KESHAV MEMORIAL ENGINEERING COLLEGE

A Unit of Keshav Memorial Technical Educational Society (KMTES)
Approved by AICTE, New Delhi, Affliated to Osmania University,Hyderabad
Survey No.35, Kachavanisingaram Village, Uppal, Hyderabad-500088, Telangana.

COMPUTER ORGANIZATION LAB MANUAL

Computer Science Engineering

IV Semester

(2023-2024)

ANNEXURE-I
UNIVERSITY SYLLABUS

COMPUTER ORGANIZATION LABORATORY

Instruction	3 Periods Per week
Duration of University Examination	3 Hours
University Examination	50 Marks
Sessionals	25 Marks

Course Objectives:

1. To become familiar with the architecture and Instruction set of Intel 8085 microprocessor.
2. To provide practical hands on experience with Assembly Language Programming.
3. To familiarize the students with interfacing of various peripheral devices with 8085 microprocessor.

List of Experiments

1. Tutorials on 8085 Programming.
2. Interfacing and programming of 8255. (E.g. traffic light controller).
3. Interfacing and programming of 8254.
4. Interfacing and programming of 8279.
5. A/D and D/A converter interface.
6. Stepper motor interface.
7. Display interface.

Note: Adequate number of programs covering all the instructions of 8085 instruction set should be done on the 8085 microprocessor trainer kit.

COMPUTER ORGANIZATRION LAB

8085 Programs

Exp 1: INTRODUCTION TO 8085

INTEL 8085 is one of the most popular 8-bit microprocessor capable of addressing 64 KB of memory and its architecture is simple. The device has 40 pins, requires +5 V power supply and can operate with 3MHz single phase clock.

ALU (Arithmetic Logic Unit):

The 8085A has a simple 8-bit ALU and it works in coordination with the accumulator, temporary registers, 5 flags and arithmetic and logic circuits. ALU has the capability of performing several mathematical and logical operations. The temporary registers are used to hold the data during an arithmetic and logic operation. The result is stored in the accumulator and the flags are set or reset according to the result of the operation. The flags are affected by the arithmetic and logic operation. They are as follows:

- Sign flag

  After the execution of the arithmetic - logic operation if the bit D7 of the result is 1, the sign flag is set. This flag is used with signed numbers. If it is 1, it is a negative number and if it is 0, it is a positive number.

- Zero flag

  The zero flag is set if the ALU operation results in zero. This flag is modified by the result in the accumulator as well as in other registers.

- Auxillary carry flag

  In an arithmetic operation when a carry is generated by digit D3 and passed on to D4, the auxillary flag is set.

- Parity flag

  After arithmetic – logic operation, if the result has an even number of 1’s the flag is set. If it has odd number of 1’s it is reset.

- Carry flag

  If an arithmetic operation results in a carry, the carry flag is set. The carry flag also serves as a borrow flag for subtraction.

Timing and control unit

This unit synchronizes all the microprocessor operation with a clock and generates the control signals necessary for communication between the microprocessor and peripherals. The control signals RD (read) and WR (write) indicate the availability of data on the data bus.

Instruction register and decoder

The instruction register and decoder are part of the ALU. When an instruction is fetched from memory it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequence of events to follow.

Register array

The 8085 has six general purpose registers to store 8-bit data during program execution. These registers are identified as B, C, D, E, H and L. they can be combined as BC, DE and HL to perform 16-bit operation.

Accumulator

Accumulator is an 8-bit register that is part of the ALU. This register is used to store 8-bit data and to perform arithmetic and logic operation. The result of an operation is stored in the accumulator.

Program counter

The program counter is a 16-bit register used to point to the memory address of the next instruction to be executed.

Stack pointer

It is a 16-bit register which points to the memory location in R/W memory, called the Stack.

Communication lines

8085 microprocessor performs data transfer operations using three communication linescalled buses. They are address bus, data bus and control bus.

- Address bus – it is a group of 16-bit lines generally identified as $A_0 – A_{15.}$ The address bus is unidirectional i.e., the bits flow in one direction from microprocessor to the peripheral devices. It is capable of addressing 2^{16} memory locations.
- Data bus – it is a group of 8 lines used for data flow and it is bidirectional. The data ranges from 00 – FF.
- Control bus – it consist of various single lines that carry synchronizing signals. The microprocessor uses such signals for timing purpose.

Exp 2. 8 BIT DATA ADDITION

AIM:

To add two 8 bit numbers stored at consecutive memory locations.

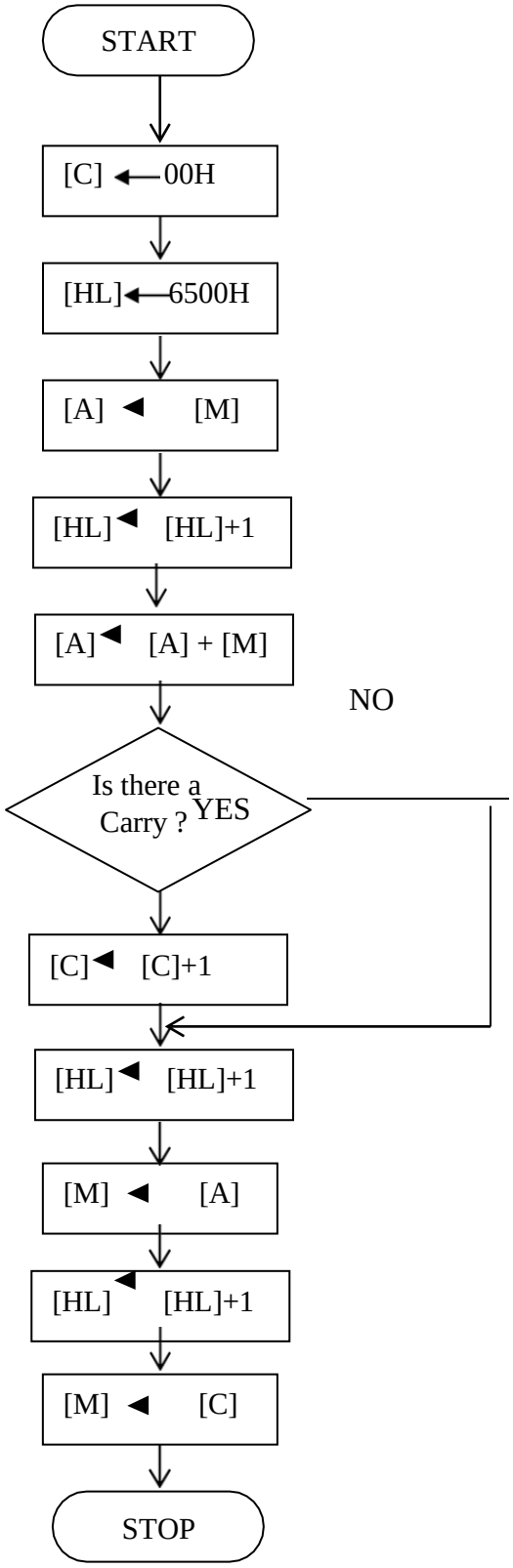
ALGORITHM:

1. **Initialize memory pointer to data location.**
2. **Get the first number from memory in accumulator.**
3. **Get the second number and add it to the accumulator.**
4. **Store the answer at another memory location.**

RESULT:

Thus the 8 bit numbers stored at 6500 & 6501 are added and the result stored at 6502 and 6503.

FLOW CHART:


START
[C] ← 00H
[HL] ← 6500H
[A] ◄ [M]
[HL] ◄ [HL]+1
[A] ◄ [A] + [M]
NO
Is there a Carry ?
YES
[C] ◄ [C]+1
[HL] ◄ [HL]+1
[M] ◄ [A]
[HL] ◄ [HL]+1
[M] ◄ [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
6100		START	MVI	C, 00	Clear C reg.
6101					
6102			LXI	H, 6500	Initialize HL reg. to
6103					6500
6104					
6105			MOV	A, M	Transfer first data to accumulator
6106			INX	H	Increment HL reg. to point next memory Location.
6107			ADD	M	Add first number to acc. Content.
6108			JNC	L1	Jump to location if
6109					result does not yield
610A					carry.
610B			INR	C	Increment C reg.
610C		L1	INX	H	Increment HL reg. to point next memory Location.
610D			MOV	M, A	Transfer the result from acc. to memory.
610E			INX	H	Increment HL reg. to point next memory Location.
610F			MOV	M, C	Move carry to memory
6110			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
6500		6502	
6501		6503	

Exp 3: 8 BIT DATA SUBTRACTION

AIM:

To Subtract two 8 bit numbers stored at consecutive memory locations.

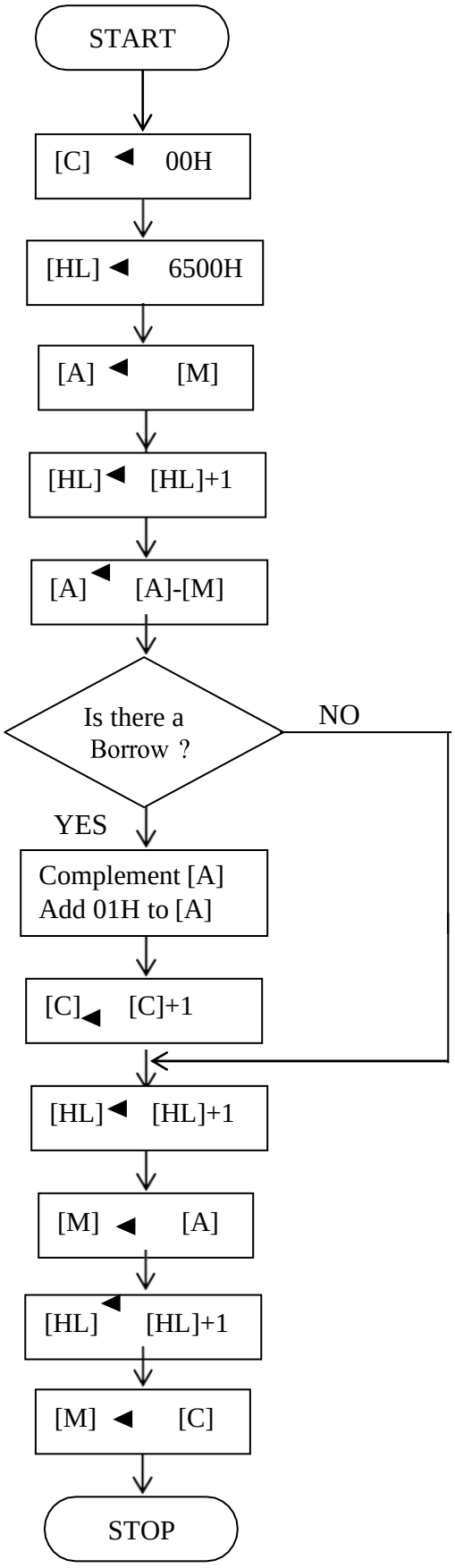
ALGORITHM:

1. **Initialize memory pointer to data location.**
2. **Get the first number from memory in accumulator.**
3. **Get the second number and subtract from the accumulator.**
4. **If the result yields a borrow, the content of the acc. is complemented and 01H is added to it (2's complement). A register is cleared and the content of that reg. is incremented in case there is a borrow. If there is no borrow the content of the acc. is directly taken as the result.**
5. **Store the answer at next memory location.**

RESULT:

Thus the 8 bit numbers stored at 6500 & 6501 are subtracted and the result stored at 6502 and 6503.

