	None	Enable interrupt

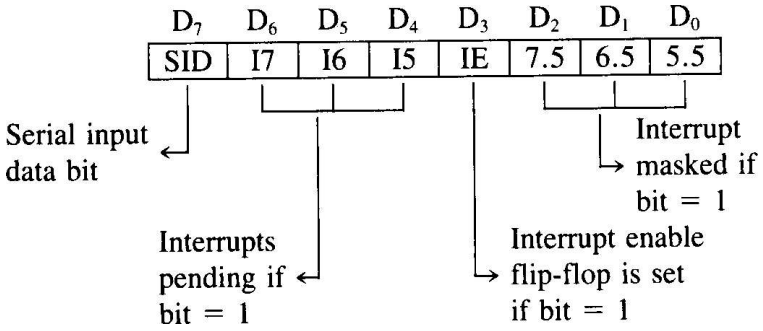
- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags areaffected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- **Example:** EI

CONTROL INSTRUCTIONS

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- **Example:** RIM

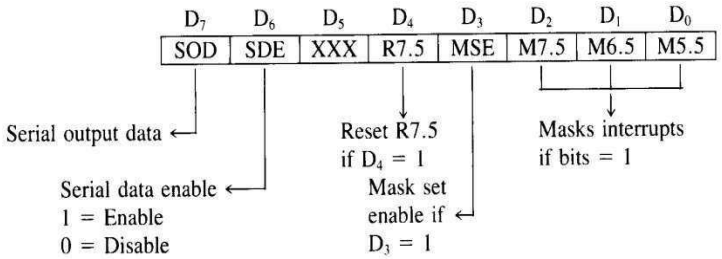
RIM Instruction



SIM Instruction

Opcode	Operand	Description
SIM	None	Set Interrupt Mask

- This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
- The instruction interprets the accumulator contents as follows.
- **Example:** SIM



ASSEMBLY LANGUAGE PROGRAMMING

1. Write a program to transfer a block of data from one location to the other.

```
        5000 Start  LXI    B, 4A01
                    LXI    H, 5101
                    MVI    D,05
                Loop MOV   A, M
STAX B INX                 H
INX                        B
DCR                        D
JNZ

Loop HLT
```

2. Write an assembly language program to add two 8 bit umbers.

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Store the value of sum and carry in memorylocation.
7) Terminate the program.

```
MVI    C, 00    Initialize C register to 00
LDA    4150     Load the value to Accumulator.
MVI    C, 00    Initialize C register to 00
LDA    4150     Load the value to Accumulator.
MOV    B, A     Move the content of Accumulator to B register.
LDA    4151     Load the value to Accumulator.
ADD    B        Add the value of register B to A
JNC    LOOP     Jump on no carry.
INR    C        Increment value of register C
LOOP   4152     Store the value of Accumulator (SUM).
: STA
MOV    A, C     Move content of register C to Acc.
```

STA	4153	Store the value of Accumulator (CARRY)
HLT		Halt the program.

3. Write an assembly language program to subtract two 8 bit numbers.

- Start the program by loading the first data into Accumulator.
- Move the data to a register (B register).
- Get the second data and load into Accumulator.
- Subtract the two register contents.
- Check for carry.
- If carry is present take 2's complement of Accumulator.
- Store the value of borrow in memory location.
- Store the difference value (present in Accumulator) to amemory
- location and terminate the program.

MVI	C, 00	Initialize C to 00
LDA	4150	Load the value to Acc.
MOV	B, A	Move the content of Acc to B register.
LDA	4151	Load the value to Acc.
SUB	B	

4.

5. JNC LOOP Jump on no carry.

6. CMA Complement Accumulator contents.

7. INR A Increment value in

Accumulator. INR C Increment value in register C

LOOP: STA	4152	Store the value of A-reg to memory address.
MOV	A, C	Move contents of register C to Accumulator.
4. STA	4153	Store the value of Accumulator memory address.
5. HLT		Terminate the program.

6.

3. Subtraction two 8-bit BCD number using 8085

1 Perform subtraction by tens complement method
2 Take nine's complement of second no.(99-no)
3 Add one to nine's complement [(99-no) +1] to get 10's complement
4 Add with first no.
5 Convert to BCD using DAA instr.
6 Store in memory location.

