ST.ANNE'S COLLEGE OF ENGINEERING AND TECHNOLOGY

ANGUCHETTYPALAYAM

DEPARTMENT ELECTRICAL AND ELECTRONICS ENGINEERING

EE6612- MICROPROCESSOR AND MICROCONTROLLER LAB MANUAL

VI SEMESTER

NAME: _____

YEAR/SEM: _____BRANCH:_____

REG. NO.:_____

Prepared by
Mr. R.RADHAKRISHNAN, AP/**ECE** **HOD**

SYLLABUS
EE6612 MICROPROCESSORS AND MICROCONTROLLERS LABORATORY

OBJECTIVES:

To provide training on programming of microprocessors and microcontrollers and understand the interface requirements.

LIST OF EXPERIMENTS:

1. Simple arithmetic operations: addition / subtraction / multiplication / division.
2. Programming with control instructions:
(i) Ascending / Descending order, Maximum / Minimum of numbers
(ii) Programs using Rotate instructions
(iii) Hex / ASCII / BCD code conversions.
3. Interface Experiments: with 8085
(i) A/D Interfacing. & D/A Interfacing.
4. Traffic light controller.
5. I/O Port / Serial communication
6. Programming Practices with Simulators/Emulators/open source
7. Read a key interface display
8. Demonstration of basic instructions with 8051 Micro controller execution, including:
(i) Conditional jumps, looping
(ii) Calling subroutines.
9. Programming I/O Port 8051
(i) Study on interface with A/D & D/A
(ii) Study on interface with DC & AC motor.
10. Mini project development with processors.

TOTAL: 45 PERIODS

EXPT NO: **8085 PROGRAMMING** **DATE:**

1(A). 8 BIT - ADDITION

AIM:

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

Apparatus required:

S.NO	Apparatus required	Quantity
1	8085 Microprocessor kit	1
2	Power supply	
3	Opcode sheet	1

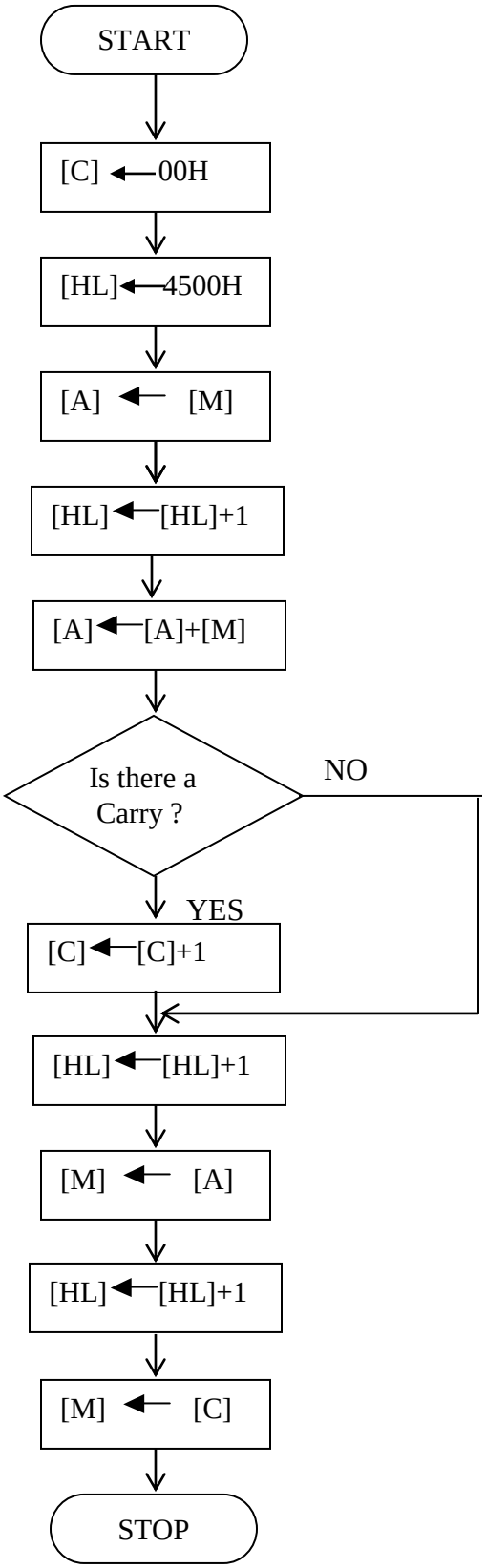
ALGORITHM:

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.

OUTPUT: WITH CCARRY

INPUT		OUTPUT	
4500	9A	4502	56
4501	BC	4503	01

FLOW CHART:


START
[C] ← 00H
[HL] ← 4500H
[A] ← [M]
[HL] ← [HL]+1
[A] ← [A]+[M]
Is there a Carry ?
NO
YES
[C] ← [C]+1
[HL] ← [HL]+1
[M] ← [A]
[HL] ← [HL]+1
[M] ← [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	MVI	C, 00	Clear C reg.
4101					
4102			LXI	H, 4500	Initialize HL reg. to 4500
4103					
4104					
4105			MOV	A, M	Transfer first data to accumulator
4106			INX	H	Increment HL reg. to point next memory Location.
4107			ADD	M	Add first number to acc. Content.
4108			JNC	L1	Jump to location if result does not yield carry.
4109					
410A					
410B			INR	C	Increment C reg.
410C		L1	INX	H	Increment HL reg. to point next memory Location.
410D			MOV	M, A	Transfer the result from acc. to memory.
410E			INX	H	Increment HL reg. to point next memory Location.
410F			MOV	M, C	Move carry to memory
4110			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
4500		4502	
4501		4503	

RESULT:

Thus the 8 bit numbers stored at 4500 &4501 are added and the result stored at 4502 & 4503.

1(B). 8 BIT DATA SUBTRACTION

AIM:

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

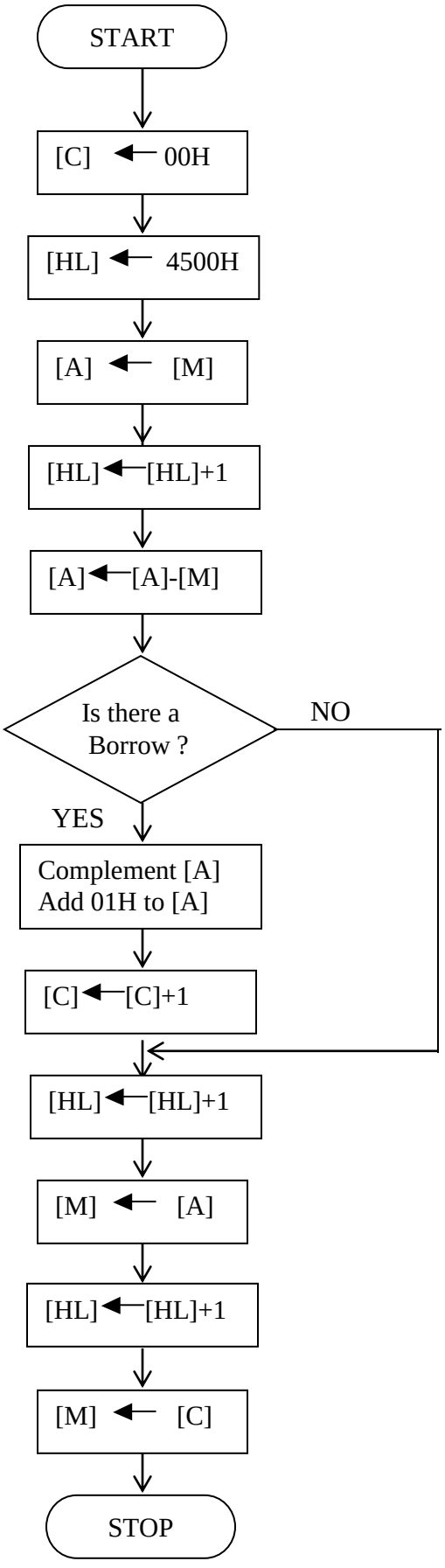
ALGORITHM:

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.

OUTPUT: WITH BORROW

INPUT		OUTPUT	
4500		4502	
4501		4503	

FLOW CHART:


START
[C] ← 00H
[HL] ← 4500H
[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
4100		START	MVI	C, 00	Clear C reg.
4102					
4102			LXI	H, 4500	Initialize HL reg. to 4500
4103					
4104					
4105			MOV	A, M	Transfer first data to accumulator
4106			INX	H	Increment HL reg. to point next mem. Location.
4107			SUB	M	Subtract first number from acc. Content.
4108			JNC	L1	Jump to location if result does not yield borrow.
4109					
410A					
410B			INR	C	Increment C reg.
410C			CMA		Complement the Acc. Content
410D			ADI	01H	Add 01H to content of acc.
410E					
410F		L1	INX	H	Increment HL reg. to point next mem. Location.
4110			MOV	M, A	Transfer the result from acc. to memory.
4111			INX	H	Increment HL reg. to point next mem. Location.
4112			MOV	M, C	Move carry to mem.
4113			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
4500		4502	
4501		4503	

RESULT: Thus the 8 bit numbers stored at 4500 &4501 are subtracted and the result stored at 4502 & 4503.

1(C). 8 BIT DATA MULTIPLICATION

AIM:

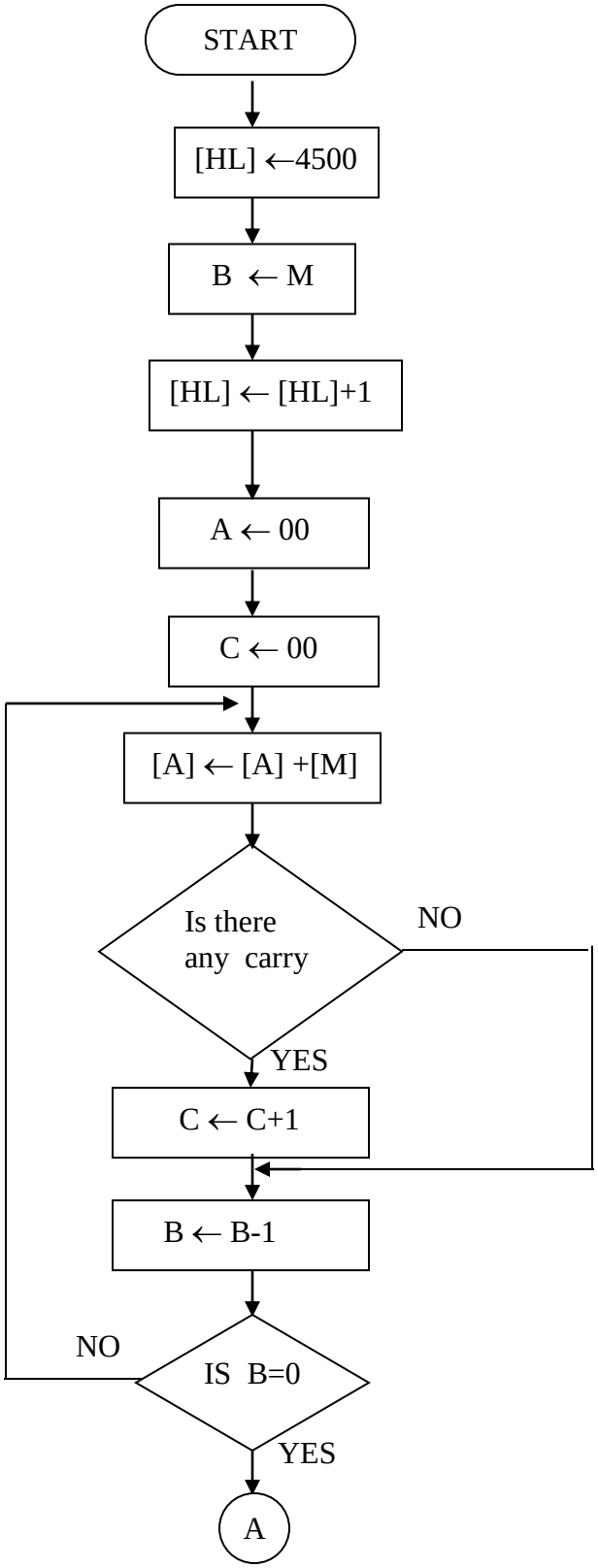
To multiply two 8 bit numbers stored at consecutive memory locations and store the result in memory.

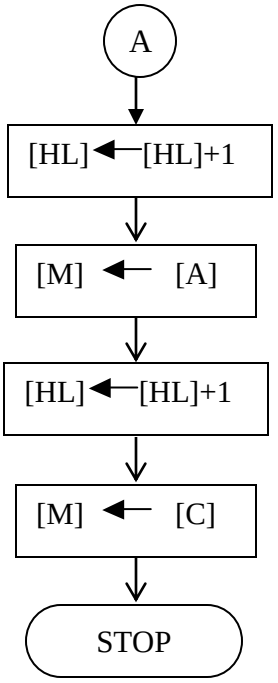
ALGORITHM:

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.