FLOW CHART:


START
[C] ◄ 00H
[HL] ◄ 6500H
[A] ◄ [M]
[HL] ◄ [HL]+1
[A] ◄ [A]-[M]
Is there a Borrow ?
NO
YES
Complement [A]
Add 01H to [A]
[C] ◄ [C]+1
[HL] ◄ [HL]+1
[M] ◄ [A]
[HL] ◄ [HL]+1
[M] ◄ [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
6100		START	MVI	C, 00	Clear C reg.
6101					
6102			LXI	H, 6500	Initialize HL reg. to
6103					6500
6104					
6105			MOV	A, M	Transfer first data to accumulator
6106			INX	H	Increment HL reg. to point next mem. Location.
6107			SUB	M	Subtract first number from acc. Content.
6108			JNC	L1	Jump to location if
6109					result does not yield
610A					borrow.
610B			INR	C	Increment C reg.
610C			CMA		Complement the Acc. content
610D			ADI	01H	Add 01H to content of
610E					acc.
610F		L1	INX	H	Increment HL reg. to point next mem. Location.
6110			MOV	M, A	Transfer the result from acc. to memory.
6111			INX	H	Increment HL reg. to point next mem. Location.
6112			MOV	M, C	Move carry to mem.
6113			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
6500		6502	
6501		6503	

Exp 4: 8 BIT DATA MULTIPLICATION

AIM:

To multiply two 8 bit numbers stored at consecutive memory locations and store the result in memory.

ALGORITHM:

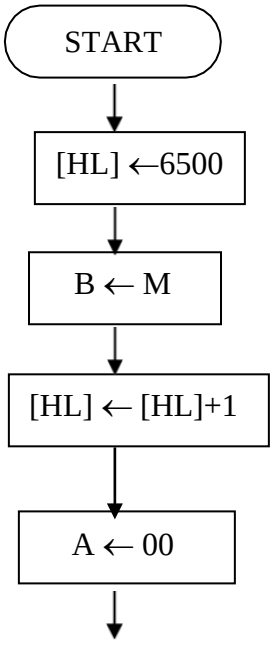
LOGIC: **Multiplication can be done by repeated addition.**

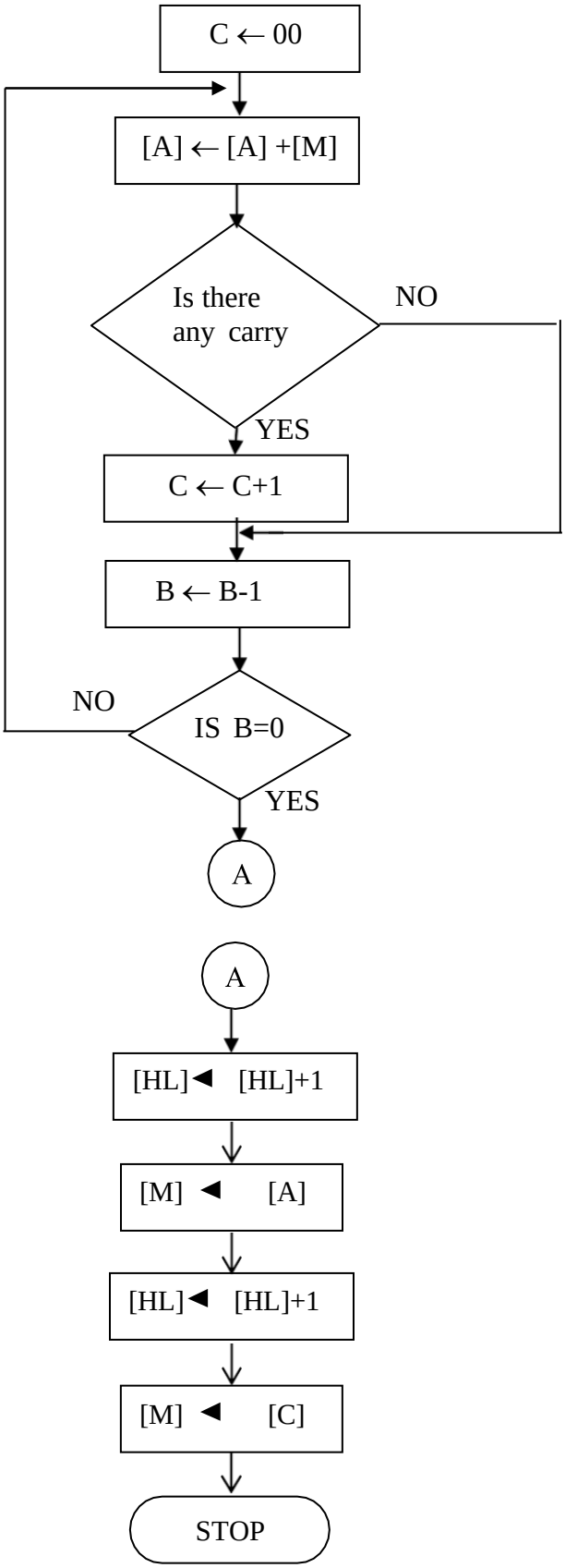
1. **Initialize memory pointer to data location.**
2. **Move multiplicand to a register.**
3. **Move the multiplier to another register.**
4. **Clear the accumulator.**
5. **Add multiplicand to accumulator**
6. **Decrement multiplier**
7. **Repeat step 5 till multiplier comes to zero.**
8. **The result, which is in the accumulator, is stored in a memory location.**

RESULT:

Thus the 8-bit multiplication was done in 8085μp using repeated addition method.

FLOW CHART:




C ← 00
[A] ← [A] +[M]
Is there any carry
NO
YES
C ← C+1
B ← B-1
NO
IS B=0
YES
A
A
[HL]◄ [HL]+1
[M] ◄ [A]
[HL]◄ [HL]+1
[M] ◄ [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
6100		START	LXI	H, 6500	Initialize HL reg. to 6500
6101					
6102					
6103			MOV	B, M	Transfer first data to reg. B
6104			INX	H	Increment HL reg. to point next mem. Location.
6105			MVI	A, 00H	Clear the acc.
6106					
6107			MVI	C, 00H	Clear C reg for carry
6108					
6109		L1	ADD	M	Add multiplicand multiplier times.
610A			JNC	NEXT	Jump to NEXT if there is no carry
610B					
610C					
610D			INR	C	Increment C reg
610E		NEXT	DCR	B	Decrement B reg
610F			JNZ	L1	Jump to L1 if B is not zero.
6110					
6111					
6112			INX	H	Increment HL reg. to point next mem. Location.
6113			MOV	M, A	Transfer the result from acc. to memory.
6114			INX	H	Increment HL reg. to point next mem. Location.
6115			MOV	M, C	Transfer the result from C reg. to memory.
6116			RST 1 END		Stop the program

OBSERVATION:

INPUT		OUTPUT	
6500		6502	
6501		6503	

Exp 5: 8 BIT DATA DIVISION

AIM:

To divide two 8-bit numbers and store the result in memory.

ALGORITHM:

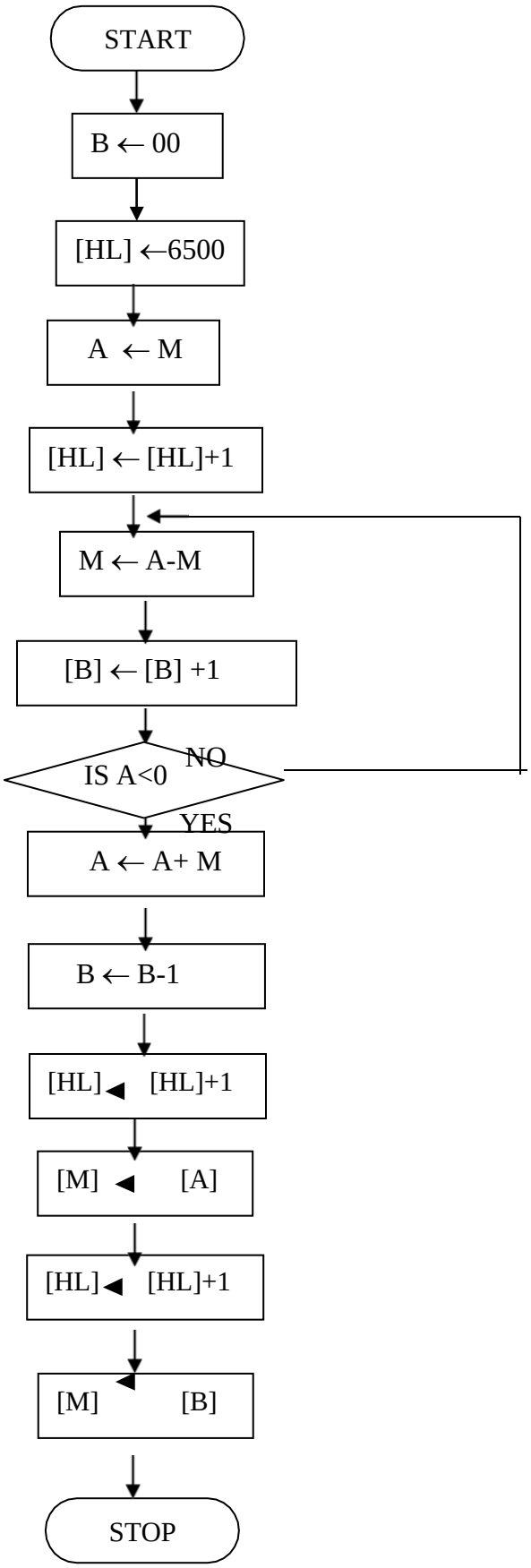
LOGIC: **Division is done using the method Repeated subtraction.**

1. **Load Divisor and Dividend**
2. **Subtract divisor from dividend**
3. **Count the number of times of subtraction which equals the quotient**
4. **Stop subtraction when the dividend is less than the divisor .The dividend now becomes the remainder. Otherwise go to step 2.**
5. **stop the program execution.**

RESULT:

Thus an ALP was written for 8-bit division using repeated subtraction method and executed using 8085μ p kits

FLOWCHART:


START
B ← 00
[HL] ←6500
A ← M
[HL] ← [HL]+1
M ← A-M
[B] ← [B] +1
IS A<0
NO
YES
A ← A+ M
B ← B-1
[HL] [HL]+1
[M] [A]
[HL] [HL]+1
[M] [B]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			MVI	B,00	Clear B reg for quotient
6101					
6102			LXI	H,6500	Initialize HL reg. to
6103					6500H
6104					
6105			MOV	A,M	Transfer dividend to acc.
6106			INX	H	Increment HL reg. to point next mem. Location.
6107		LOOP	SUB	M	Subtract divisor from dividend
6108			INR	B	Increment B reg
6109			JNC	LOOP	Jump to LOOP if result does
610A					not yield borrow
610B					
610C			ADD	M	Add divisor to acc.
610D			DCR	B	Decrement B reg
610E			INX	H	Increment HL reg. to point next mem. Location.
610F			MOV	M,A	Transfer the remainder from acc. to memory.
6110			INX	H	Increment HL reg. to point next mem. Location.
6111			MOV	M,B	Transfer the quotient from B reg. to memory.
6112			RST 1		Stop the program
			END		

OBSERVATION:

S.NO	INPUT		OUTPUT	
	ADDRESS	DATA	ADDRESS	DATA
1	6500		6502	
	6501		6503	
2	6500		6502	
	6501		6503	

Exp 6: LARGEST ELEMENT IN AN ARRAY

AIM:

To find the largest element in an array.