LDA	2050 H	Load the first number to accumulator from Memory
MOV B	A	Store the number in B reg.
LDA	2051H	Load the second number to accumulator from memory
MOV C	A	Store the number in C reg.
MVI A	99H	Load acc. With 99H
SUB	C	Subtract second no from C reg.
ADD	B	Add the content with B reg.
DAA		Convert to BCD using DAA instr.
STA	5052	Store in memory location.
HLT		Halt the program.

4. Write an assembly language program to add two 16 bit numbers.

2050	
2051	

2060	
2061	

1. Clear the content in accumulator
2. Set the no. of bytes to be added in C reg.
3. Point to the first no.memory location by loading the address in HL reg. pair
4. Point to the second no.memory location by loading the address in DE reg. pair.
5. Add the first byte and store in first memory location
6. Decrement the counter reg. ; check for zero
6. Until zero continue adding
7. HLT

	XRA	A	Clear the acc.
	MVI C	02H	Add 02H immediate data in C reg.
	LXI H	2050H	Load HL reg. pair with first memory location address
	LXI D	2060H	Load DE reg. pair with second memory location address
HERE		LDAX D	load the content from memory whose address is in DE reg. pair
	ADC	M	Add with carry with the content in acc.
	MOV	M,A	Copy the content from acc. to memory location whose address is in HL reg.pair
	INX	H	Increment the content in HL reg.pair
	INX	D	; Decrement the content in DE reg.pair DCR C; Increment the content in C reg.
	JNZ	HERE	: Continue the process from HERE; until zero
	HLT		Halt the program.

5. Write an assembly language program to subtract two 16 bit numbers.

1. Load the first no.from memory location to accumulator
2. Store it in B reg.
3. Load the second no.from memory
4. Subtract with first no.
5. Check for carry
6. If carry is produced; increment C reg.
7. Store the LSB and MSB to memory location.

LDA	2050 H	Load the first no.from memory location to accumulator
MOV B	A	Move the content from Acc. to B reg
LDA	2051H	Load the second no.from memory location to accumulator
MVI C	OOH	Clear C reg
SUB	B	Subtract the content from acc. with B reg
JNC	GOTO	Continue until Carry
INR	C	increment the content in C reg.
GOTO:	STA 2052H	Store the content in acc. to memory (LSB)
MOV A	C	Copy the content from C.reg. to acc.(MSB
STA	2053H	Store the content from acc. to memory location(MSB)
HLT		End program

6. Write an assembly language program to subtract two 8 bit BCD numbers.

LDA 2050 H MOV B,A LDA 2051H

MOV C,A MVI A,99H SUB C

INR A ADD B DAA

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing.

2. A.K.Ray and Bhurchandi, "Advanced Microprocessor", 1st Edition, TMH Publication.

3. Kenneth J.Ayala, "The 8051 microcontroller Architecture, Programming and applications" 2nd Edition ,Penram international.

4. Doughlas V.Hall, "Microprocessors and Digital system", 2nd Editon, Mc Graw Hill,1983.

5. Md.Rafiquzzaman, "Microprocessors and Microcomputer based system design", 2nd Editon,Universal Book Stall, 1992.

6. Hardware Reference Manual for 80X86 family", Intel Corporation, 1990.

Question Bank

Part A

1. How many operations are there in the instruction set of 8085?
2. List out the five categories of the 8085 instructions. Give examples of the instructions for each group?
3. Explain the difference between a JMP instruction and CALL instruction
4. Explain the purpose of the I/O instructions IN and OUT.
5. What is the difference between the shifts and rotate instructions?
6. How many address lines in a 4096 x 8 EPROM CHIP?
7. What are the control signals used for DMA operation
8. What is meant by Wait State?
9. List the four instructions which control the interrupt structure of the 8085 microprocessor.
10. What is meant by interrupt?

Part B

1. A pharmacist is tasked with sorting and arranging ten drugs based on their MRP values in a cold storage unit. The drug which costs less should be placed at the last of the row and the drug with high MRP value should be placed at the top of the row. Assist the pharmacist by developing an Assembly language program using 8085 for the above said sorting application.
2. Examine the different Data Transfer instructions available in 8085 microprocessor in detail with necessary examples.
3. Interpret the use of different machine control instructions used in 8 bit 8085 processor.
4. Examine the use of different 8085 Logical instructions with necessary examples