FLOW CHART:


START
[HL] ←4500
B ← M
[HL] ← [HL]+1
A ← 00
C ← 00
[A] ← [A] +[M]
Is there any carry
NO
YES
C ← C+1
B ← B-1
IS B=0
NO
YES
A


A
[HL] ← [HL]+1
[M] ← [A]
[HL] ← [HL]+1
[M] ← [C]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	LXI	H, 4500	Initialize HL reg. to 4500
4101					
4102					
4103			MOV	B, M	Transfer first data to reg. B
4104			INX	H	Increment HL reg. to point next mem. Location.
4105			MVI	A, 00H	Clear the acc.
4106					
4107			MVI	C, 00H	Clear C reg for carry
4108					
4109		L1	ADD	M	Add multiplicand multiplier times.
410A			JNC	NEXT	Jump to NEXT if there is no carry
410B					
410C					
410D			INR	C	Increment C reg
410E		NEXT	DCR	B	Decrement B reg
410F			JNZ	L1	Jump to L1 if B is not zero.
4110					
4111					
4112			INX	H	Increment HL reg. to point next mem. Location.
4113			MOV	M, A	Transfer the result from acc. to memory.
4114			INX	H	Increment HL reg. to point next mem. Location.
4115			MOV	M, C	Transfer the result from C reg. to memory.
4116			HLT		Stop the program

OBSERVATION: WITTHOUT BORROW

INPUT		OUTPUT	
4500		4502	
4501		4503	

WITH BORROW

INPUT		OUTPUT	
4500		4502	
4501		4503	

RESULT:

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

1(D). 8 BIT DIVISION

AIM:

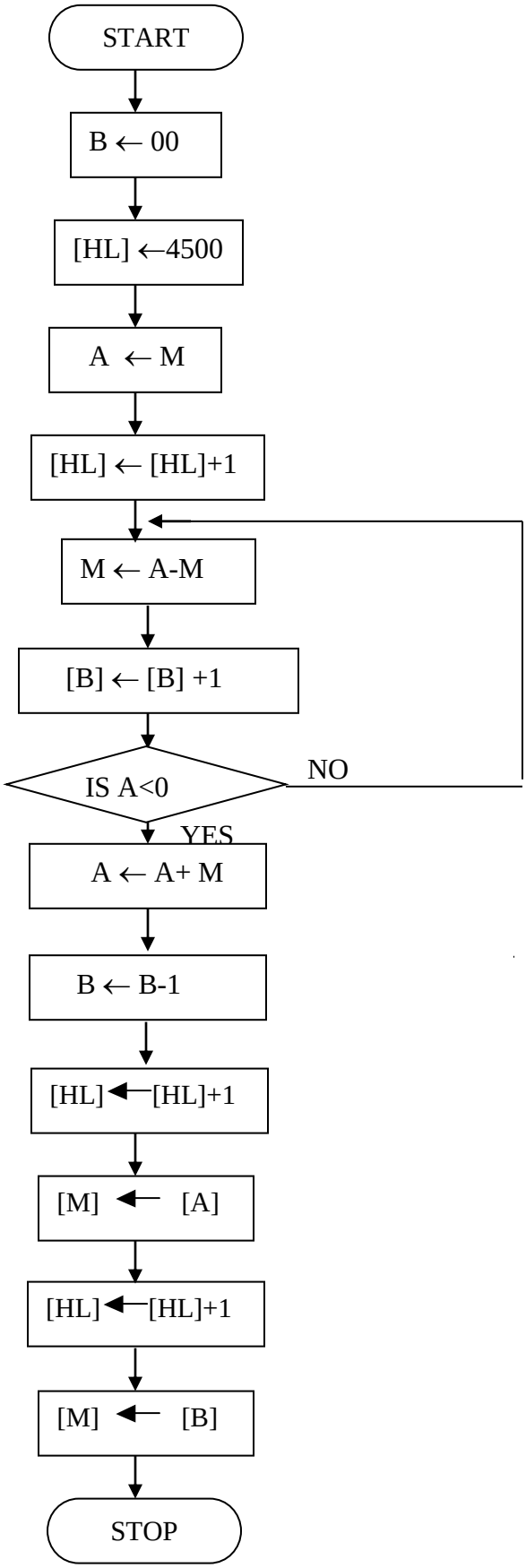
To divide two 8-bit numbers and store the result in memory.

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.

FLOWCHART:


START
B ← 00
[HL] ←4500
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
4100			MVI	B,00	Clear B reg for quotient
4101					
4102			LXI	H,4500	Initialize HL reg. to 4500H
4103					
4104					
4105			MOV	A,M	Transfer dividend to acc.
4106			INX	H	Increment HL reg. to point next mem. Location.
4107		LOOP	SUB	M	Subtract divisor from dividend
4108			INR	B	Increment B reg
4109			JNC	LOOP	Jump to LOOP if result does not yield borrow
410A					
410B					
410C			ADD	M	Add divisor to acc.
410D			DCR	B	Decrement B reg
410E			INX	H	Increment HL reg. to point next mem. Location.
410F			MOV	M,A	Transfer the remainder from acc. to memory.
4110			INX	H	Increment HL reg. to point next mem. Location.
4111			MOV	M,B	Transfer the quotient from B reg. to memory.
4112			HLT		Stop the program

OBSERVATION:

S.NO	INPUT		OUTPUT	
	ADDRESS	DATA	ADDRESS	DATA
1	4500		4502	
	4501		4503	
2	4500		4502	
	4501		4503	

RESULT:

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

2. PROGRAMMING WITH CONTROL INSTRUCTIONS:
I. (A). LARGEST ELEMENT IN AN ARRAY

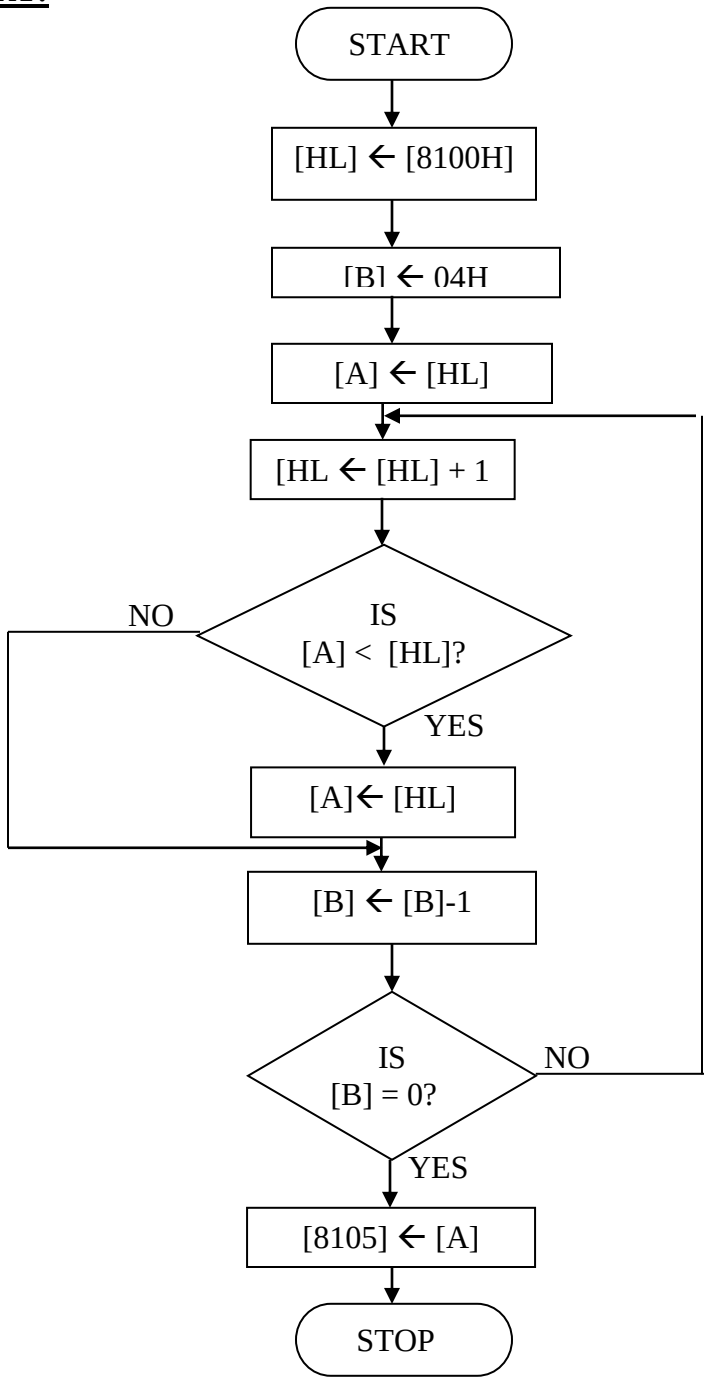
AIM:

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.

FLOW CHART:


START
[HL] ← [8100H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
NO
YES
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[8105] ← [A]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4101			LXI	H,4200	Initialize HL reg. to 8100H
4102					
4103					
4104			MVI	B,04	Initialize B reg with no. of comparisons(n-1)
4105					
4106			MOV	A,M	Transfer first data to acc.
4107		LOOP1	INX	H	Increment HL reg. to point next memory location
4108			CMP	M	Compare M & A
4109			JNC	LOOP	If A is greater than M then go to loop
410A					
410B					
410C			MOV	A,M	Transfer data from M to A reg
410D		LOOP	DCR	B	Decrement B reg
410E			JNZ	LOOP1	If B is not Zero go to loop1
410F					
4110					
4111			STA	4205	Store the result in a memory location.
4112					
4113					
4114			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200		4205	
4201			
4202			
4203			
4204			

RESULT:

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

(B). SMALLEST ELEMENT IN AN ARRAY

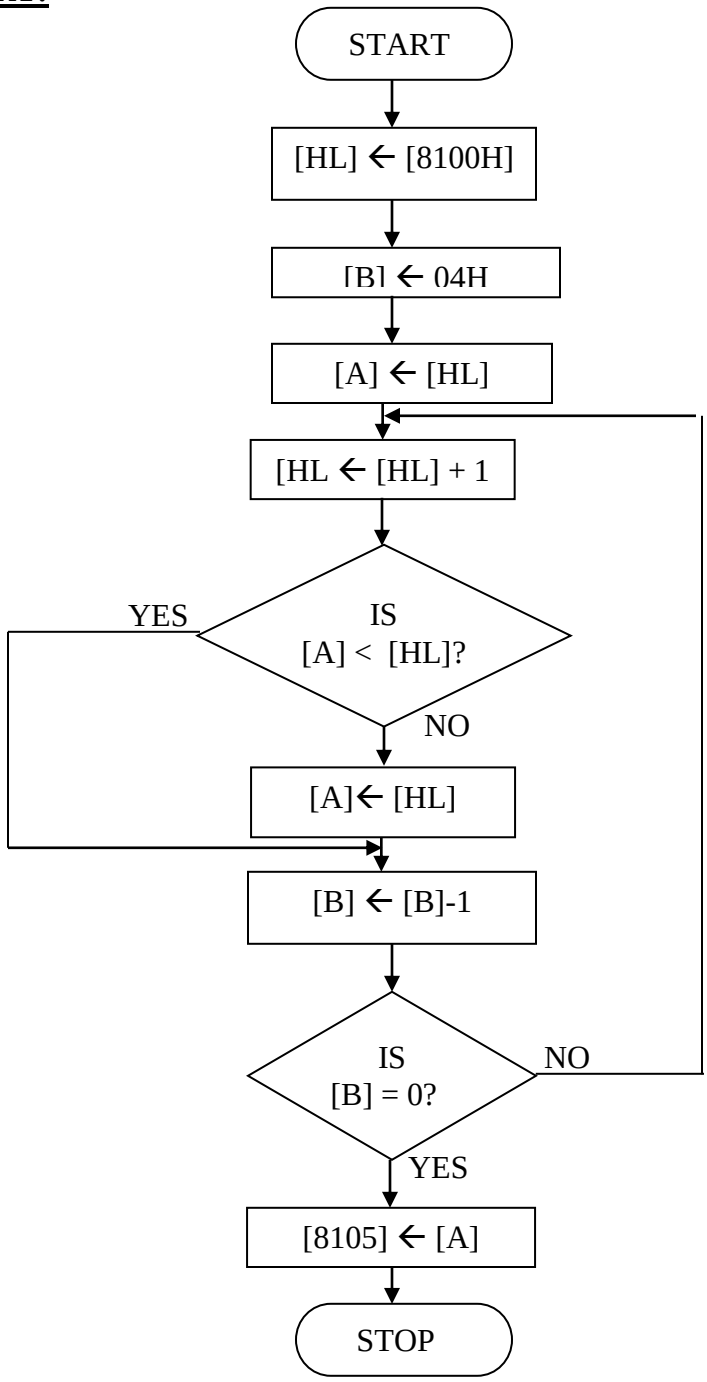
AIM:

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.

FLOW CHART:


START
[HL] ← [8100H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
YES
NO
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[8105] ← [A]
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4101			LXI	H,4200	Initialize HL reg. to 8100H
4102					
4103					
4104			MVI	B,04	Initialize B reg with no. of comparisons(n-1)
4105					
4106			MOV	A,M	Transfer first data to acc.
4107		LOOP1	INX	H	Increment HL reg. to point next memory location
4108			CMP	M	Compare M & A
4109			JC	LOOP	If A is lesser than M then go to loop
410A					
410B					
410C			MOV	A,M	Transfer data from M to A reg
410D		LOOP	DCR	B	Decrement B reg
410E			JNZ	LOOP1	If B is not Zero go to loop1
410F					
4110					
4111			STA	4205	Store the result in a memory location.
4112					
4113					
4114			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200		4205	
4201			
4202			
4203			
4204			

RESULT:

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

(C).ASCENDING ORDER

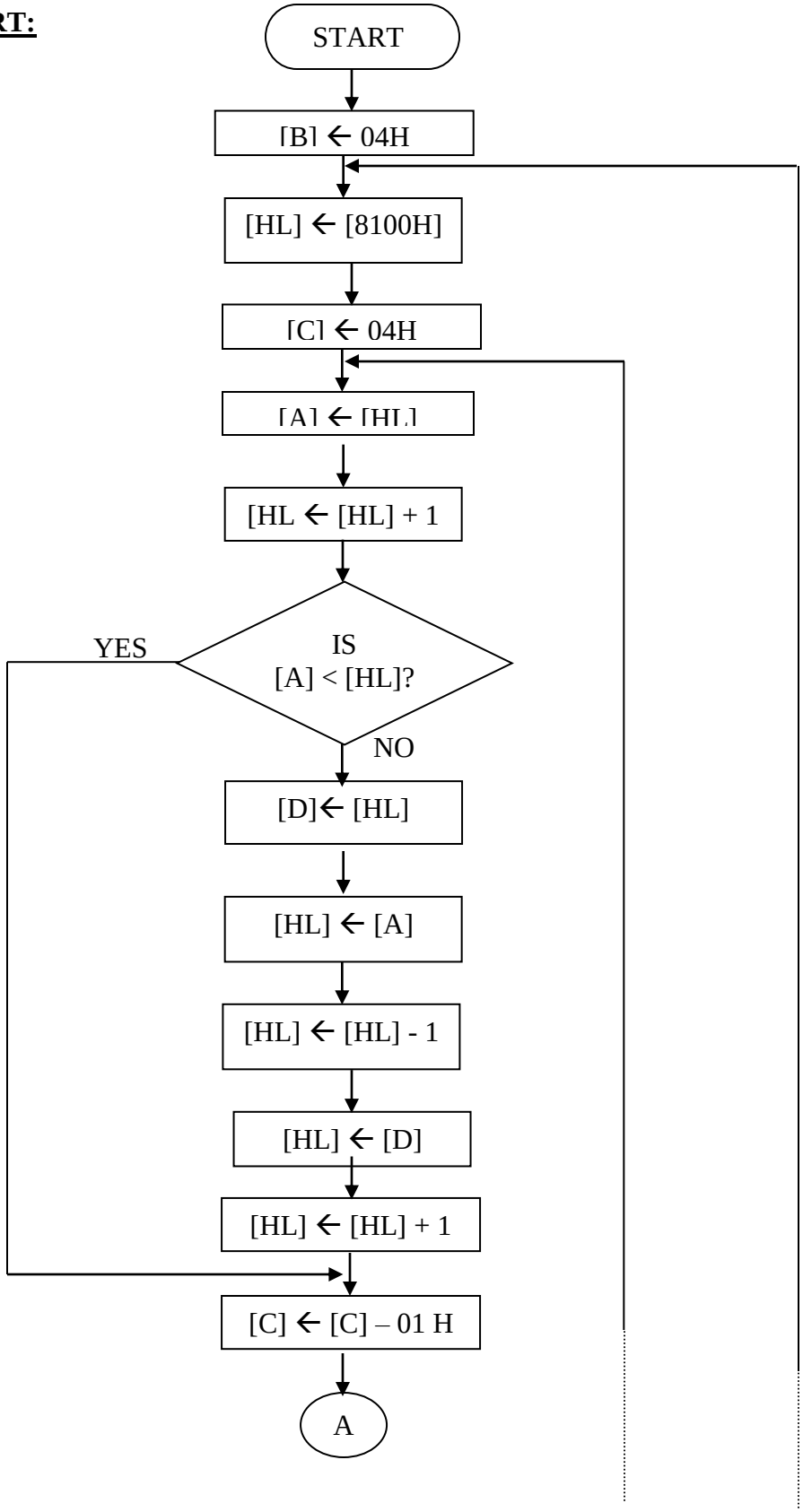
AIM:

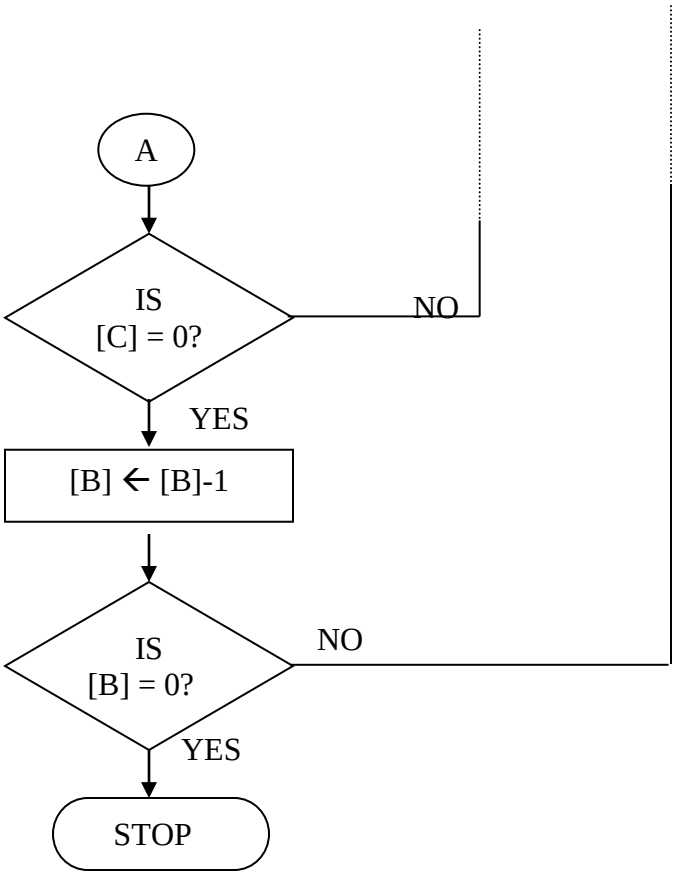
To sort the given number in the ascending order using 8085 microprocessor.

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 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

FLOWCHART:


START
[B] ← 04H
[HL] ← [8100H]
[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
4100			MVI	B,04	Initialize B reg with number of comparisons (n-1)
4101					
4102		LOOP 3	LXI	H,4200	Initialize HL reg. to 8100H
4103					
4104					
4105			MVI	C,04	Initialize C reg with no. of comparisons(n-1)
4106					
4107		LOOP2	MOV	A,M	Transfer first data to acc.
4108			INX	H	Increment HL reg. to point next memory location
4109			CMP	M	Compare M & A
410A			JC	LOOP1	If A is less than M then go to loop1
410B					
410C					
410D			MOV	D,M	Transfer data from M to D reg
410E			MOV	M,A	Transfer data from acc to M
410F			DCX	H	Decrement HL pair
4110			MOV	M,D	Transfer data from D to M
4111			INX	H	Increment HL pair
4112		LOOP1	DCR	C	Decrement C reg
4113			JNZ	LOOP2	If C is not zero go to loop2
4114					
4115					
4116			DCR	B	Decrement B reg
4117			JNZ	LOOP3	If B is not Zero go to loop3
4118					
4119					
411A			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
4200		4200	
4201		4201	
4202		4202	
4203		4203	
4204		4204	

RESULT:

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

(D). DESCENDING ORDER

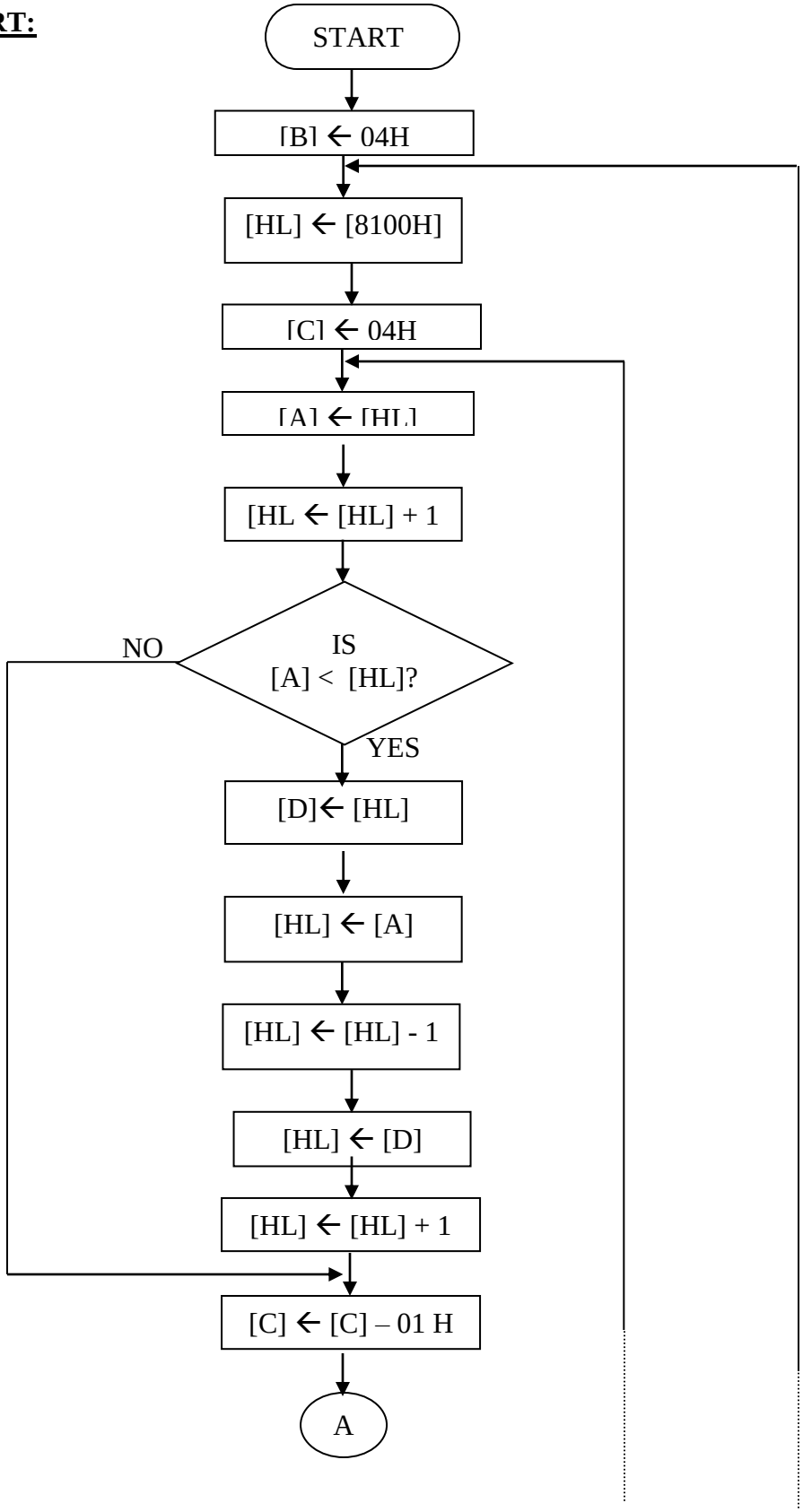
AIM:

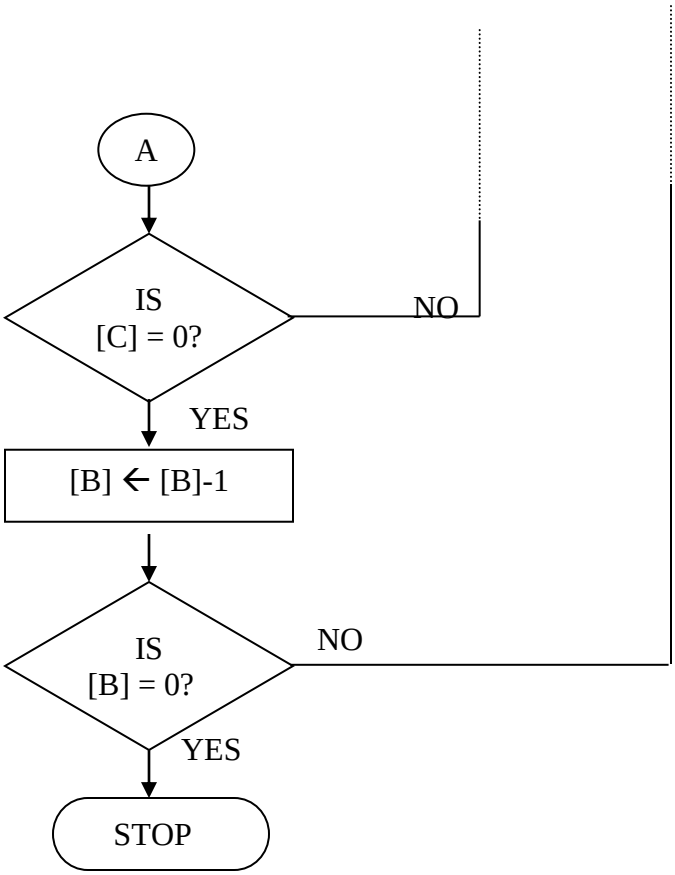
To sort the given number in the descending order using 8085 microprocessor.

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 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

FLOWCHART:


START
[B] ← 04H
[HL] ← [8100H]
[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
4100			MVI	B,04	Initialize B reg with number of comparisons (n-1)
4101					
4102		LOOP 3	LXI	H,4200	Initialize HL reg. to 8100H
4103					
4104					
4105			MVI	C,04	Initialize C reg with no. of comparisons(n-1)
4106					
4107		LOOP2	MOV	A,M	Transfer first data to acc.
4108			INX	H	Increment HL reg. to point next memory location
4109			CMP	M	Compare M & A
410A			JNC	LOOP1	If A is greater than M then go to loop1
410B					
410C					
410D			MOV	D,M	Transfer data from M to D reg
410E			MOV	M,A	Transfer data from acc to M
410F			DCX	H	Decrement HL pair
4110			MOV	M,D	Transfer data from D to M
4111			INX	H	Increment HL pair
4112		LOOP1	DCR	C	Decrement C reg
4113			JNZ	LOOP2	If C is not zero go to loop2
4114					
4115					
4116			DCR	B	Decrement B reg
4117			JNZ	LOOP3	If B is not Zero go to loop3
4118					
4119					
411A			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
4200		4200	
4201		4201	
4202		4202	
4203		4203	
4204		4204	

RESULT:

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

II. Programs using Rotate instructions

AIM:

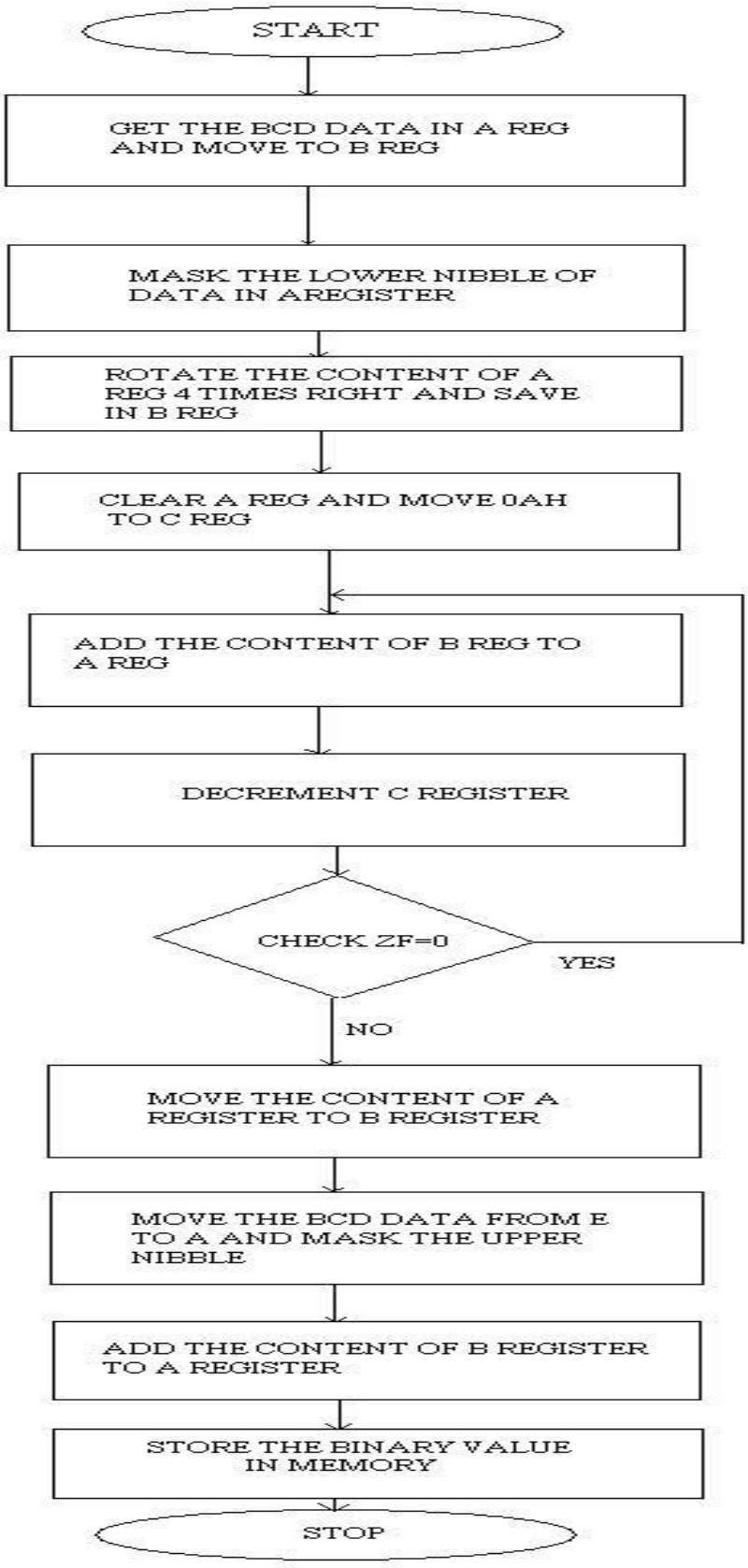
To write an assembly language program for rotate instruction using 8085 Microprocessor kit.