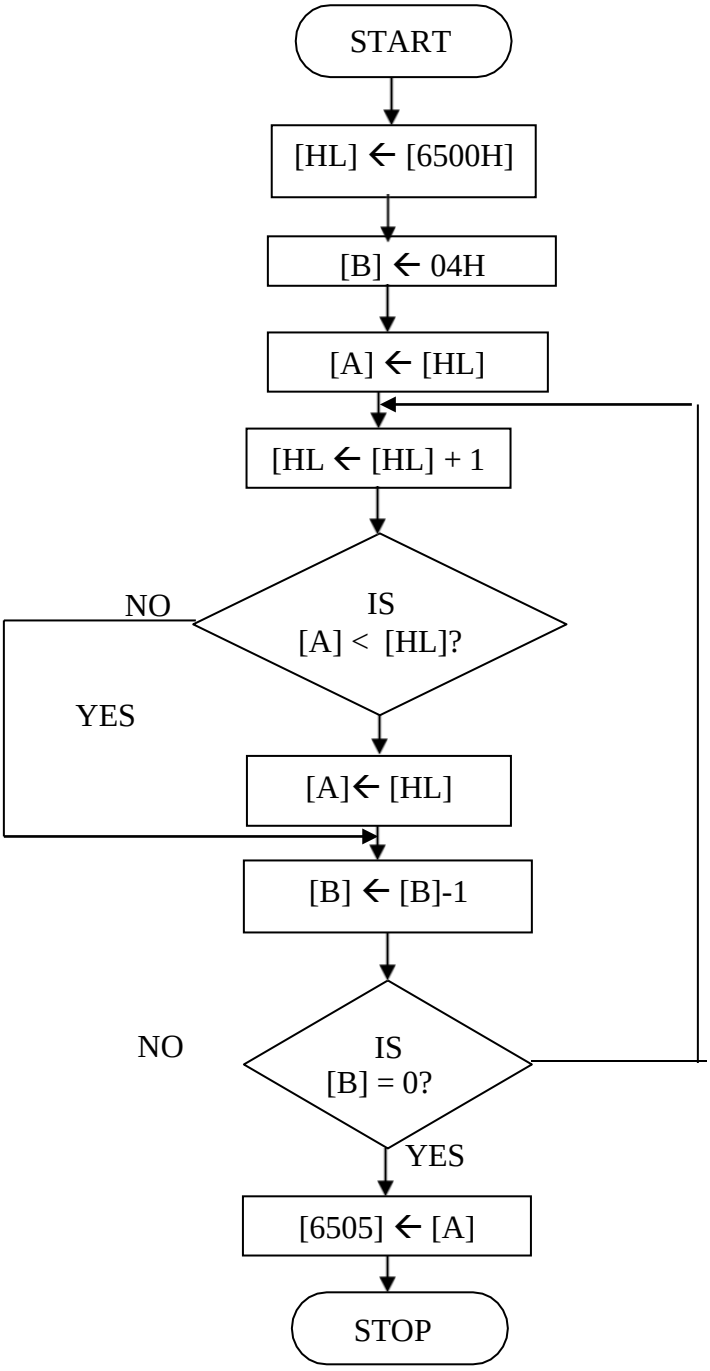
ALGORITHM:

1. Place all the elements of an array in the consecutive memory locations.
2. Fetch the first element from the memory location and load it in the accumulator.
3. Initialize a counter (register) with the total number of elements in an array.
4. Decrement the counter by 1.
5. Increment the memory pointer to point to the next element.
6. Compare the accumulator content with the memory content (next element).
7. If the accumulator content is smaller, then move the memory content (largest element) to the accumulator. Else continue.
8. Decrement the counter by 1.
9. Repeat steps 5 to 8 until the counter reaches zero
10. Store the result (accumulator content) in the specified memory location.

RESULT:

Thus the largest number in the given array is found out.

FLOW CHART:


START
[HL] ← [6500H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
NO
YES
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[6505] ← [A]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			LXI	H,6500	Initialize HL reg. to
6101					6500H
6102					
6103			MVI	B,04	Initialize B reg with no. of
6104					comparisons(n-1)
6105			MOV	A,M	Transfer first data to acc.
6106		LOOP1	INX	H	Increment HL reg. to pointnext memory location
6107			CMP	M	Compare M & A
6108			JNC	LOOP	If A is greater than M then
6109					goto loop
610A					
610B			MOV	A,M	Transfer data from M to A reg
610C		LOOP	DCR	B	Decrement B reg
610D			JNZ	LOOP1	If B is not Zero go to loop1
610E					
610F					
6110			STA	6505	Store the result in a memory
6111					location.
6112					
6113			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
6500		6505	
6501			
6502			
6503			
6504			

Exp 7: SMALLEST ELEMENT IN AN ARRAY

AIM:

To find the smallest element in an array.

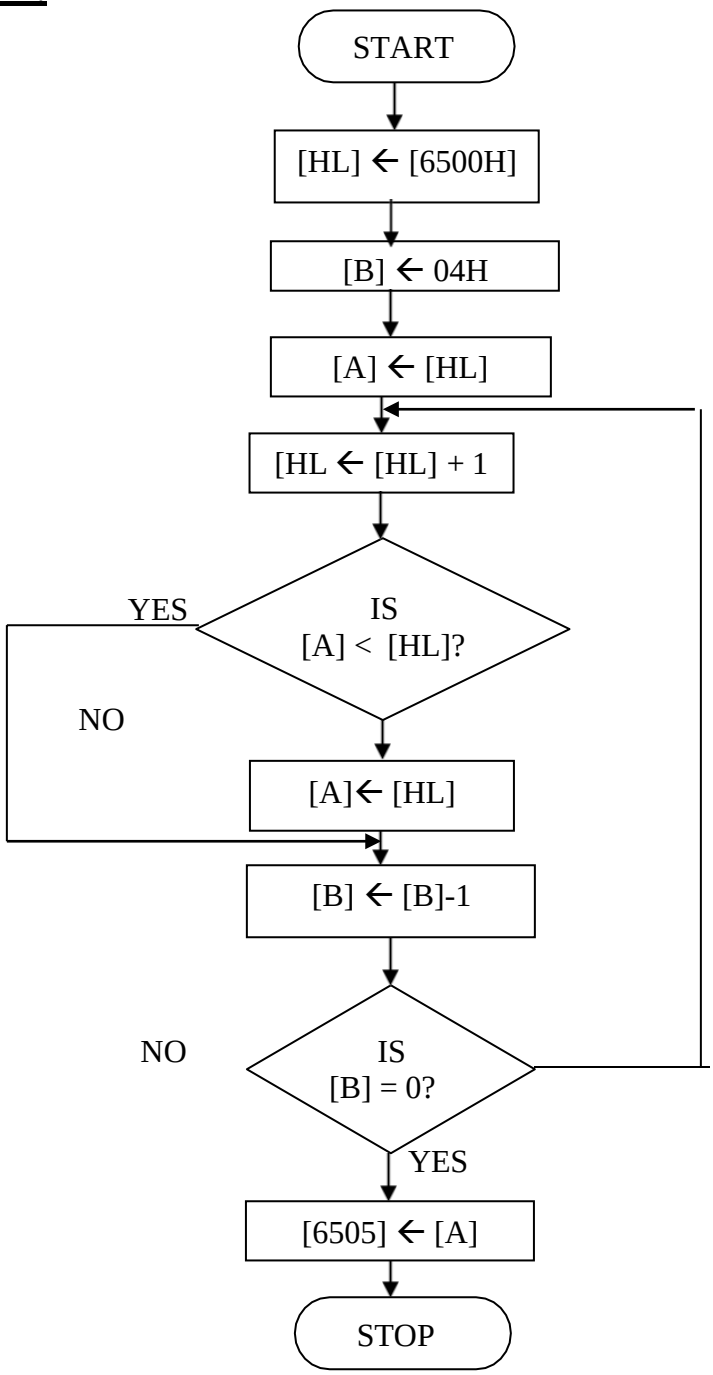
ALGORITHM:

1. Place all the elements of an array in the consecutive memory locations.
2. Fetch the first element from the memory location and load it in the accumulator.
3. Initialize a counter (register) with the total number of elements in an array.
4. Decrement the counter by 1.
5. Increment the memory pointer to point to the next element.
6. Compare the accumulator content with the memory content (next element).
7. If the accumulator content is smaller, then move the memory content (largest element) to the accumulator. Else continue.
8. Decrement the counter by 1.
9. Repeat steps 5 to 8 until the counter reaches zero
10. Store the result (accumulator content) in the specified memory location.

RESULT:

Thus the smallest number in the given array is found out.

FLOW CHART:


START
[HL] ← [6500H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
YES
NO
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[6505] ← [A]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			LXI	H,6500	Initialize HL reg. to
6101					6500H
6102					
6103			MVI	B,04	Initialize B reg with no. of
6104					comparisons(n-1)
6105			MOV	A,M	Transfer first data to acc.
6106		LOOP1	INX	H	Increment HL reg. to pointnext memory location
6107			CMP	M	Compare M & A
6108			JC	LOOP	If A is lesser than M
6109					then goto loop
610A					
610B			MOV	A,M	Transfer data from M to A reg
610C		LOOP	DCR	B	Decrement B reg
610D			JNZ	LOOP1	If B is not Zero go to loop1
610E					
610F					
6110			STA	6505	Store the result in a memory
6111					location.
6112					
6113			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
6500		6505	
6501			
6502			
6503			
6504			

Exp 8: SORTING IN ASCENDING ORDER

AIM:

To sort the given number in the ascending order using 8085 microprocessor.

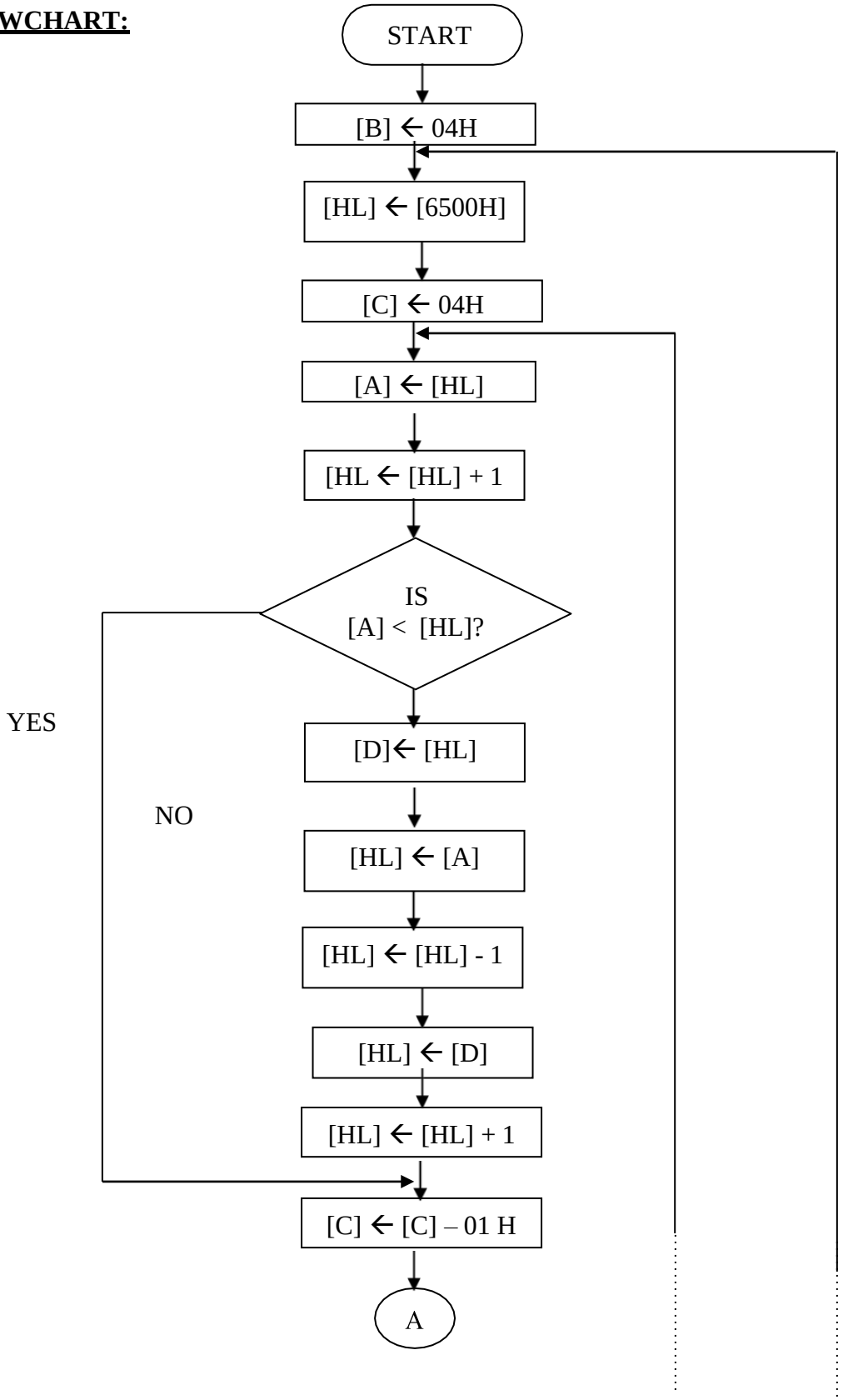
ALGORITHM:

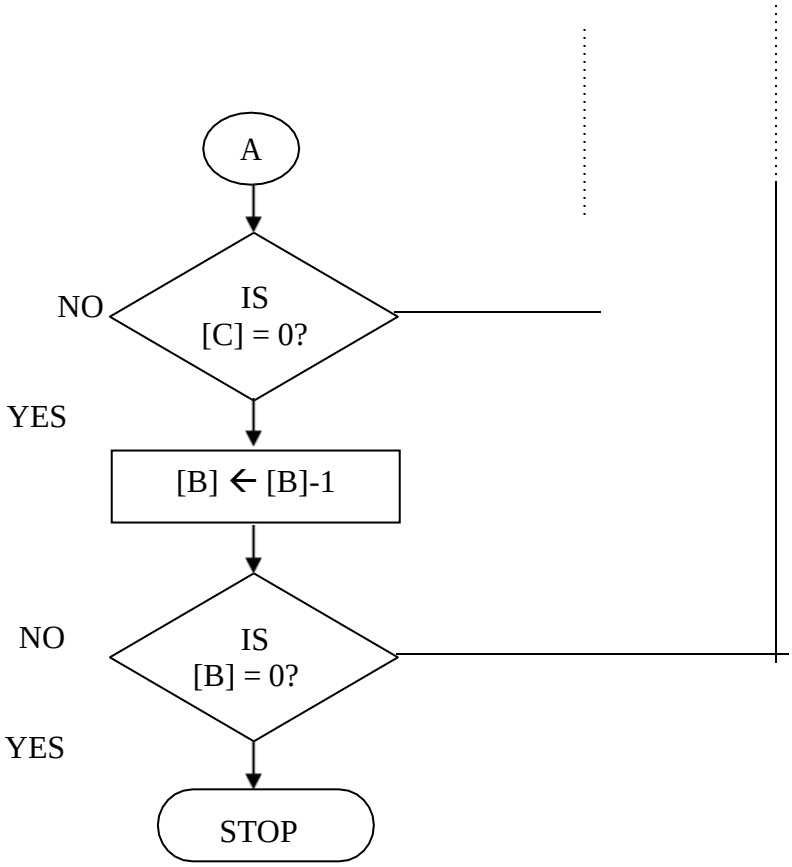
1. Get the numbers to be sorted from the memory locations.
2. Compare the first two numbers and if the first number is larger than second then I interchange the number.
3. If the first number is smaller, go to step 4
4. Repeat steps 2 and 3 until the numbers are in required order

RESULT:

Thus the ascending order program is executed and thus the numbers are arranged in ascending order.

FLOWCHART:


START
[B] ← 04H
[HL] ← [6500H]
[C] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
YES
NO
[D]← [HL]
[HL] ← [A]
[HL] ← [HL] - 1
[HL] ← [D]
[HL] ← [HL] + 1
[C] ← [C] – 01 H
A


A
IS
[C] = 0?
NO
YES
[B] ← [B]-1
IS
[B] = 0?
NO
YES
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			MVI	B,04	Initialize B reg with number
6101					of comparisons (n-1)
6102		LOOP 3	LXI	H,6500	Initialize HL reg. to
6103					6500H
6104					
6105			MVI	C,04	Initialize C reg with no. of
6106					comparisons(n-1)
6107		LOOP2	MOV	A,M	Transfer first data to acc.
6108			INX	H	Increment HL reg. to point next memory location
6109			CMP	M	Compare M & A
610A			JC	LOOP1	If A is less than M then go to
610B					loop1
610C					
610D			MOV	D,M	Transfer data from M to D reg
610E			MOV	M,A	Transfer data from acc to M
610F			DCX	H	Decrement HL pair
6110			MOV	M,D	Transfer data from D to M
6111			INX	H	Increment HL pair
6112		LOOP1	DCR	C	Decrement C reg
6113			JNZ	LOOP2	If C is not zero go to loop2
6114					
6115					
6116			DCR	B	Decrement B reg
6117			JNZ	LOOP3	If B is not Zero go to loop3
6118					
6119					
611A			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
6500		6500	
6501		6501	
6502		6502	
6503		6503	
6504		6504	

Exp 9: SORTING IN DESCENDING ORDER

AIM:

To sort the given number in the descending order using 8085 microprocessor.

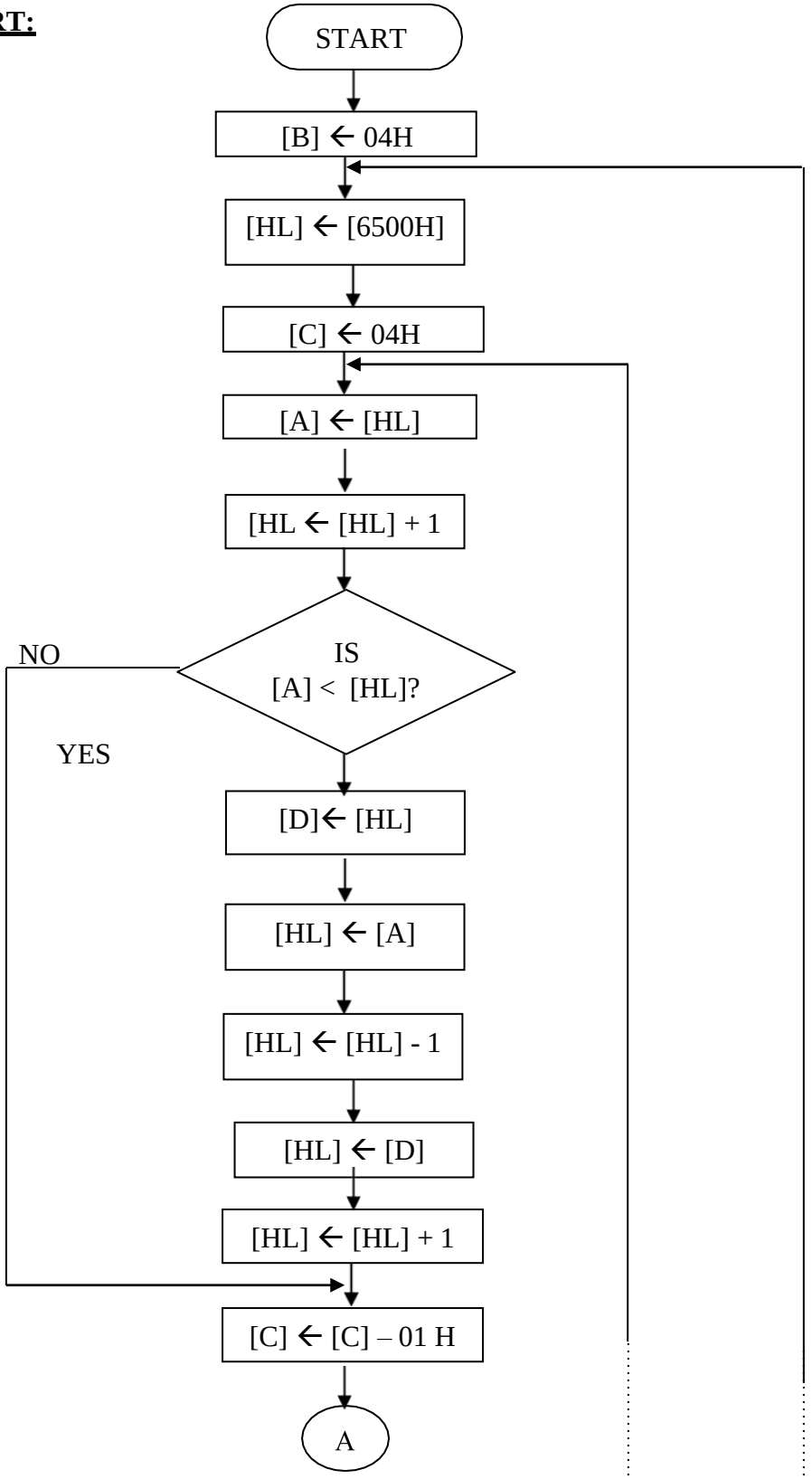
ALGORITHM:

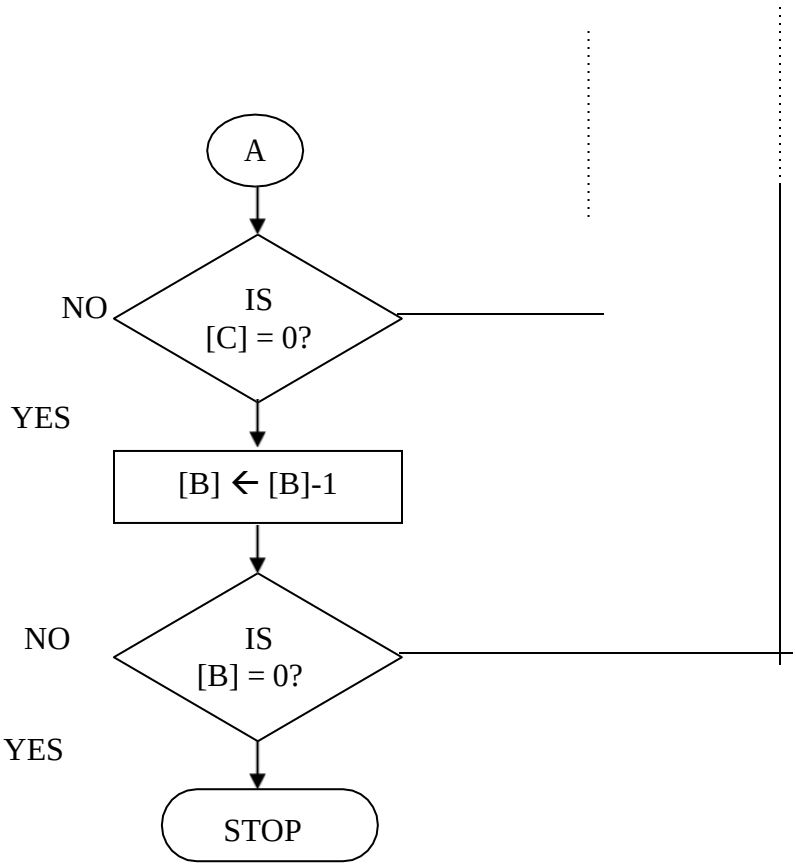
1. Get the numbers to be sorted from the memory locations.
2. Compare the first two numbers and if the first number is smaller than second then I interchange the number.
3. If the first number is larger, go to step 4
4. Repeat steps 2 and 3 until the numbers are in required order

RESULT:

Thus the descending order program is executed and thus the numbers are arranged in descending order.

FLOWCHART:


START
[B] ← 04H
[HL] ← [6500H]
[C] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
NO
YES
[D]← [HL]
[HL] ← [A]
[HL] ← [HL] - 1
[HL] ← [D]
[HL] ← [HL] + 1
[C] ← [C] – 01 H
A


A
IS
[C] = 0?
NO
YES
[B] ← [B]-1
IS
[B] = 0?
NO
YES
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			MVI	B,04	Initialize B reg with number
6101					of comparisons (n-1)
6102		LOOP 3	LXI	H,6500	Initialize HL reg. to
6103					6500H
6104					
6105			MVI	C,04	Initialize C reg with no. of
6106					comparisons(n-1)
6107		LOOP2	MOV	A,M	Transfer first data to acc.
6108			INX	H	Increment HL reg. to point next memory location
6109			CMP	M	Compare M & A
610A			JNC	LOOP1	If A is greater than M then go
610B					to loop1
610C					
610D			MOV	D,M	Transfer data from M to D reg
610E			MOV	M,A	Transfer data from acc to M
610F			DCX	H	Decrement HL pair
6110			MOV	M,D	Transfer data from D to M
6111			INX	H	Increment HL pair
6112		LOOP1	DCR	C	Decrement C reg
6113			JNZ	LOOP2	If C is not zero go to loop2
6114					
6115					
6116			DCR	B	Decrement B reg
6117			JNZ	LOOP3	If B is not Zero go to loop3
6118					
6119					
611A			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
6500		6500	
6501		6501	
6502		6502	
6503		6503	
6504		6504	

Exp 10: CODE CONVERSION –DECIMAL TO HEXDECIMAL

AIM:

To convert a given decimal number to hexadecimal.

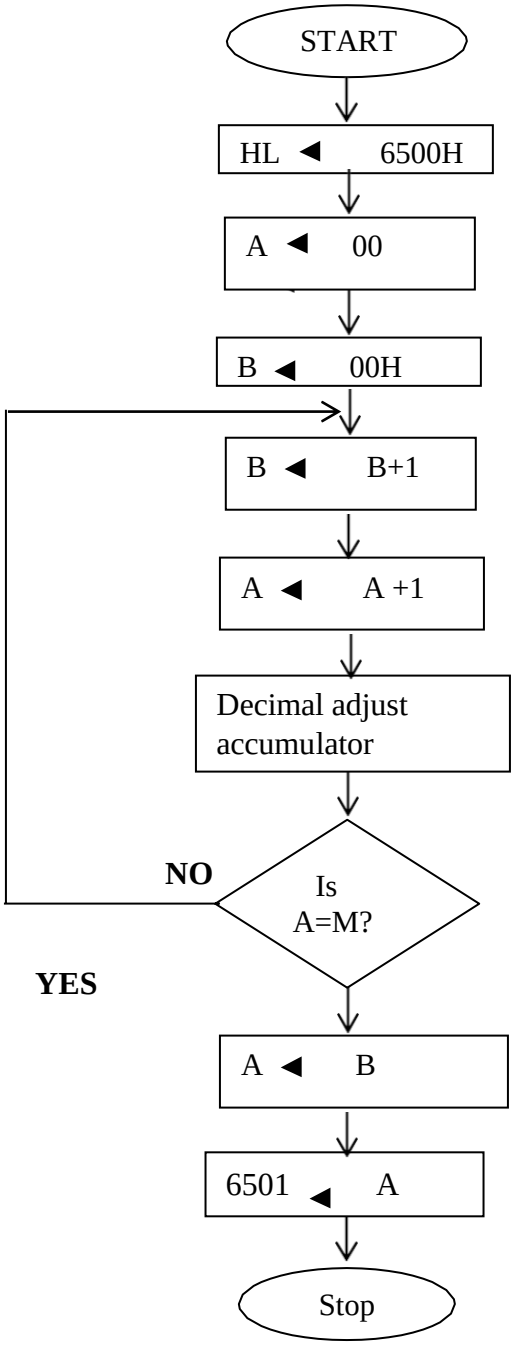
ALGORITHM:

1. **Initialize the memory location to the data pointer.**
2. **Increment B register.**
3. **Increment accumulator by 1 and adjust it to decimal every time.**
4. **Compare the given decimal number with accumulator value.**
5. **When both matches, the equivalent hexadecimal value is in B register.**
6. **Store the resultant in memory location.**

RESULT:

Thus an ALP program for conversion of decimal to hexadecimal was written and executed.

FLOWCHART:


START
HL ◄ 6500H
A ◄ 00
B ◄ 00H
B ◄ B+1
A ◄ A +1
Decimal adjust accumulator
Is A=M?
NO
YES
A ◄ B
6501 ◄ A
Stop

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			LXI	H,6500	Initialize HL reg. to
6101					6500H
6102					
6103			MVI	A,00	Initialize A register.
6104					
6105			MVI	B,00	Initialize B register..
6106					
6107		LOOP	INR	B	Increment B reg.
6108			ADI	01	Increment A reg
6109					
610A			DAA		Decimal Adjust Accumulator
610B			CMP	M	Compare M & A
610C			JNZ	LOOP	If acc and given number are
610D					not equal, then go to LOOP
610E					
610F			MOV	A,B	Transfer B reg to acc.
6110			STA	6501	Store the result in a memory
6111					location.
6112					
6113			RST 1		Stop the program
			END		

RESULT:

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**
6500		**6501**	

Exp 11: CODE CONVERSION –HEXADECIMAL TO DECIMAL

AIM:

To convert a given hexadecimal number to decimal.

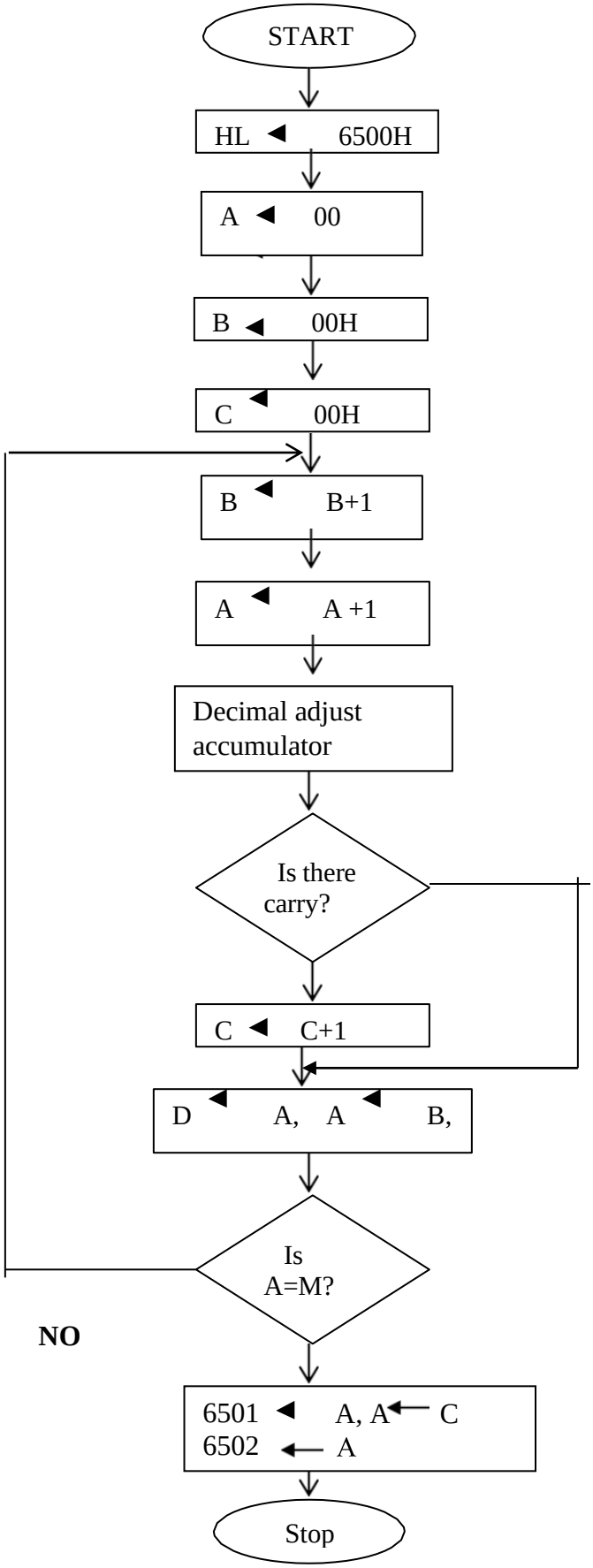
ALGORITHM:

1. **Initialize the memory location to the data pointer.**
2. **Increment B register.**
3. **Increment accumulator by 1 and adjust it to decimal every time.**
4. **Compare the given hexadecimal number with B register value.**
5. **When both match, the equivalent decimal value is in A register.**
6. **Store the resultant in memory location.**

RESULT:

Thus an ALP program for conversion of hexadecimal to decimal was written and executed.