APPARATUS REQUIRED:

S.No	APPARATUS	QUANTITY
1	8085 Microprocessor Kit	1
2	Power Supply	-
3	Opcode Sheet	1

ALGORITHM FOR BCD TO BINARY CONVERSION:

1. Get the BCD data in A register and save in E register.
2. Mark the lower units of BCD data in A register.
3. Rotate the upper units to lower units' position and save in B register.
4. Clear the accumulator.
5. Move 0AH to C register.
6. Decrement C register.
7. If zf=0, go to the previous step.
8. Add B register to A register.
9. Save the product in B register.
10. Get the BCD data in A register from E register and mark the upper nibble.
11. Add the units in A register with the product in B register.
12. Store the binary value in A register.
13. Stop the program.


START
GET THE BCD DATA IN A REG AND MOVE TO B REG
MASK THE LOWER NIBBLE OF DATA IN AREGISTER
ROTATE THE CONTENT OF A REG 4 TIMES RIGHT AND SAVE IN B REG
CLEAR A REG AND MOVE 0AH TO C REG
ADD THE CONTENT OF B REG TO A REG
DECREMENT C REGISTER
CHECK ZF=0
YES
NO
MOVE THE CONTENT OF A REGISTER TO B REGISTER
MOVE THE BCD DATA FROM E TO A AND MASK THE UPPER NIBBLE
ADD THE CONTENT OF B REGISTER TO A REGISTER
STORE THE BINARY VALUE IN MEMORY
STOP

PROGRAM:

MEMORY ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
4100	START	MVI A,94H		
4102		RLC		
4103		STA 5000		
4106		RRC		
4107		STA 5001		
410A		RAR		
410B		STA 5002		
410E		RAL		
410F		STA 5003		
4112		HLT		

OUTPUT:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
4100		5000	
		5001	
		5002	
		5003	

RESULT:

Thus the conversions from BCD to Binary were obtained.

III. Hex / ASCII / BCD code conversions
(A). CODE CONVERSION – BCD TO HEX

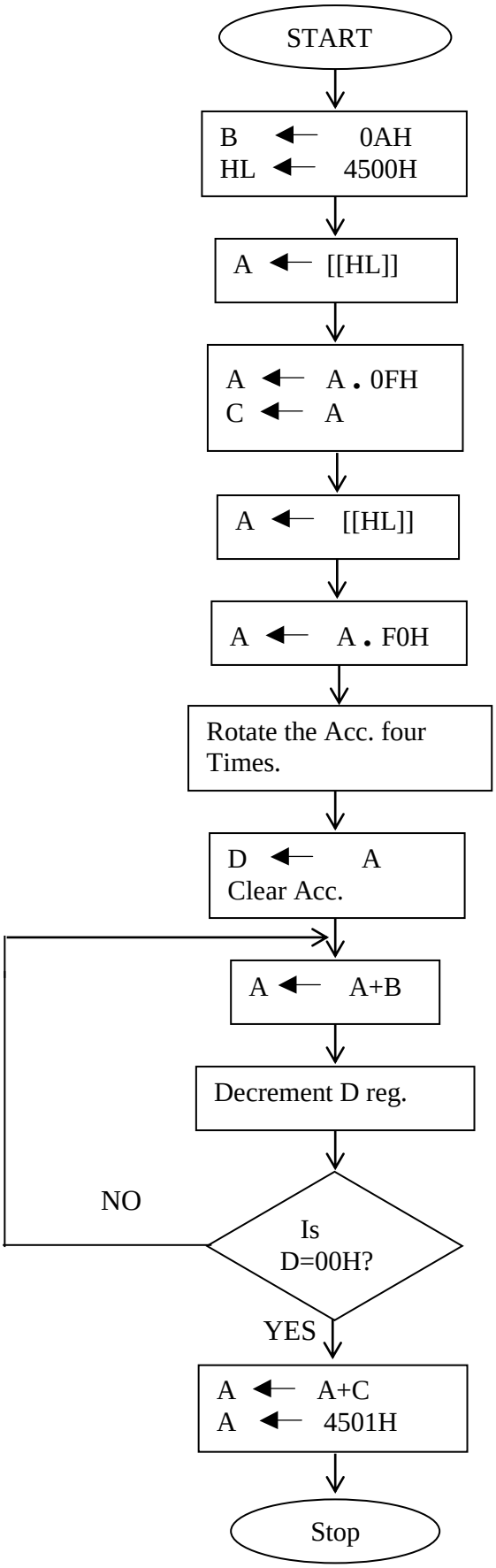
AIM:

To convert a given BCD number to hexadecimal.

ALGORITHM:

1. Initialize the memory location to the data pointer.
2. Get the BCD number from memory and separate LSB and MSB digits.
3. Multiply MSB No. of BCD to 0AH times and add the LSB to the resultant.
4. Store the resultant in memory location.

FLOWCHART:


START
B ← 0AH
HL ← 4500H
A ← [[HL]]
A ← A . 0FH
C ← A
A ← [[HL]]
A ← A . F0H
Rotate the Acc. four Times.
D ← A
Clear Acc.
A ← A+B
Decrement D reg.
NO
Is D=00H?
YES
A ← A+C
A ← 4501H
Stop

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4300		START	LXI	H,5000	Initialize HL reg to 5000
4301					
4302					Move M to Acc Add to Acc Move Acc to b
4303			MOV	A,M	
4304			ADD	A	
4305			MOV	B,A	
4306			ADD	A	Add a,b
4307			ADD	A	
4308			ADD	B	
4309			INX	H	Increment H
430A			ADD	M	
430B			INX	H	
430C			MOV	M,A	Stop progrm
430D			HLT		

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
5000		5002	
5001			

RESULT:

Thus an ALP program for conversion of BCD to HEX was written and executed.

(B). CODE CONVERSION –HEX TO BCD

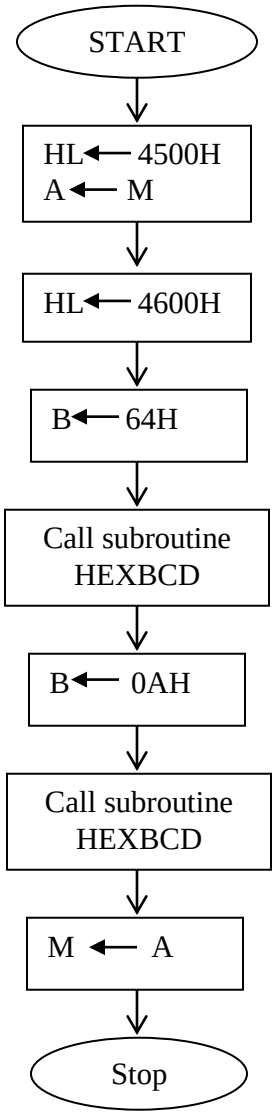
AIM:

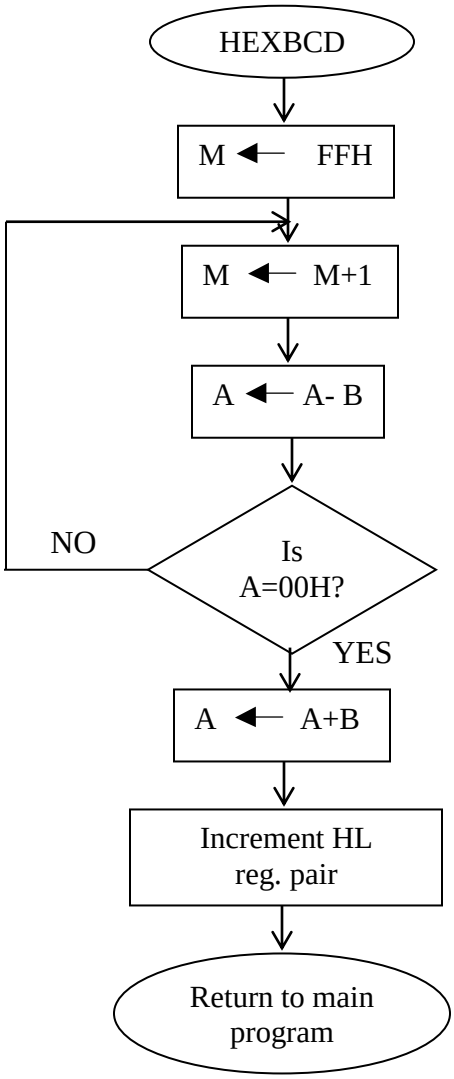
To convert a given number hexadecimal to BCD

ALGORITHM:

1. Initialize the memory location to the data pointer.
2. Get the HEX number from memory.
3. Initialize the memory to store the output.
4. Subtract the given HEX No. by 64H(100_{BCD}) .Repeat the subtraction with the resultant & 64H and keep count until there is a carry.
5. Store the count, which is MSB of BCD in a memory location.
6. Subtract the 0AH (10_{BCD}) from the result of the previous step. Repeat the subtraction with the resultant & 0AH and keep count until there is a carry.
7. Store the count, which is next significant bit of BCD in next memory location.
8. Store the result of step 6, which is LSB of BCD in next memory location.

FLOW CHART:


START
HL ← 4500H
A ← M
HL ← 4600H
B ← 64H
Call subroutine
HEXBCD
B ← 0AH
Call subroutine
HEXBCD
M ← A
Stop


HEXBCD
M ← FFH
M ← M+1
A ← A- B
Is
A=00H?
NO
YES
A ← A+B
Increment HL
reg. pair
Return to main
program

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4300		START	LXI	H,5000	
4301					
4302					
4303			MVI	D,00	
4304					
4305			XRA	A	
4306			MOV	C,M	
4307		LOOP2	ADI	01	
4308					
4309			DAA		
430A			JNC	LOOP1	
430B					
430C					
430D			INR	D	
430E			DCR	C	
430F			JNZ	LOOP2	
4310					
4311					
4312			STA	5001	
4313					
4314					
4315			MOV	A,D	
4316			STA	5002	
4317					
4318					
4319			HLT		

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
5000		5001	
		5002	

RESULT:

Thus an ALP program for conversion of HEX to BCD was written and executed.

C. ASCII TO HEXADECIMAL AND HEXADECIMAL TO ASCII

AIM:

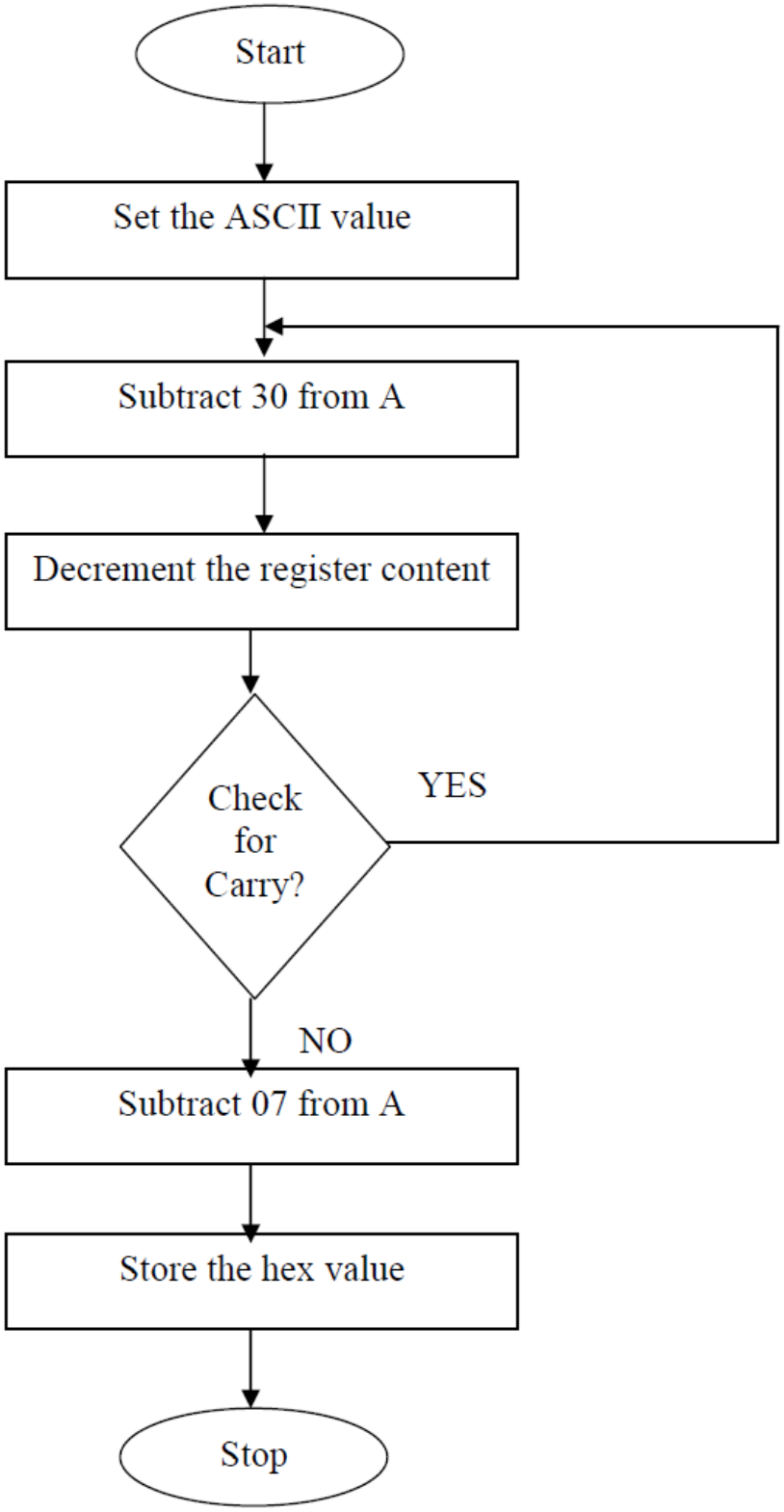
To Write an Assembly Language Program to Perform the Conversions of ASCII to Hexadecimal Number, Hexadecimal to ASCII,