FLOWCHART:


START
HL ◄ 6500H
A ◄ 00
B ◄ 00H
C ◄ 00H
B ◄ B+1
A ◄ A +1
Decimal adjust accumulator
Is there carry?
C ◄ C+1
D ◄ A, A ◄ B,
Is A=M?
NO
6501 ◄ A, A ← C
6502 ← A
Stop

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
6100			LXI	H,6500	Initialize HL reg. to
6101					6500H
6102					
6103			MVI	A,00	Initialize A register.
6104					
6105			MVI	B,00	Initialize B register.
6106					
6107			MVI	C,00	Initialize C register for carry.
6108					
6109		LOOP	INR	B	Increment B reg.
610A			ADI	01	Increment A reg
610B					
610C			DAA		Decimal Adjust Accumulator
610D			JNC	NEXT	If there is no carry go to
610E					NEXT.
610F					
6110			INR	C	Increment c register.
6111		NEXT	MOV	D,A	Transfer A to D
6112			MOV	A,B	Transfer B to A
6113			CMP	M	Compare M & A
6114			MOV	A,D	Transfer D to A
6115			JNZ	LOOP	If acc and given number are
6116					not equal, then go to LOOP
6117					
6118			STA	6501	Store the result in a memory
6119					location.
611A					
611B			MOV	A,C	Transfer C to A
611C			STA	6502	Store the carry in another
611D					memory location.
611E					
611F			RST 1		Stop the program
			END		

RESULT:

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**
6500		**6501**	
		6502	

Exp 12: BCD ADDITION

AIM:

To add two 8 bit BCD numbers stored at consecutive memory locations.

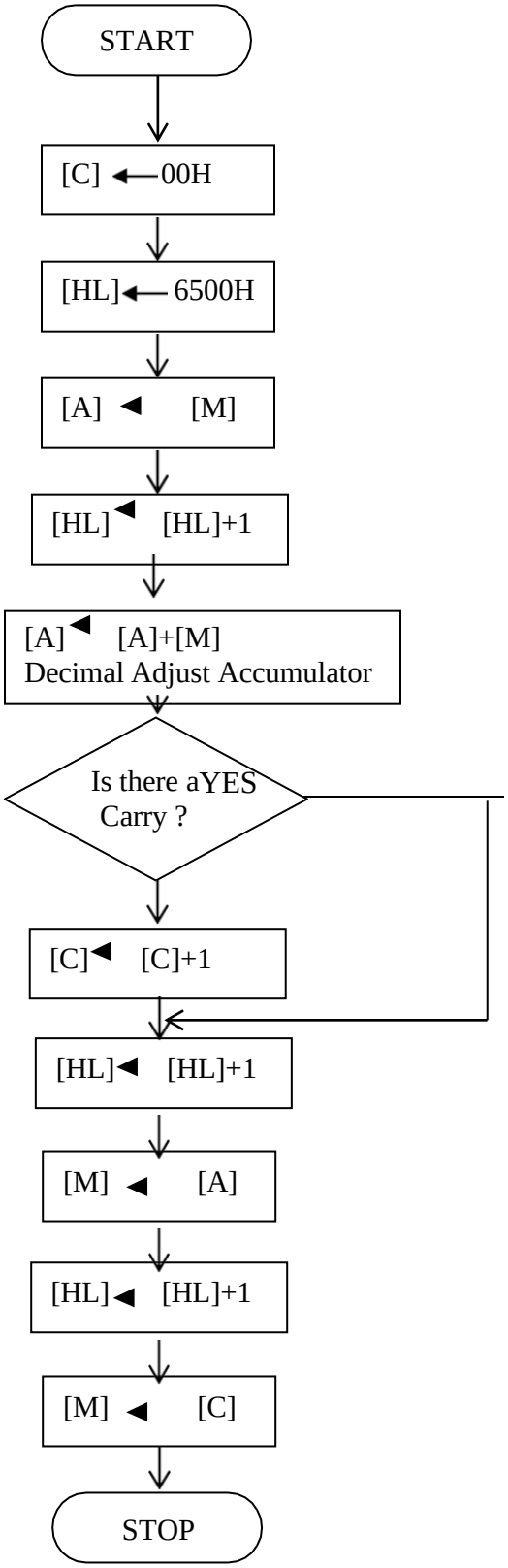
ALGORITHM:

1. **Initialize memory pointer to data location.**
2. **Get the first number from memory in accumulator.**
3. **Get the second number and add it to the accumulator**
4. **Adjust the accumulator value to the proper BCD value using DAA instruction.**
5. **Store the answer at another memory location.**

RESULT:

Thus the 8 bit BCD numbers stored at 6500 & 6501 are added and the result stored at 6502 & 6503.

FLOW CHART:


START
[C] ← 00H
[HL] ← 6500H
[A] ◄ [M]
[HL] ◄ [HL]+1
[A] ◄ [A]+[M]
Decimal Adjust Accumulator
Is there a Carry ?
YES
[C] ◄ [C]+1
[HL] ◄ [HL]+1
[M] ◄ [A]
[HL] ◄ [HL]+1
[M] ◄ [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
6100		START	MVI	C, 00	Clear C reg.
6101					
6102			LXI	H, 6500	Initialize HL reg. to
6103					6500H
6104					
6105			MOV	A, M	Transfer first data to accumulator
6106			INX	H	Increment HL reg. to point next memory Location.
6107			ADD	M	Add first number to acc. Content.
6108			DAA		Decimal adjust accumulator
6109			JNC	L1	Jump to location if
610A					result does not yield
610B					carry.
610C			INR	C	Increment C reg.
610D		L1	INX	H	Increment HL reg. to point next memory Location.
610E			MOV	M, A	Transfer the result from acc. to memory.
610F			INX	H	Increment HL reg. to point next memory Location.
6110			MOV	M, C	Move carry to memory
6111			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
6500		6502	
6501		6503	

Exp 13: BCD SUBTRACTION

AIM:

To Subtract two 8 bit BCD numbers stored at consecutive memory locations.

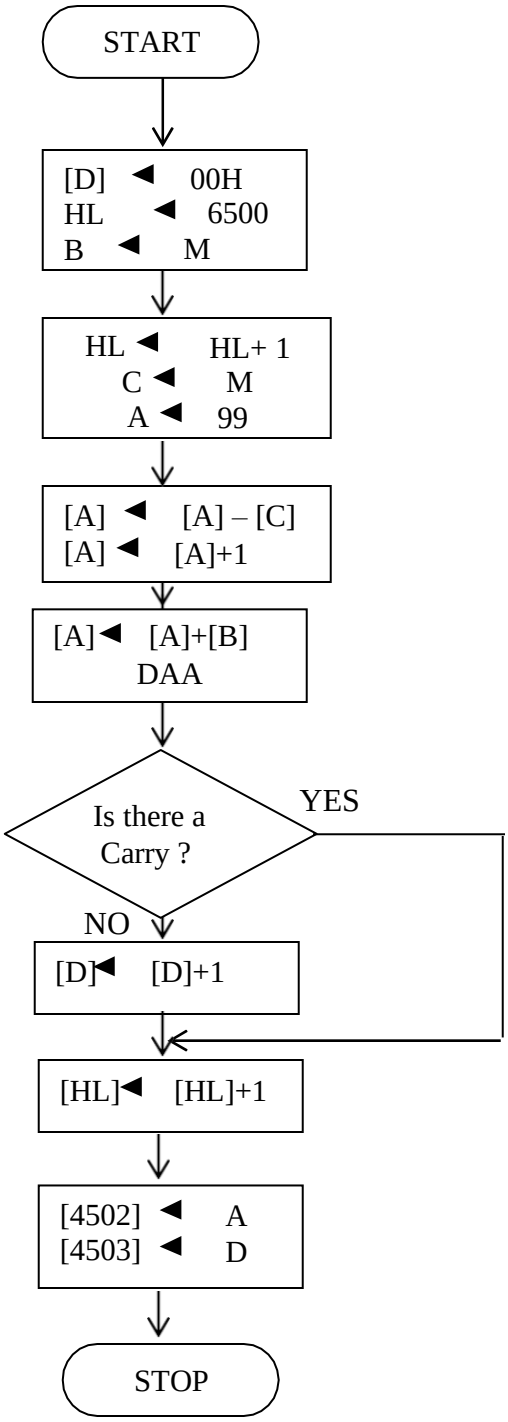
ALGORITHM:

1. **Load the minuend and subtrahend in two registers.**
2. **Initialize Borrow register to 0.**
3. **Take the 100's complement of the subtrahend.**
4. **Add the result with the minuend which yields the result.**
5. **Adjust the accumulator value to the proper BCD value using DAA instruction. If there is a carry ignore it.**
6. **If there is no carry, increment the carry register by 1**
7. **Store the content of the accumulator (result)and borrow register in the specified memory location**

RESULT:

Thus the 8 bit BCD numbers stored at 6500 & 6501 are subtracted and the result stored at 6502 & 6503.

FLOW CHART:


START
[D] ◄ 00H
HL ◄ 6500
B ◄ M
HL ◄ HL+ 1
C ◄ M
A ◄ 99
[A] ◄ [A] – [C]
[A] ◄ [A]+1
[A] ◄ [A]+[B]
DAA
Is there a Carry ?
YES
NO
[D] ◄ [D]+1
[HL] ◄ [HL]+1
[4502] ◄ A
[4503] ◄ D
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
6100		START	MVI	D, 00	Clear D reg.
6101					
6102			LXI	H, 6500	Initialize HL reg. to
6103					6500H
6104					
6105			MOV	B, M	Transfer first data to accumulator
6106			INX	H	Increment HL reg. to point next mem. Location.
6107			MOV	C, M	Move second no. to B reg.
6108			MVI	A, 99	Move 99 to the
6109					Accumulator
610A			SUB	C	Subtract [C] from acc. Content.
610B			INR	A	Increment A register
610C			ADD	B	Add [B] with [A]
610D			DAA		Adjust Accumulator value for Decimal digits
610E			JC	LOOP	Jump on carry to loop
610F					
6110					
6111			INR	D	Increment D reg.
6112		LOOP	INX	H	Increment HL register pair
6113			MOV	M , A	Move the Acc.content to the memory location
6114			INX	H	Increment HL reg. to point next mem. Location.
6115			MOV	M, D	Transfer D register content to memory.
6116			RST 1		Stop the program
			END		

OBSERVATION:

INPUT		OUTPUT	
6500		6502	
6501		6503	

Exp 14: INTRODUCTION TO 8255 PPI

AIM:

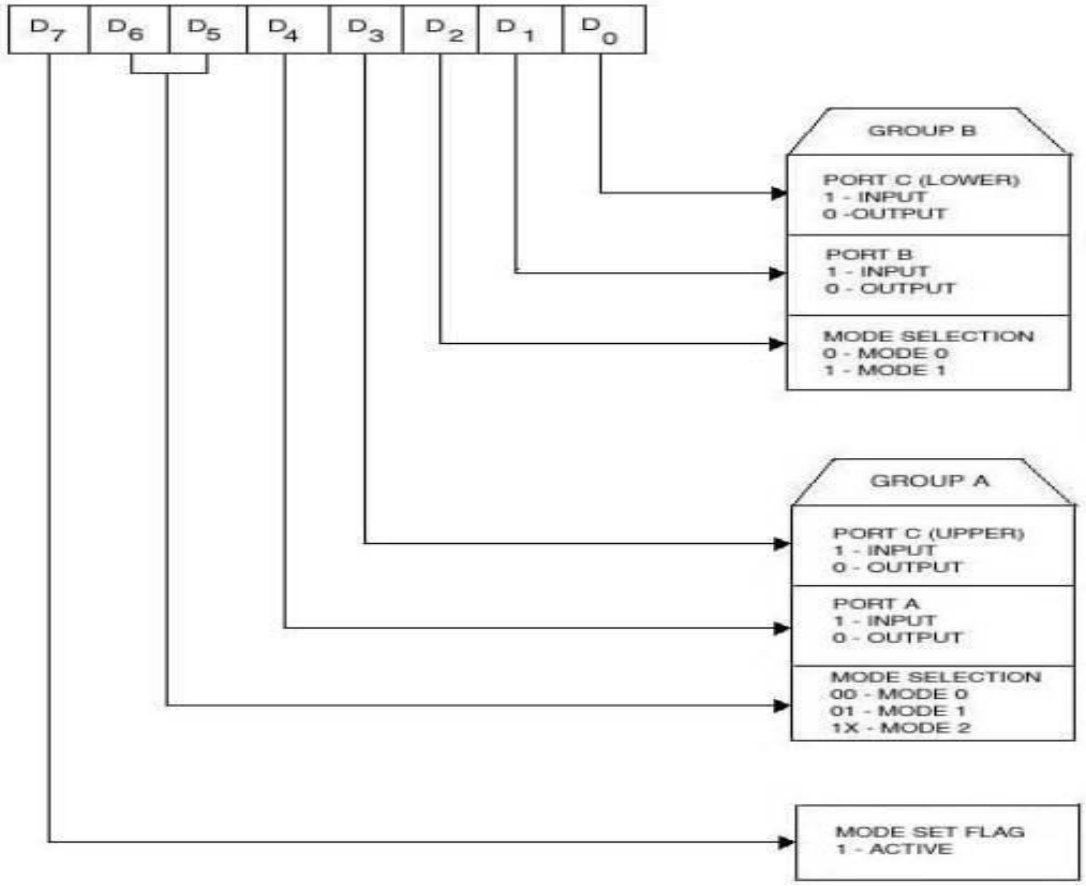
To interface programmable peripheral interface 8255 with 8085 and study its characteristics in mode0,mode1 and BSR mode.

APPARATUS REQUIRED:

8085 μp kit, 8255Interface board, DC regulated power supply, VXT parallel bus

I/O MODES:

Control Word:



MODE 0 – SIMPLE I/O MODE:

This mode provides simple I/O operations for each of the three ports and is suitable for synchronous data transfer. In this mode all the ports can be configured either as input or output port.

Let us initialize port A as input port and port B as output port

PROGRAM:

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
6100		START:	MVI	A, 90	Initialize port A as Input
6101					and Port B as output.
6102			OUT	C6	Send Mode Control
6103					Word
6104			IN	C0	Read from Port A
6105					
6106			OUT	C2	Display the data in port
6107					B
6108			STA	6200	Store the data read from
6109					Port A in 6200
610A					
610B			RST 1		Stop the program.

MODE1 STROBED I/O MODE:

In this mode, port A and port B are used as data ports and port C is used as control signals for strobed I/O data transfer.

Let us initialize port A as input port in mode1

MAIN PROGRAM:

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
6100		START:	MVI	A, B4	Initialize port A as Input
6101					port in mode 1.
6102			OUT	C6	Send Mode Control
6103					Word
6104			MVI	A,09	Set the PC4 bit for INTE
					A
6105					
6106			OUT	C6	Display the data in port
					B
6107					
			EI		
6108			MVI	A,08	Enable RST5.5
6109					
610A			SIM		
			EI		
610B			RST 1		Stop the program.

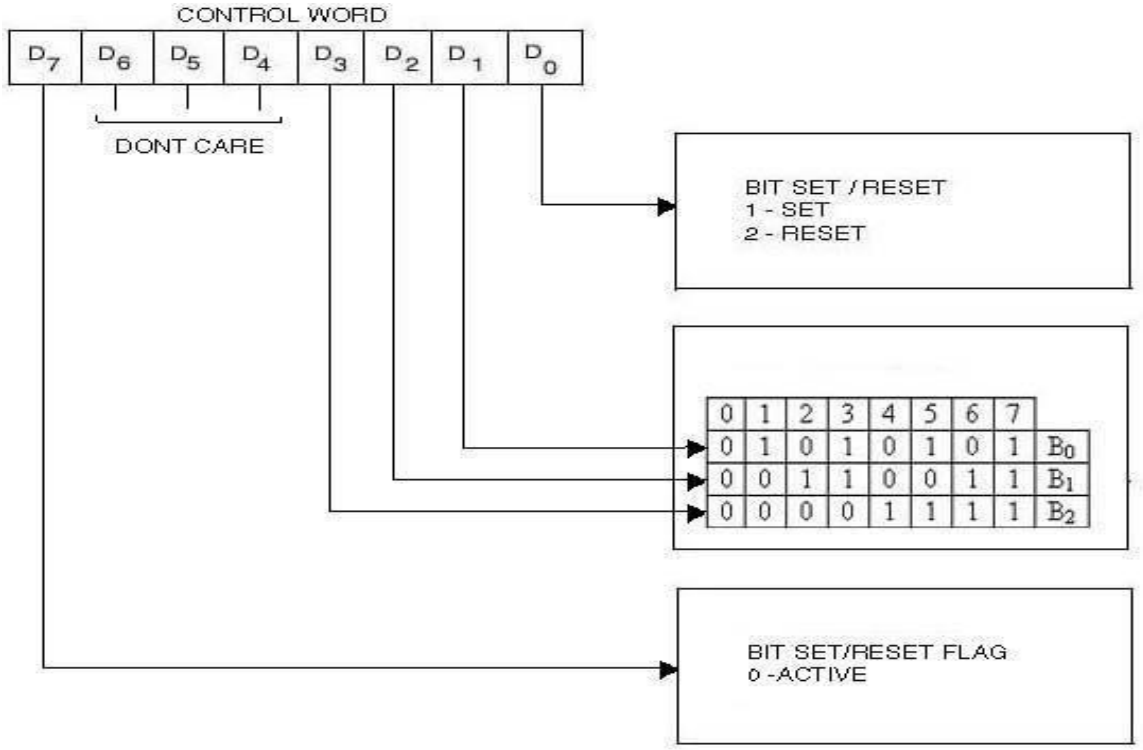
ISR (Interrupt Service Routine)

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
6200		START:	IN	C0	Read from port A
6201					
6202			STA	6500	Store in 6500.
6203					
6204					
6205			RST 1 END		Stop the program.

Sub program:

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
605E			JMP	6200	Go to 6200
605F					
6060					

BSR MODE (Bit Set Reset mode)



Any lines of port c can be set or reset individually without affecting other lines using this mode. Let us set PC0 and PC3 bits using this mode.

PROGRAM:

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
6100		START:	MVI	A, 01	Set PC0
6101					
6102			OUT	C6	Send Mode Control
6103					word
6104			MVI	A,07	Set PC3
6105					
6106			OUT	C6	Send Mode Control
6107					word
6109			RST 1 END		Stop the program.

RESULT:

Thus 8255 is interfaced and its characteristics in mode0,mode1 and BSR mode is studied.

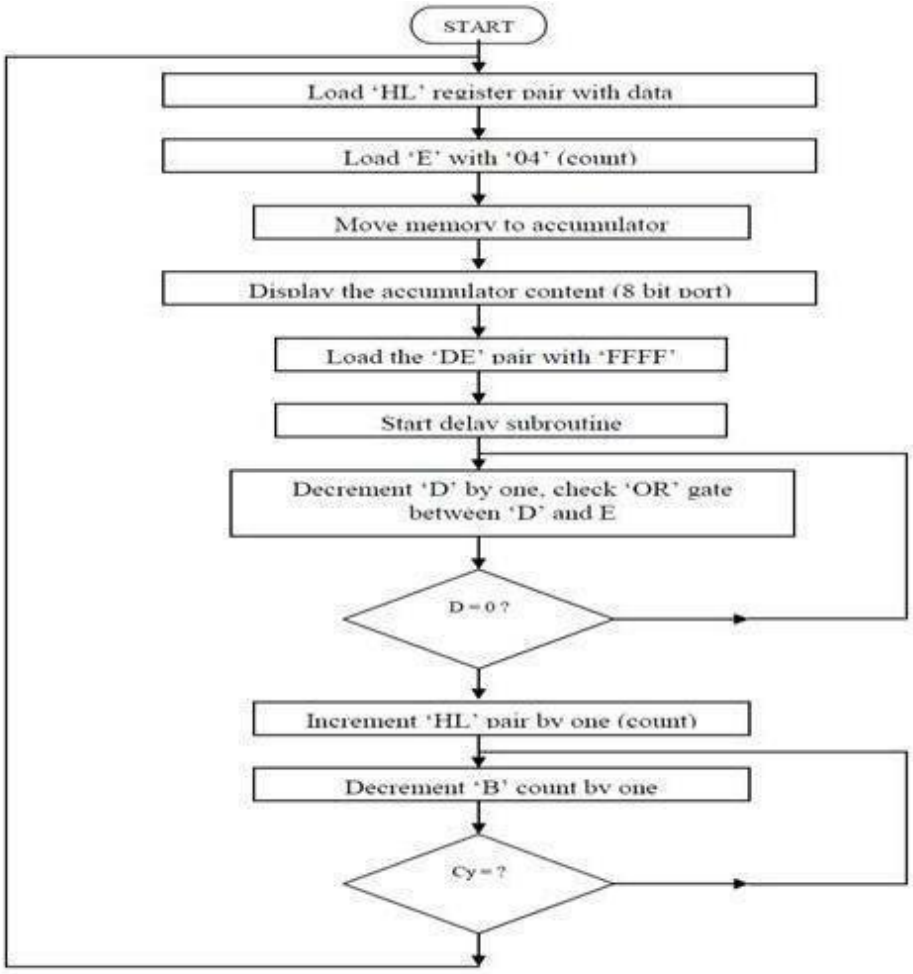
Exp 15: INTERFACING STEPPER MOTOR

Aim: To rotate stepper motor in forward and reverse direction.

Apparatus required:
Stepper motor, 8085 microprocessor kit, (0-5V) power supply.

Algorithm:
Step 1: Load the 'HL' pair wit value from table
Step 2: Move it to 'B' register for setting the counter
Step 3: Move the memory value to accumulator and display it by control word
Step 4: Load 'DE' register pair with FFFF for starting delay subroutine
Step 5: Run the delay loop control D-register becomes zero.
Step 6: Increment 'H' address for next value from table
Step 7: Jump on no zero
Step 8: When B = 0, go to start and restart the program.