A.ASCII TO HEXADECIMAL

ALGORITHM:

1. Start the program
2. Load the data from address 4200 to A
3. Move data from accumulator to C
4. Move data from M to HL pair to accumulator
5. Subtract the data 30 from A
6. Decrement content of register
7. Stop the program if C is zero
8. Jump to Step 5
9. End the program

FLOWCHART:


Start
Set the ASCII value
Subtract 30 from A
Decrement the register content
Check for Carry?
YES
NO
Subtract 07 from A
Store the hex value
Stop

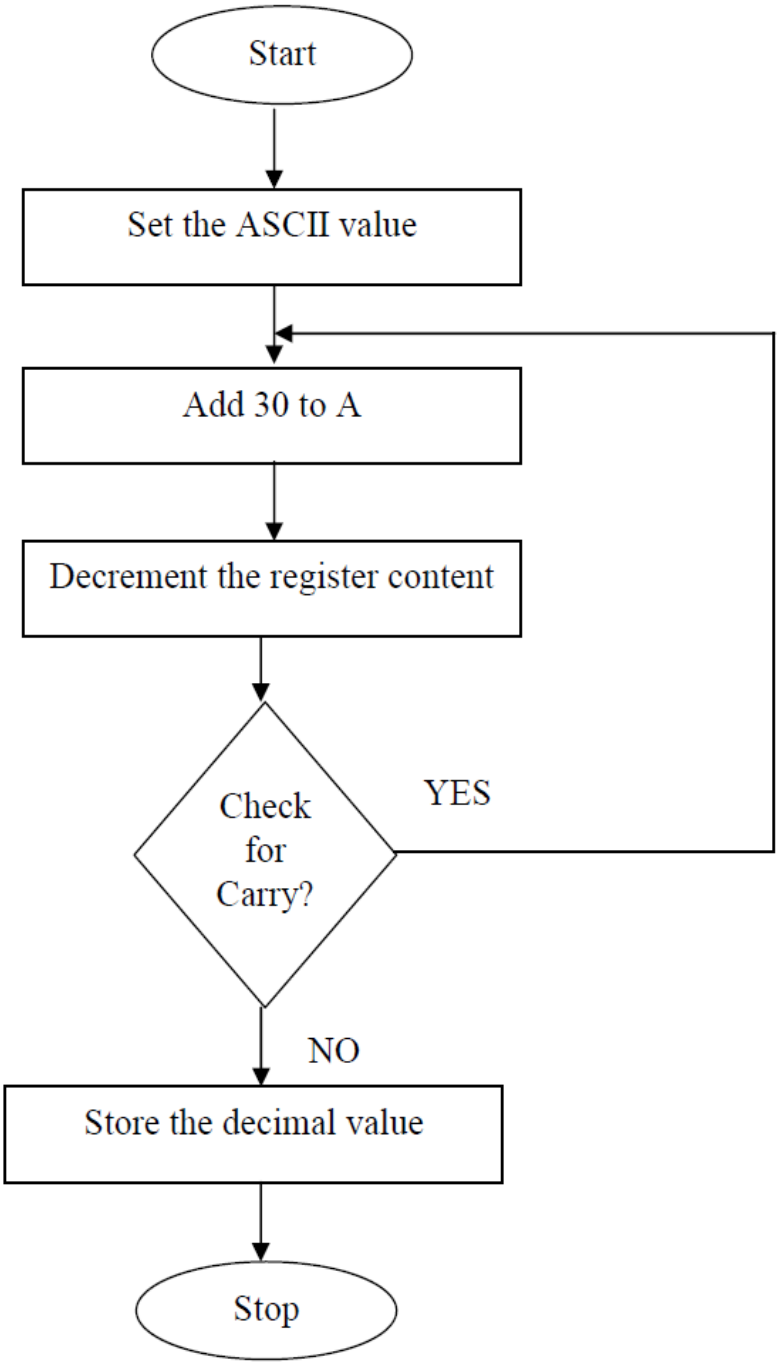
ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4300		START	LDA	5000	
4301					
4302					
4303			SUI	30	
4304					
4305			CPI	0A	
4306					
4307			JC	SKIP	
4308					
4309					
430A			SUI	07	
430B					
430C		SKIP	STA	5001	
430D					
430E					
430F			HLT		

B. HEXADECIMAL TO ASCII

ALGORITHM:

1. Start the program
2. Load the data from address 4200 to A
3. Move data from accumulator to C
4. Move data from M to HL pair to accumulator
5. Add the data 30 to A
6. Decrement content of register
7. Stop the program if C is zero
8. Jump to Step 5
9. End the program

FLOWCHART:


Start
Set the ASCII value
Add 30 to A
Decrement the register content
Check for Carry?
YES
NO
Store the decimal value
Stop

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4300		START	LDA	4400	
4301					
4302					
4303			CPI	0A	
4304					
4305			JC	AHEAD	
4306					
4307					
4308			ADI	07	
4309					
430A		AHEAD	ADI	30	
430B					
430C			STA	4401	
430D					
430E					
430F			HLT		

OBSERVATION:

ASCII TO HEXADECIMAL:

INPUT		OUTPUT	
MEMORY LOCATION	**DATA**	**MEMORY LOCATION**	**DATA**
4000		4001	

HEXADECIMAL TO ASCII:

INPUT		OUTPUT	
MEMORY LOCATION	**DATA**	**MEMORY LOCATION**	**DATA**
4400		4401	

RESULT:

Thus the assembly language programs for various code conversions are executed using 8085 microprocessor.

3. Interface Experiments: with 8085

(i) .DAC INTERFACING

AIM:

To write an assembly language program to convert a digital signal into an analog signal using DAC interfacing.

APPARATUS REQUIRED:

8085Trainer Kit
DAC Interface Board
Power supply

THEORY:

DAC 0800 is an 8–bit DAC and the output voltage variation is between –5V and 5V.The output voltage varies in steps of 10/256=0.04(appx.).The digital data input and The corresponding output voltages are presented in the Table below.

Input Data in HEX	Output Voltage
00	5.00
01	4.96
02	..
…	
7F	
…	
FD	
FE	4.96
FF	5.00

Referring toTable1, with 00 H as input to DAC, the analog output is –5 V.Similarly, With FF H input, the output is +5V.Outputting digital data 00 and FF at regular intervals, to DAC, results in different waveforms namely square, triangular, etc, the port address of DAC is 08H.

ALGORITHM:

(a) Square Wave Generation

1.Load the initial value (00) to Accumulator and move it to DAC
2.Call the delay program
3.Load the final value (FF)to accumulator and move it to DAC
4.Call the delay program.
5.Repeat Steps 2 to 5

(b) Saw tooth Wave Generation

1. Load the initial value(00) to Accumulator
2. Move the accumulator content to DAC
3. Increment the accumulator content by 1.
4. Repeat Steps 3 and 4.

(c) Triangular Wave Generation

1.Load the initial value (00) to Accumulator
2.Move the accumulator content to DAC
3.Increment the accumulator content by1.
4.If accumulator content is zero proceed to next step. Else go to step3.
5.Load value (FF) to Accumulator
6.Move the accumulator content to DAC
7.Decrement the accumulator content by1.
8.If accumulator content is zero go to step2. Else go to step7.

PROGRAM:

(a) Square Wave Generation

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
4100	START	MVI A, 00		Load the initial value (00) to Accumulator and move it to DAC
4102		OUT C8		
4104		CALL DELAY		Call the delay program
4107		MVI A, FF		Load the final value (FF)to accumulator and move it to DAC
4109		OUT C8		
410B		CALL DELAY		Call the delay program
410E		JMP START		
4111	DELAY	MVI B, 05		
4113	L1	MVI C, FF		
4115	L2	DCR C		
4116		JNZ L2		
4119		DCR B		
411A		JNZ L1		
411D		RET		

(B) Saw tooth Wave Generation

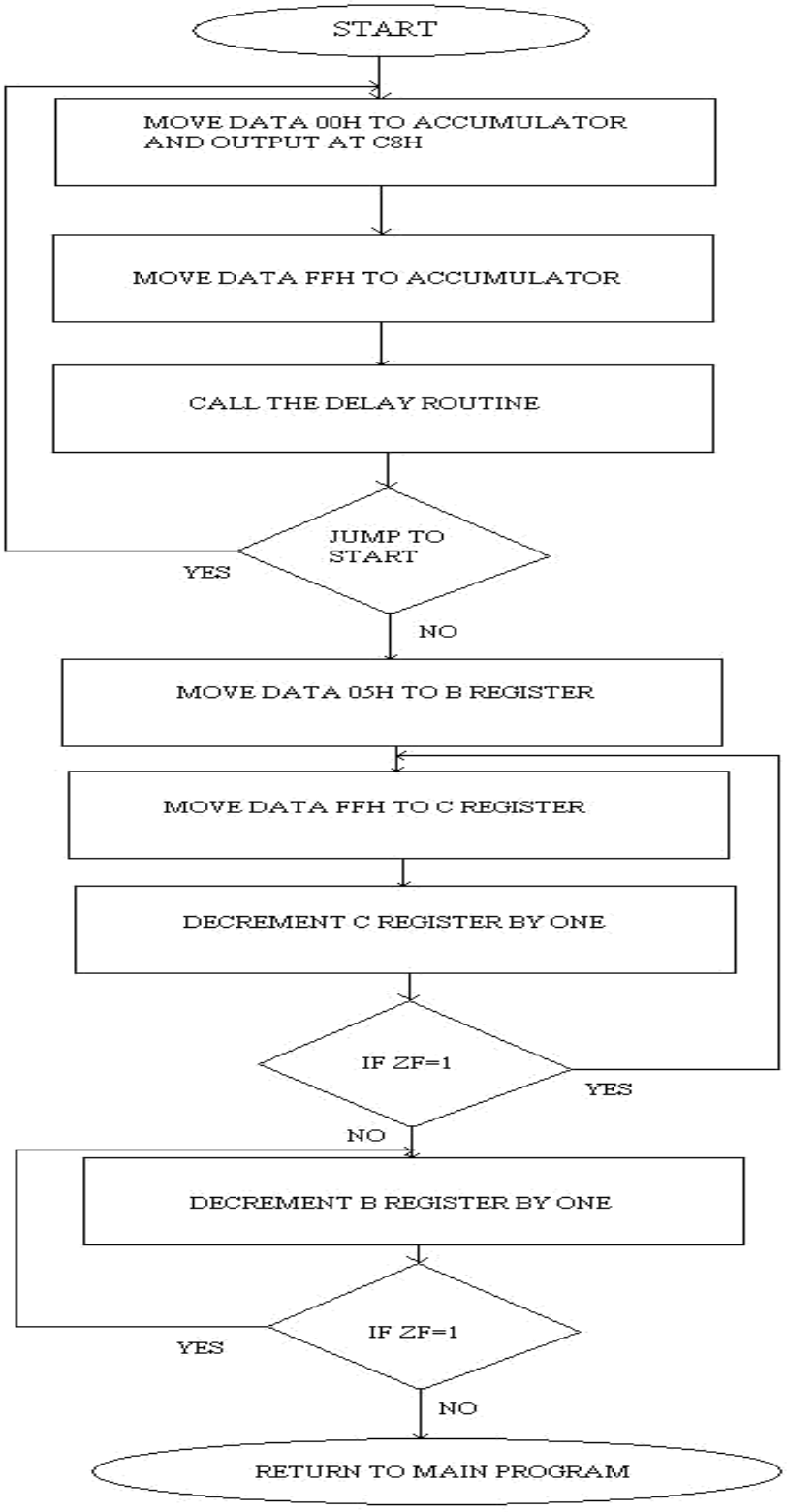
ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
4100	START	MVI		A,00	
4102	L1	OUT		C0	
4104		INR		A	
4105		JNZ		L1	
4108		JMP		START	

c) Triangular Wave Generation

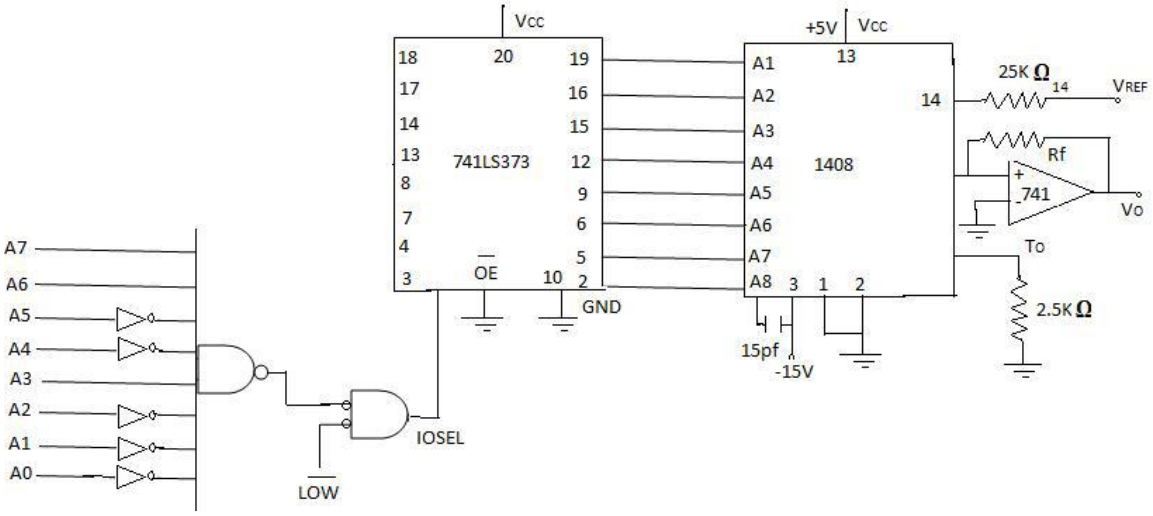
ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
4100	START	MVI L		00	Move 00 to L register
4101		MOV A,L			Load L to a register
4102	L1	OUT		C8	Load c8 to output port
4105		INR L			Increment L register
4106		JNZ		L1	Jump to L1 if no zero
4109		MVI L		FF	Load FF to L register
410B	L2	MOV A,L			Move L to a register
410C		OUT		C8	Load C8 to output port
410E		DCR L			Decrement L register
410F		JNZ		L2	Jump to L2 if no zero
4102		JMP		START	Go to START unconditionally

FLOWCHART:

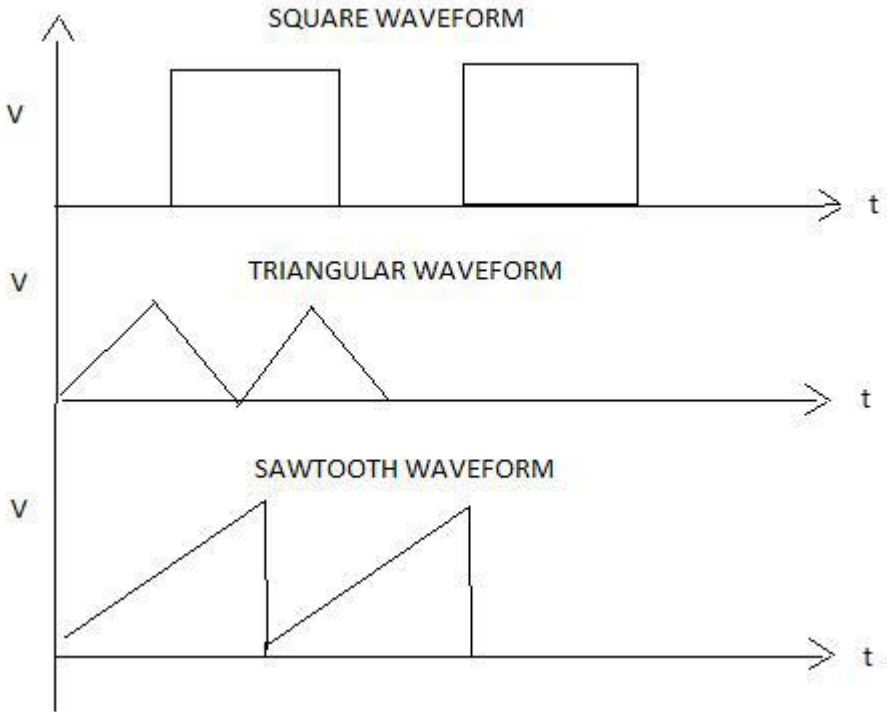
SQUARE:



DAC-CIRCUIT:



WAVEFORMS:



OBSERVATION:

WAVEFORMS	AMPLITUDE	TIME PERIOD
SQUARE		
SAWTOOTH		
TRIANGULAR		

RESULT:

Thus the square, triangular and saw tooth waveform were generated

by interfacing DAC with 8085 trainer kit.