Flowchart:



STEPPER MOTOR (CLOCKWISE)

6000,6001	3E,80		MVI	A,08
6002,6003	D3,0B		OUT	0B
6004,6005	3E,00		MVI	A,00
6006,6007	D3,08		OUT	08
6008,6009	3E,08	START	MVI	A,01
600A,600B	D3,08		OUT	08
600C,600D,600E	CD,27 60		CALL	DELAY
600F,6010	3E,04		MVI	A,02
6011,6012	D3,08		OUT	08
6013,6014,6015	CD,27,60		CALL	DELAY 1
6016,6017	3E,02		MVI	A,04
6018,6019	D3,08		OUT	08
610A,610B,610C	CD,27,60		CALL	DELAY 1
610D,610E	3E,01		MVI	A,08
601F,6020	D3,08		OUT	08
6021,6022,6023	CD,27,60		CALL	DELAY 1
6024,6025,6026	C3,08,60		JMP	START
6027,6028,6029	11,FF,FF	DELAY	LXI	D,FFFF
602A,602B,602C	CD,15,06		CALL	DELAY 0FFF
602D,602E,602F	CD,15,06		CALL	DELAY 0615
6030,6031,6032	CD,15,06		CALL	DELAY 0615
6033	C9		RET	

STEPPER MOTOR (ANTI-CLOCKWISE)

6000,6001	3E,80		MVI	A,08
6002,6003	D3,0B		OUT	0B
6004,6005	3E,00		MVI	A,00
6006,6007	D3,08		OUT	08
6008,6009	3E,08	START	MVI	A,08
600A,600B	D3,08		OUT	08
600C,600D,600E	CD,27 60		CALL	DELAY
600F,6010	3E,04		MVI	A,04
6011,6012	D3,08		OUT	08
6013,6014,6015	CD,27,60		CALL	DELAY 1
6016,6017	3E,02		MVI	A,02
6018,6019	D3,08		OUT	08
610A,610B,610C	CD,27,60		CALL	DELAY 1
610D,610E	3E,01		MVI	A,01
601F,6020	D3,08		OUT	08
6021,6022,6023	CD,27,60		CALL	DELAY 1
6024,6025,6026	C3,08,60		JMP	START
6027,6028,6029	11,FF,FF	DELAY	LXI	D,FFFF
602A,602B,602C	CD,15,06		CALL	DELAY 0FFF
602D,602E,602F	CD,15,06		CALL	DELAY 0615
6030,6031,6032	CD,15,06		CALL	DELAY 0615
6033	C9		RET	

Exp 16: INTERFACING OF DAC

Aim: To generate different waveforms through DAC using 8085 and 8255.

Apparatus required: 8085 Microprocessor trainer kit, DAC interface board, Regulated Power supply, CRO.

Description: Seven segment display interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode the square, triangular, sawtooth and staircase can be displayed on CRO.

DAC SQUARE WAVE

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND
6100,6101	3E,80		MVI	A,80
6102,6103	D3,0B		OUT	0B
6104,6105	3E,00	UP	MVI	A,00
6106,6107	D3,08		OUT	08
6108,6109	06,D7		MVI	B,D7
610A	05	XYZ	DCR	B
610B,610C,610D	C2,0A,61		JNZ	XYZ
610E,610F	3E,FF		MVI	A,FF
6110,6111	D3,08		OUT	08
6112,6113	06,D7		MVI	B,D7
6114	05	ABC	DCR	B
6115,6116,6117	C2,14,61		JNZ	ABC
6118,6119,611A	C3,04,61		JMP	UP

Exp 17: INTERFACING OF TRAFFIC LIGHT CONTROLLER

Aim: To control the traffic light system using 8085 and 8255.

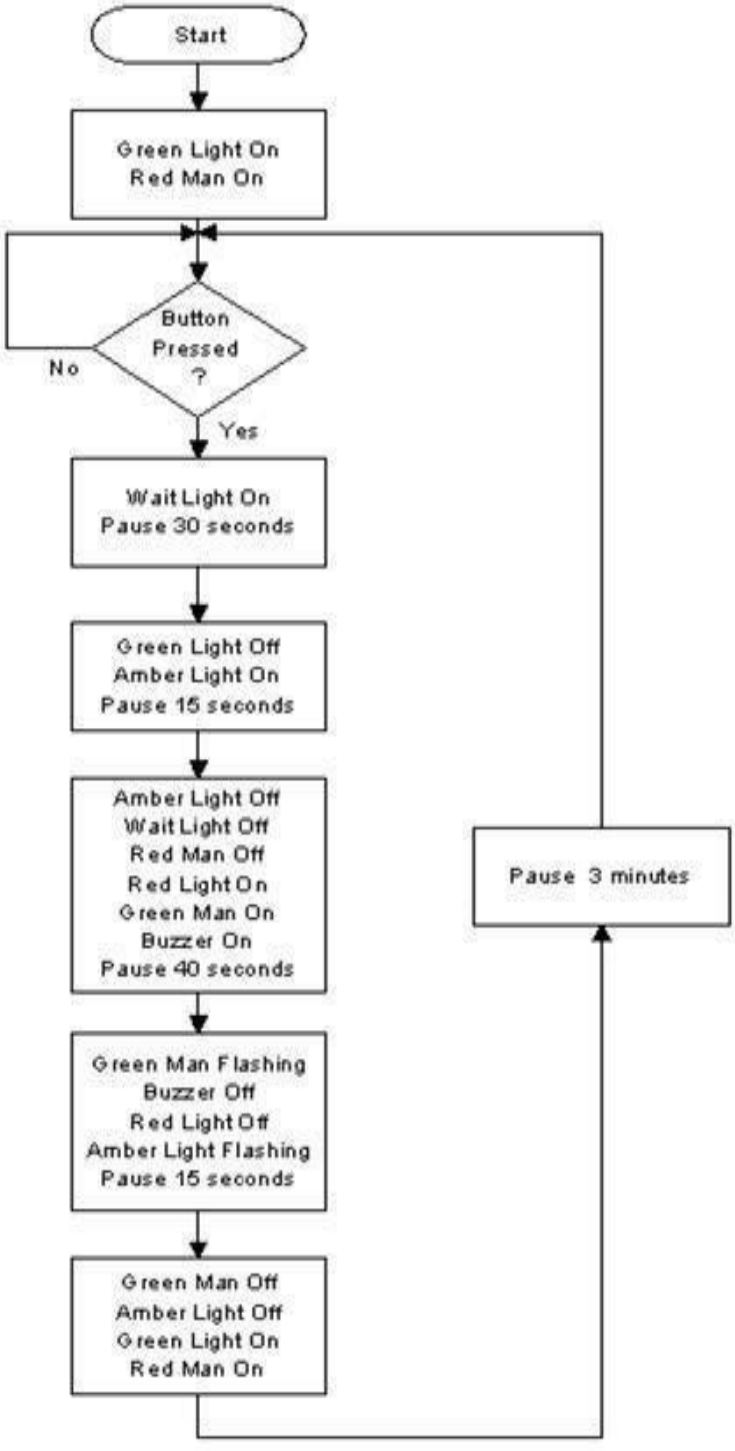
Apparatus required:
8085 Microprocessor trainer kit, Traffic light interface board, Regulated Power supply.

Description: The traffic system moves from one state to next state. By changing the delay between two signals one can change the speed of traffic.

8255 port addresses:

PORT A - 40H PORT B - 41H

PORT C - 42H Control word register -43H

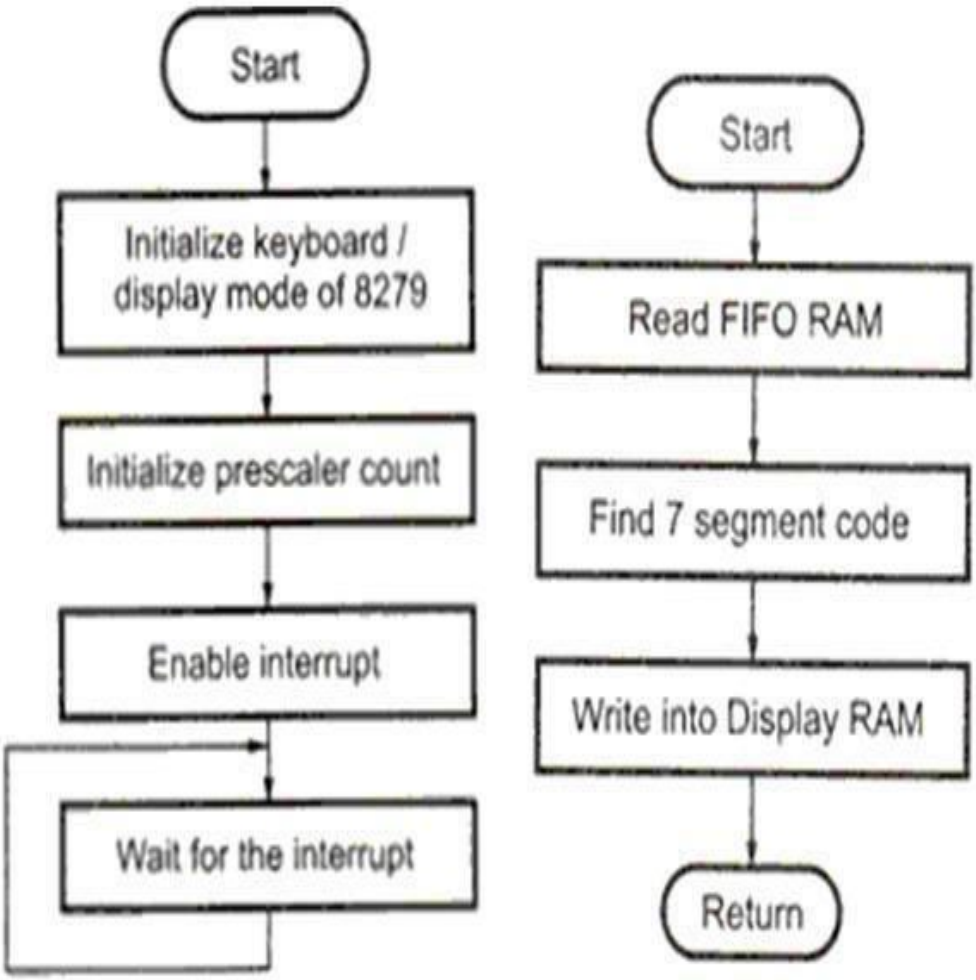

Start
Green Light On
Red Man On
Button
Pressed
?
No
Yes
Wait Light On
Pause 30 seconds
Green Light Off
Amber Light On
Pause 15 seconds
Amber Light Off
Wait Light Off
Red Man Off
Red Light On
Green Man On
Buzzer On
Pause 40 seconds
Green Man Flashing
Buzzer Off
Red Light Off
Amber Light Flashing
Pause 15 seconds
Green Man Off
Amber Light Off
Green Light On
Red Man On
Pause 3 minutes

Exp 18: INTERFACING OF 8279

Aim: To interface 8279 to 8085 microprocessor.

Apparatus required:
8085 Microprocessor trainer kit, 8279 keyboard/display interface board, Regulated Power supply.

Description: 8279 keyboard/display interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode the pressed key can be displayed on display unit.



Exp 19: DISPLAY INTERFACE

Aim: To display LED'S using 8085 and 8255.

Apparatus required: 8085 Microprocessor trainer kit, LED interface board, Regulated Power supply.

Description: LED interface is connected over J2 of the trainer. When the trainer kit in KEYBOARD or SERIAL mode it scans system key codes.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND
6200,6201	3E,80		MVI	A,80
6202,6203	D3,0B		OUT	0B
6204,6205	3E,00	UP	MVI	A,00
6206,6207	D3,0A		OUT	0A
6208,6209,620A	CD,CE,7D		CALL	DELAY
620B,620C	3E,01		MVI	A,01
620D,620E	D3,0A		OUT	0A
620F,6210,6211	CD,15,62		CALL	DELAY
6212,6213,6214	C3,04,62		JMP	UP
6215,6216,6217	11,FF,FF	DELAY	LXI	D,FFFF
6218,6219,621A	CD,FF,0F		CALL	DELAY (0FFF)
621B,621C,621D	CD,FF,0F		CALL	DELAY (0FFF)
621E,621F,6220	CD,FF,0F		CALL	DELAY (0FFF)
6221	C9		RET	