B. ADC INTERFACING

AIM:

To write an assembly language program to convert an analog signal into digital signal using ADC interfacing.

THEORY:

An ADC usually has two additional control lines: the SOC input to tell the ADC when to start the conversion and the EOC output to announce when the conversion is complete. The following program initiates the conversion process, checks the EOC pin of ADC 0419 as to whether the conversion is over and then inputs the data to the processor. It also instructs the processor to store the converted digital data at RAM 4200H.

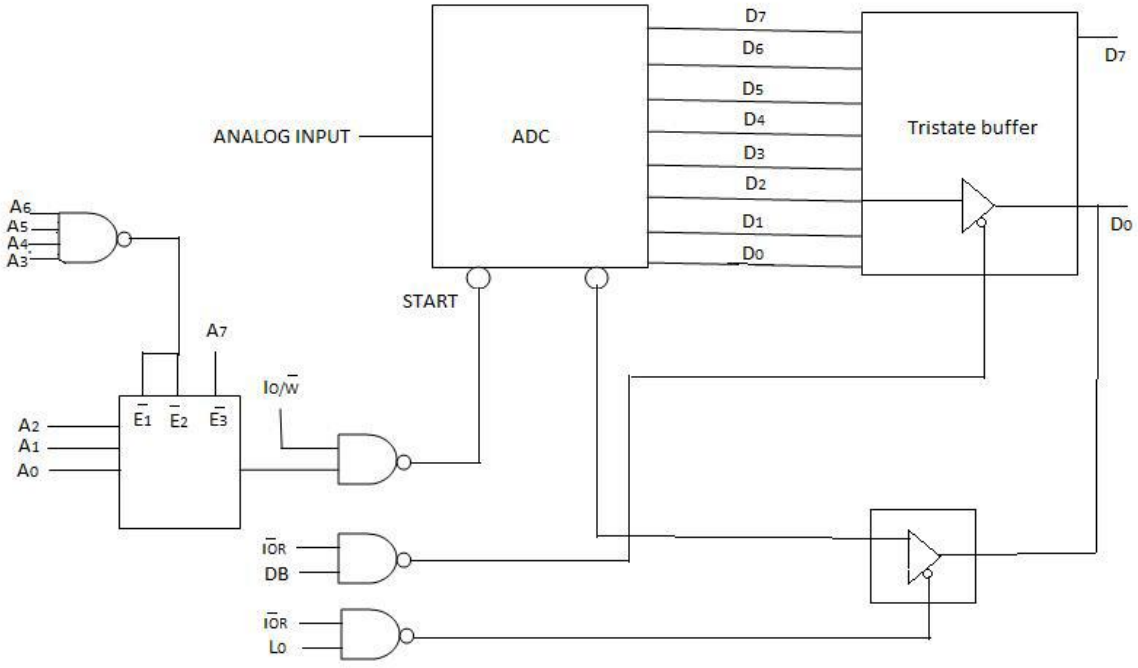
ALGORITHM:

1. Select the channel and latch the address.
2. Send the start conversion pulse.
3. Read EOC signal.
4. If EOC =1 continue else go to step (3)
5. Read the digital output.
6. Store it in a memory location.

PROGRAM:

ADDRESS	LABEL	MNEMON ICS	OPCO DE	OPERA ND	COMMENTS
4100		MVI A		10	Select channel 0 and to make accumulator low
4101					
4102		OUT		C8	Output the data
4103					
4104		MVI A		18	Make accumulator high
4105					
4106		OUT		C8	Display the data
4107					
4108		MVI A		01	Make 01 to accumulator
4109					
410A		OUT		D0	Display the data
410B					
410C		XRA			XOR with accumulator
410D		XRA			XOR with accumulator
410E		XRA			XOR with accumulator
410F		MVI A		00	Make 00 to accumulator
4110					
4111		OUT		D0	Load D0 in output port
4112					
4113	LOOP	IN		D8	
4114					
4115		ANI		01	Do and operation directly
4116					
4117		CPI		01	Compare with accumulator
4118					
4119		JNZ		LOOP	Jump to specified address
411A					
411B					
411C		IN		C0	
411D					
411E		STA		4150	Store the data
411F					
4120					
4121		HLT			End the program

CIRCUIT DIAGRAM:



ANALOG VOLTAGE	DIGITAL DATA ON LED DISPLAY	HEX CODE IN LOCATION 4150
0.5 V		
1.0 V		
1.5V		
2.0V		
2.5V		
3.0V		

RESULT:

Thus the analog to digital conversion is obtained using 8085 microprocessor.

4. TRAFFIC LIGHT CONTROLLER

AIM:

Write an ALP to control the traffic light signal using the microprocessor 8085.

THEORY:

A simple contraption of a traffic control system is shown in the figure where the signaling lights are simulated by the blinking or ON – OFF control of LED's. The signaling lights for the pedestrian crossing are simulated by the ON – OFF control of dual colour LED's. A model of a four road – four lane junction, the board has green, yellow and red LED's which are the green, orange and red signals of an actual systems. 12 LEDs are used on the board. In addition 8 dual colour LEDs are used which can be made to change either to red or to green.

The control of the LEDs is as follows:

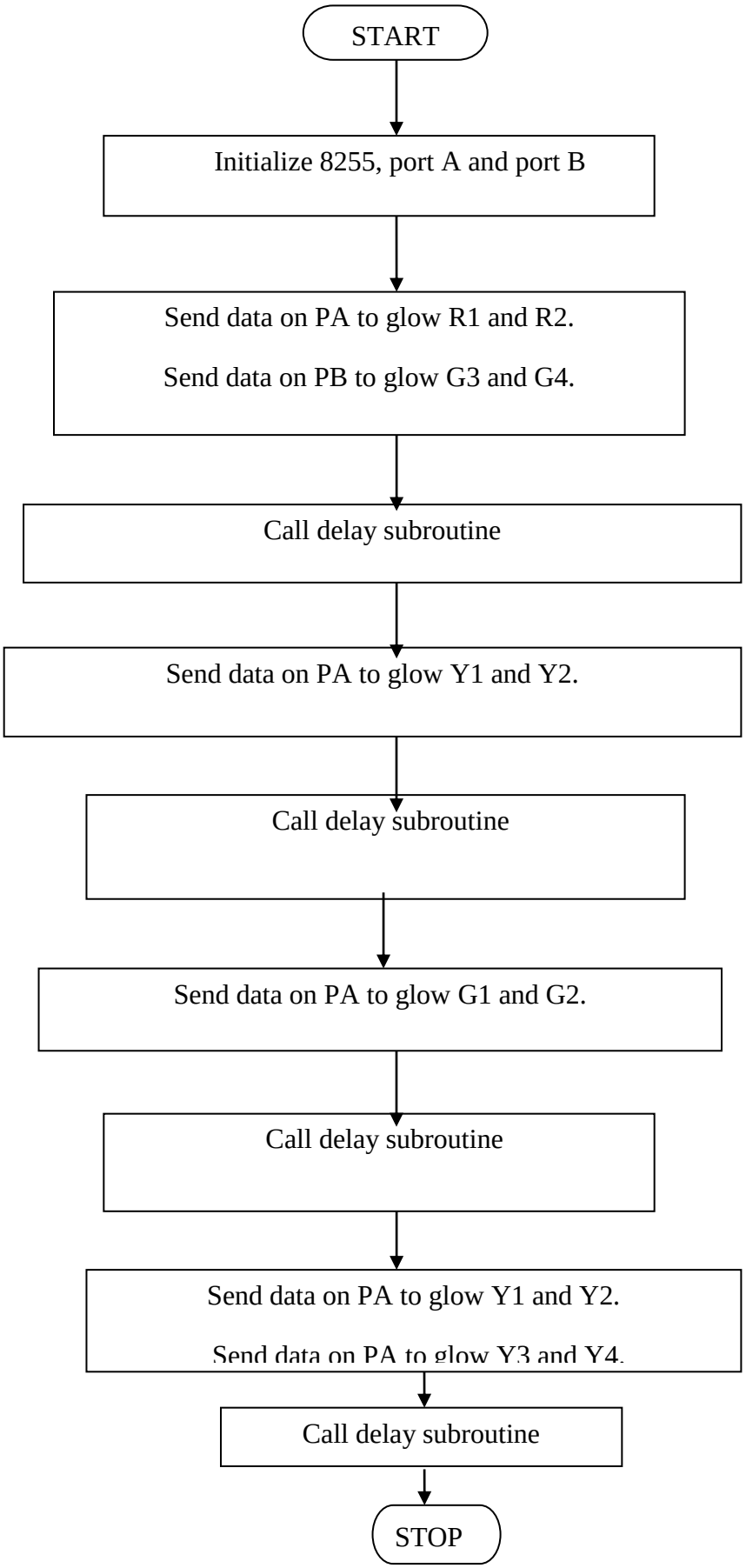
The board communicates with the microprocessor trainer by means of a 26 core cable which is connected to the output pins of any parallel port of trainer. The outputs (i.e. port) are the inputs to buffers 7406 whose outputs drive the LEDs. The buffered output applied to the cathode of the LEDs decides whether it is ON or OFF.

ALGORITHM:

1. Initialize 8255, port A and port B in output mode
2. Send data on PA to glow R1 and R2.
3. Send data on PB to glow G3 and G4.
4. Load multiplier count (40) for delay.
5. Call delay subroutine.
6. Send data on PA to glow Y1 and Y2.
7. Send data on PB to glow Y3 and Y4.
8. Load multiplier count (10) for delay.
9. Call delay subroutine.

10. Send data on PA to glow G1 and G2.
11. Send data on PB to glow R3 and R4.
12. Load multiplier count (40) for delay.
13. Call delay subroutine.
14. Send data on PA to glow Y1 and Y2.
15. Send data on PA to glow Y3 and Y4.
16. Load multiplier count (10) for delay.
17. Call delay subroutine

FLOWCHART:


START
Initialize 8255, port A and port B
Send data on PA to glow R1 and R2.
Send data on PB to glow G3 and G4.
Call delay subroutine
Send data on PA to glow Y1 and Y2.
Call delay subroutine
Send data on PA to glow G1 and G2.
Call delay subroutine
Send data on PA to glow Y1 and Y2.
Send data on PA to glow Y3 and Y4.
Call delay subroutine
STOP

Source program:

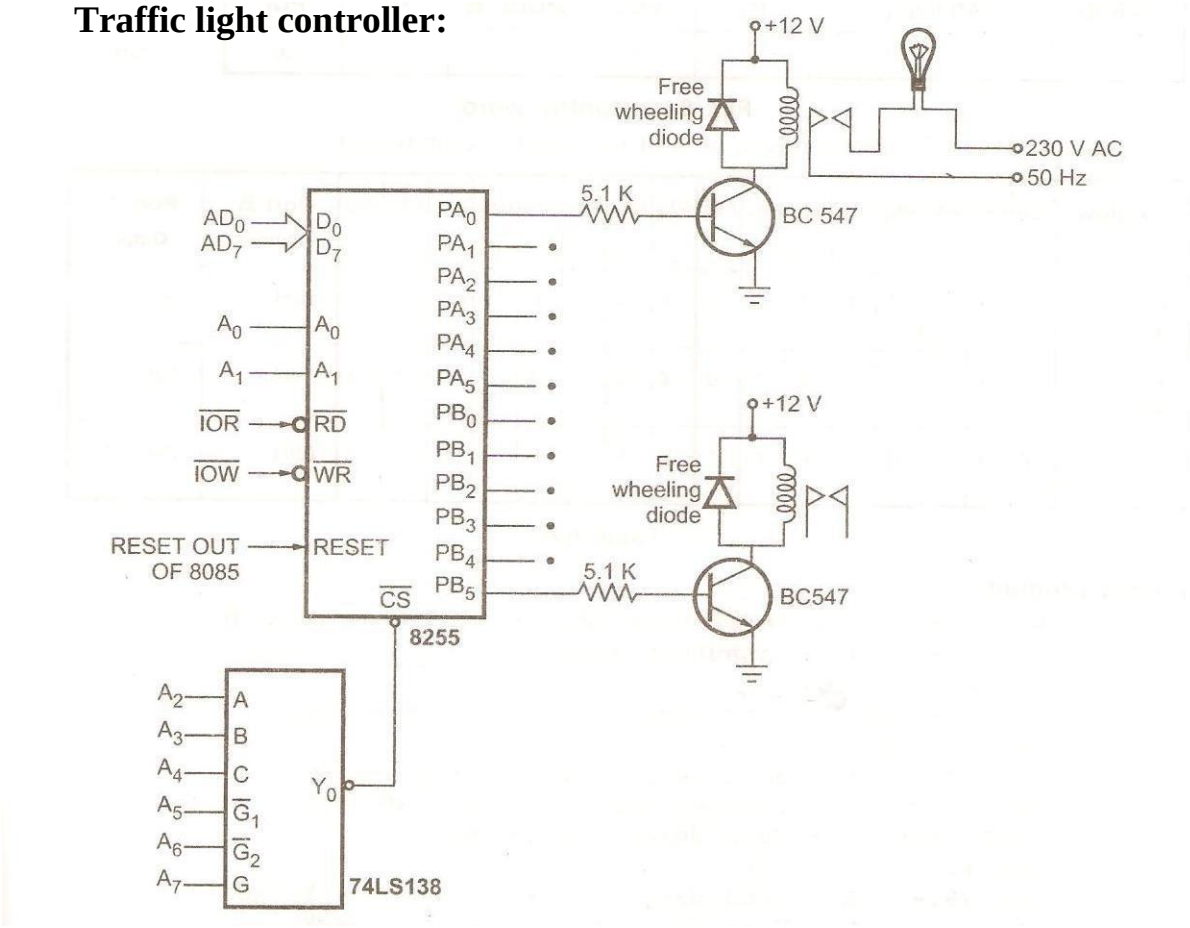
ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
8000		MVI A, 80 H	3E,80	Initialize 8255, port A and port B in output mode
8002		OUT 83H(CR)	D3,83	
8004	START	MVI A,09H	3E,09	Send data on PA to glow R1 and R2
8006		OUT 80H(PA)	D3,80	
8008		MVI A,24H	3E,24	
800A		OUT 81H(PB)	D3,81	Send data on PB to glow G3 and G4
800C		MVI C,28H	0E,28	Load multiplier count (40) for Delay
800E		CALL DELAY	CD,40,80	Call delay subroutine
8011		MVI A, 12H	3E,12	
8013		OUT (81H) PA	D3,81	Send data on PA to glow Y1 and Y2
8015		OUT (81H) PB	D3,81	Send data on PB to glow Y3 and Y4
8017		MVI C,OAH	0E,0A	Load multiplier count (10) for Delay
8019		CALL DELAY	CD,40,80	Call delay subroutine
801C		MVI A,24H	3E,24	
801E		OUT (80H) PA	D3,80	Send data on PA to glow G1 and G2
8020		MVI A, 09H	3E,09	
8022		OUT (81H) PB	D3,81	Send data on PB to glow R3 and R4
8024		MVI C,28H	0E,28	Load multiplier count (40) for

				delay
8026		CALL DELAY	CD,40,80	Call delay subroutine
8029		MVI A, 12H	3E,12	
802B		OUT PA	D3,80	Send data on PA to glow Y1 and Y2
802D		OUT PB	D3,81	Send data on PA to glow Y3 and Y4
802F		MVI C,OAH	0E,0A	Load multiplier count (10) for delay
803B		CALL DELAY	CD,40,80	Call delay subroutine
803F		JMP START	C3,04,80	

Delay subroutine:

ADDRESS	**LABEL**	**MNEMONICS**	**OPCODE**	**COMMENTS**
8040	DELAY	LXI D,COUNT	11,XXXX	Load the count to give 0.5 sec delay
8043	BACK	DCX D	1B	Decrement counter
8044		MOV A,D	7A	
8045		ORA E	B3	Check whether count is 0
8046		JNZ BACK	C2,43,80	If not zero, repeat
8049		DCR C	0D	Check if multiplier zero, otherwise repeat
804A		JNZ DELAY	C2,40,80	
804D		RET	C9	Return to main program

Traffic light controller:



RESULT:

Thus traffic light control is obtained using 8085 microprocessor

5. I/O PORT / SERIAL COMMUNICATION

(A) INTERFACING I/O (8255) WITH 8085

AIM:

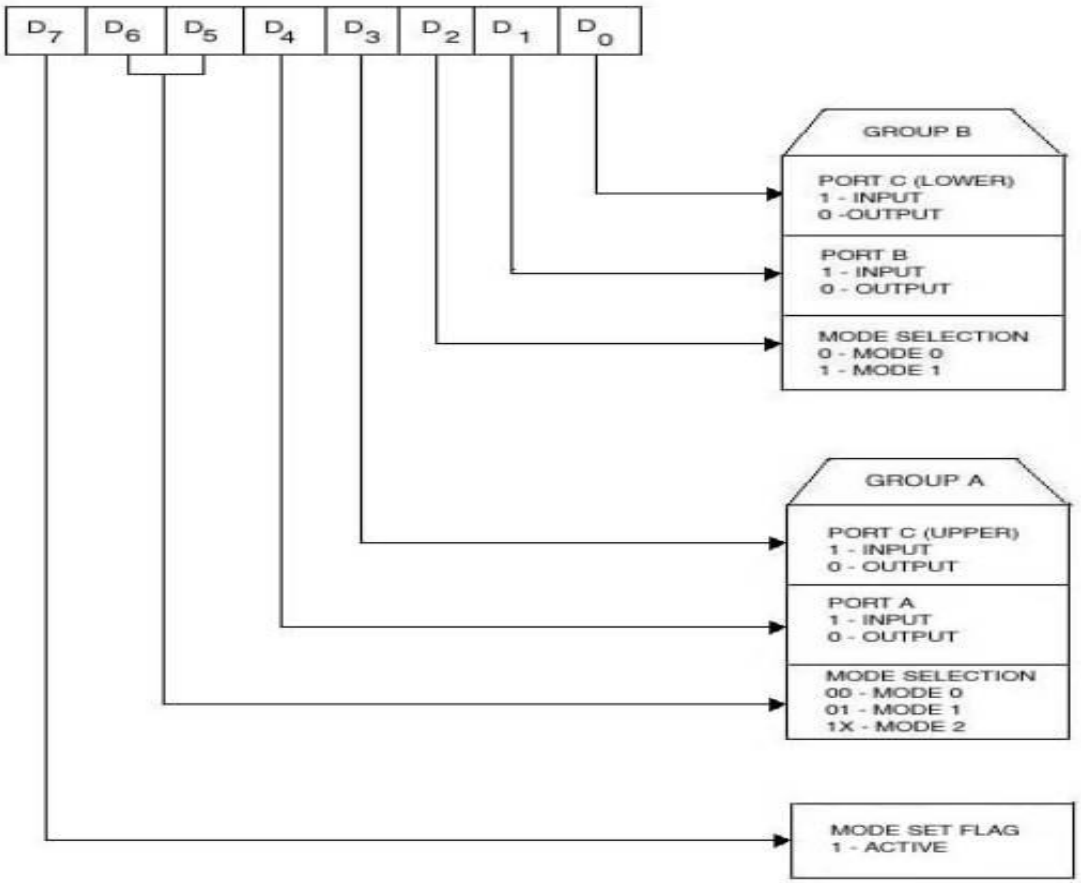
To interface programmable peripheral interface 8255 with 8085 and study its characteristics in mode0, mode1 and BSR mode.

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
4100		START:	MVI	A, 90	Initialize port A as Input and Port B as output.
4101					
4102			OUT	C6	Send Mode Control word
4103					
4104			IN	C0	Read from Port A
4105					
4106			OUT	C2	Display the data in port B
4107					
4108			STA	4200	Store the data read from Port A in 4200
4109					
410A					
410B			HLT		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
4100		START:	MVI	A, B4	Initialize port A as Input port in mode 1.
4101					
4102			OUT	C6	Send Mode Control word
4103					
4104			MVI	A,09	Set the PC4 bit for INTE A
4105					
4106			OUT	C6	Display the data in port B

4107					
			EI		
4108			MVI	A,08	Enable RST5.5
4109					
410A			SIM		
			EI		
410B			HLT		Stop the program.

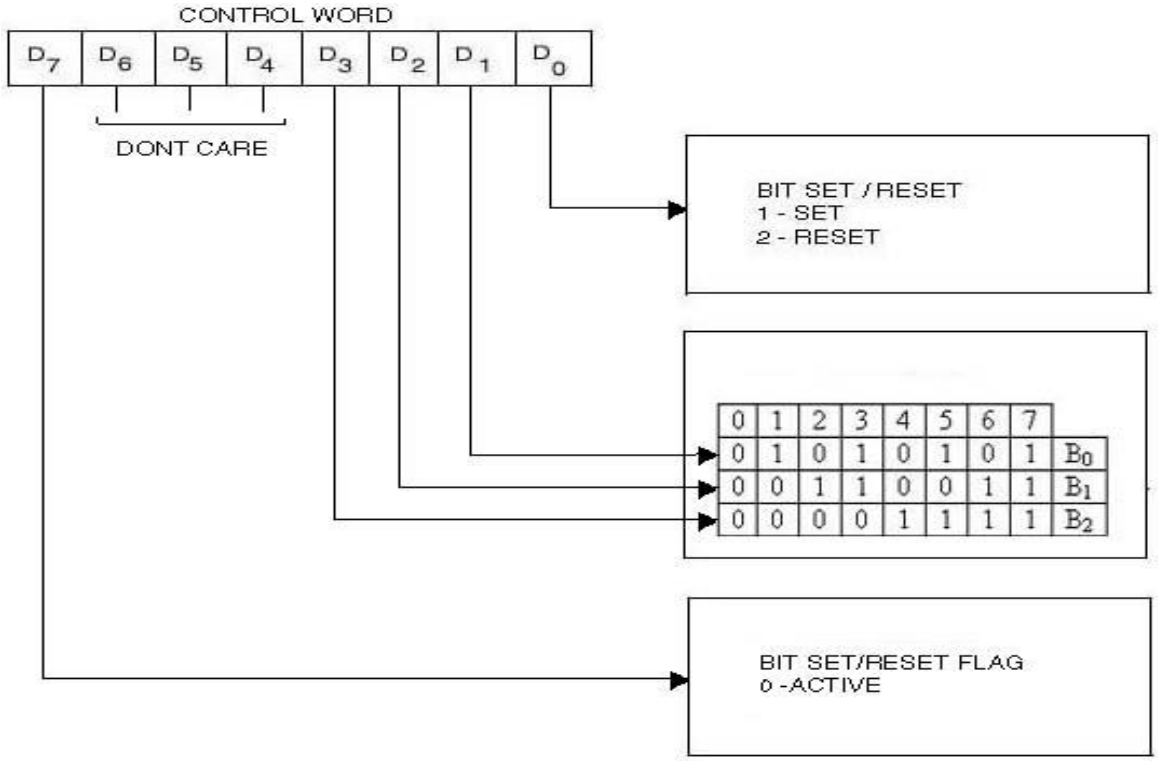
ISR (Interrupt Service Routine)

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
4200		START:	IN	C0	Read from port A
4201					
4202			STA	4500	Store in 4500.
4203					
4204					
4205			HLT		Stop the program.

BSR MODE(Bit set reset mode)

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
4100		START	MVI A,01		Go to 4200
4102			OUT C6		
4104			MVI A,07		
4106			OUT C6		
4108			HLT		

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
4100		START:	MVI	A, 01	Set PC0
4101					
4102			OUT	C6	Send Mode Control word
4103					
4104			MVI	A,07	Set PC3
4105					
4106			OUT	C6	Send Mode Control word
4107					
4109			HLT		Stop the program.

RESULT:

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

(B) INTERFACING SERIAL COMMUNICATION (8251) WITH 8085

Communication between two 8085 Microprocessors

Aim:

To transmit and receive a Character between two 8085 μps using 8251A

Apparatus Required:

8085 μp Kit – 2 No.s , RS 232C cable , Power supply – 2 No.s

Theory:

The program first initializes the 8253 to give an output clock frequency of 150KHz at channel 0 which will give a 9600 baud rate of 8251A. Then the 8251A is initialized to a dummy mode command. The internal reset to 8251A is then provided, since the 8251A is in the command mode now. Then 8251A is initialized as follows.

Initializing 8251A using the Mode instruction to the following.

8 bit data
No parity
16x Baud rate factor
1 stop bit
B2 , B1 = 1 , 0
L2 , L1 = 1 ,1
PEN = 0
EP = 0
S2 , S1 = 0 , 1

gives a Mode command word of 4E.

When 8251A is initialized as follows using the command instruction,

Reset Error flags,
Enable transmission and reception,
Make RTS and DTR active low.

EH = 0 SBRK = 0
IR = 0 RxE = 1
RTS = 1 DTR = 1
ER = 1 TxEN = 1

We get a command word of 37

The program after initializing , will read the status register and check for TxEMPTY. If the transmitter buffer is empty then it will send 41 to the serial port and then check for a character in the receive buffer. If some character is present then, it is received and stored at location 4200H.

Program:

ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
4100		MVI A,36			
4102		OUT TMRCNT			
4104		MVI A,0A			
4106		OUT MRCHO			
4108		XRA A			
4109		OUT TMRCHO			
410B		XRA A			
410C		OU UATCNT			
410E		MVI A,40			
4110		OUT UATCNT			
4112		MVI A,4E			
4114		OUT UATCNT			
4116		MVI A,37			

4118		OUT UATCNT			
ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
411A	LOOP	IN UATCNT			
411C		ANI 04			
411E		JNZ LOOP			
4121		MVI A,41			
4123		OUT UATDAT			

Program for Receiver:

ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
411A	LOOP	IN UATCNT			
411C		ANI 02			
411E		JZ LOOP1			
4121		IN UATDAT			
4123		STA 4200			
4126		HLT			

Procedure:

Feed the above program in two 8085 μps (One acts as Transmitter and the other acts as Receiver). Execute the two programs simultaneously. Check the Receiver at location 4200H. It‘s content will be 41.

Exercise: Write a program to transmit a block of data from transmitter and receive them at the receiver.

INPUT		OUPUT	
Address	Data	Address	Data
A reg		4200	

Result:

Thus the communication between two microprocessors has been established.

7. READ A KEY ,INTERFACE DISPLAY

INTERFACING 8279 WITH 8085

AIM:

To write the program to show the LED segment in 8279 by interfacing 8085 with 8279.

APPARATUS REQUIRED:

(i) 8085 microprocessor

(ii) Power supply

(iii) Keyboard

(iv) 8279 interfacing card.

ALGORITHM:

1. Start the program.
2. Get the Hex code at the memory location 809, 800B,8011,8015
3. State results interfacing card 0123.
4. Stop the execution

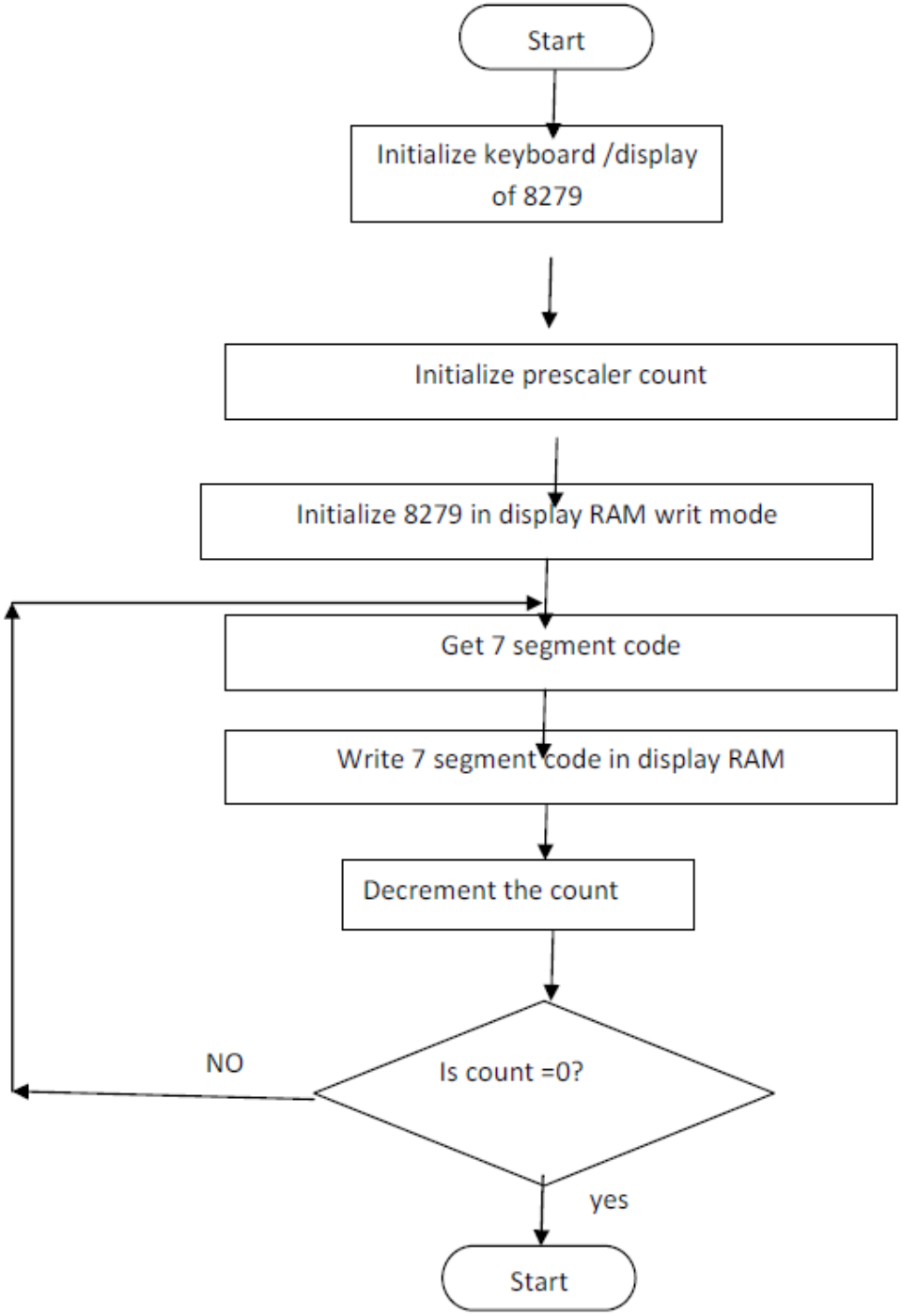
PROGRAM:

ADDRESS	LABEL	MNEMONICS	OPCODE	OPERAND	COMMENT
4100	START	LXI H		412C	
4103		MVI D		0F	
4105		MVI A		10	
4107		OUT		C2	
4109		MVI A		CC	
410B		OUT		C2	
410D		MVI A		90	
410F		OUT		C2	
4111		MOV A,M			
4112		OUT		C0	
4114		CALL DELAY		DELAY	
4117		INX H			
4118		DCR D			
4119		JNZ LOOP		LOOP	
411C		JMP START		START	
411F	DELAY	MVI B		A0	
4121	LOOP1	MVI C		FF	
4123	LOOP2	DCR C			
4124		JNZ LOOP1		LOOP1	
4127		DCR B			
4128		JNZ LOOP2		LOOP2	
412B		RET			

OBSERVATION:

LETTER	SEGMENT	DATA BUS								HEXADECIMAL
		D7	D6	D5	D4	D3	D2	D1	D0	

FLOW CHART:



RESULT:
Thus 8279 Controller was interfaced with 8085 and Program for Read Key and Rolling Display was Executed Successfully.

8051 MICROCONTROLLER PROGRAMS

8. DEMONSTRATION OF BASIC INSTRUCTIONS WITH 8051 MICRO CONTROLLER EXECUTION, INCLUDING:

(i) Conditional Jumps, Looping

&

9. PROGRAMMING I/O PORT 8051

(i). Interfacing DAC With 8051

AIM:

To interface DAC with 8051 parallel port to demonstrate the generation of square, Saw tooth and triangular wave.

APPARATUS REQUIRED:

- 8051Trainer Kit
- DAC Interface Board

THEORY:

DAC 0800 is an 8–bit DAC and the output voltage variation is between –5V and 5V.The output voltage varies in steps of 10/256=0.04(appx.).The digital data input and The corresponding output voltages are presented in the Table below.

Input Data in HEX	Output Voltage
00	5.00
01	4.96
02	....
…	…..
7F	…
FD	….
FE	4.96
FF	5.00

Referring to Table1,with 00 H as input to DAC, the analog output is –5 V. Similarly, With FF H as input, the output is +5V. Outputting digital data 00 and FF at regular intervals, to DAC , results in different waveforms namely square, triangular ,etc,.

ALGORITHM:

(a) **Square Wave Generation**

1. Move the port address of DAC to DPTR
2. Load the initial value(00) to Accumulator and move it to DAC
3. Call the delay program
4. Load the final value(FF) to accumulator and move it to DAC
5. Call the delay program.
6. Repeat Steps 2 to 5

(b) Saw tooth Wave Generation

1. Move the port address of DAC to DPTR
2. Load the initial value (00) to Accumulator
3. Move the accumulator content to DAC
4. Increment the accumulator content by 1.
5. Repeat Steps 3 and 4.

(c) Triangular Wave Generation

1. Move the port address of DAC to DPTR
2. Load the initial value (00) to Accumulator
3 Move the accumulator content to DAC
4. Increment the accumulator content by 1.
5. If accumulator content is zero proceed to next step. Else go to step3.
6. Load value (FF) to Accumulator
7. Move the accumulator content to DAC
8. Decrement the accumulator content by 1.
9. If accumulator content is zero go to step 2. Else go to step 7.

PROGRAM:

a) Square Wave Generation

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
		ORG 4100		
4100		MOV DPTR,#FF,C8		
		MOV A,#00		
	START	MOVX @DPTR,A		
		LCALL DELAY		
		MOV A,#FF		
		MOVX @DPTR,A		
		LCALL DELAY		
		LJUMP START		
	DELAY	MOV R1,#05		
		MOV R2,#FF		
	LOOP:	DJNZ R2,HERE		
	HERE			
		DJNZ R1,LOOP		
		RET		
		SJMP START		

(b) Saw tooth Wave Generation

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS

		ORG 4100
		MOV DPTR, #FF, C0
		
		MOV A,#00		

	LOOP	MOVX @DPTR ,A INC A SJMP LOOP		

(c) Triangular Wave Generation

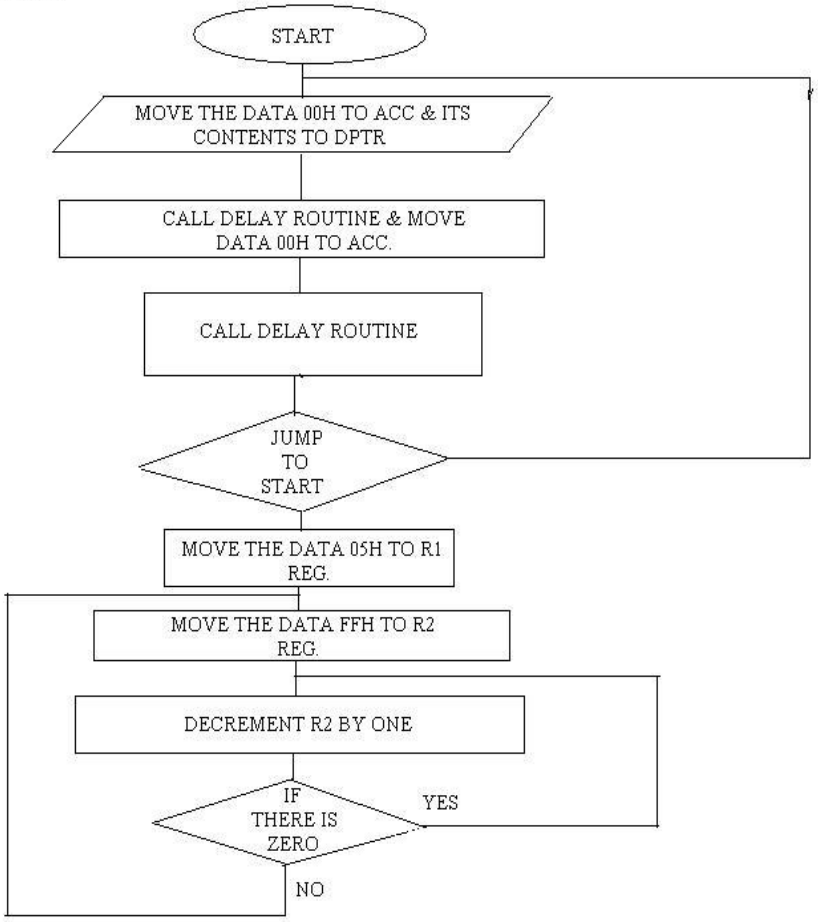
ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS

		ORG 4100
		MOV DPTR, #FF,C8
		
		MOV A,#00		

	START LOOP1	MOVX @DPTR ,A INC A JNZ LOOP1 MOV A,#FF		
	LOOP2:	MOVX @DPTR,A DEC A JNZ LOOP2 LJMP START		

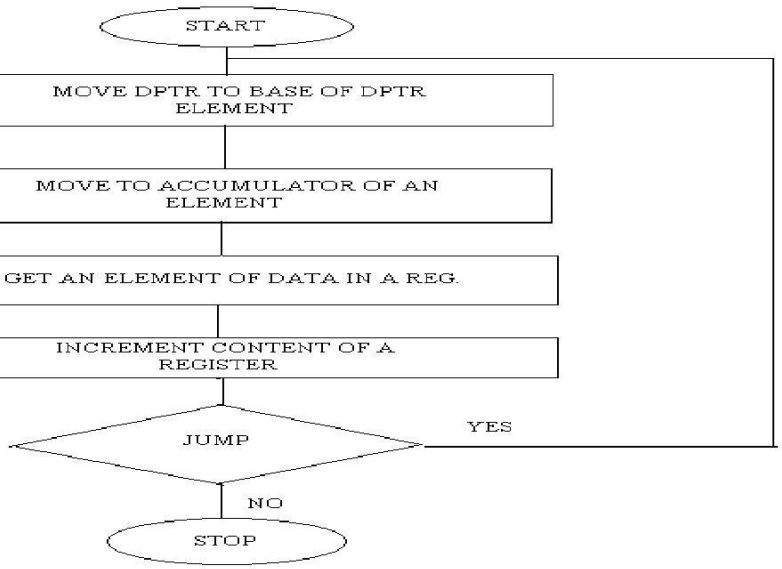
FLOWCHART:

SQUARE WAVE FORM :-



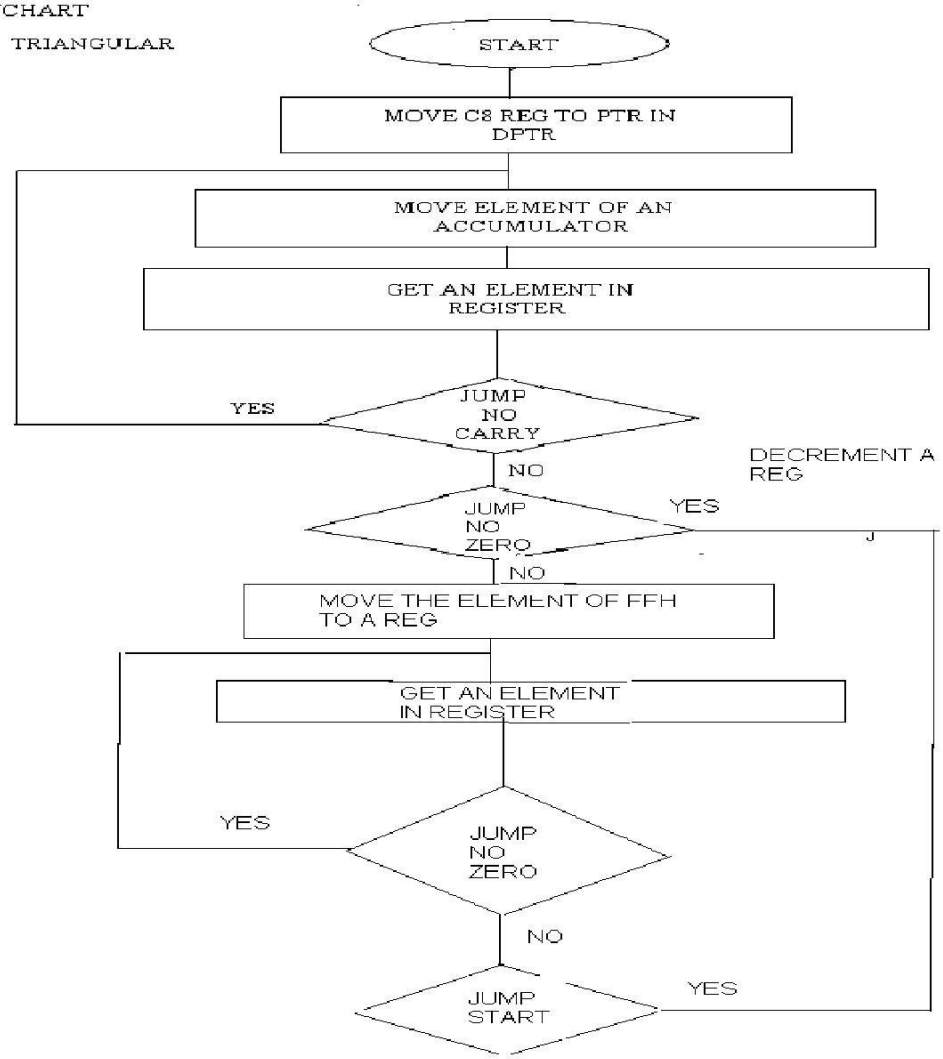
FLOW CHART

SAW TOOTH WAVEFORM :

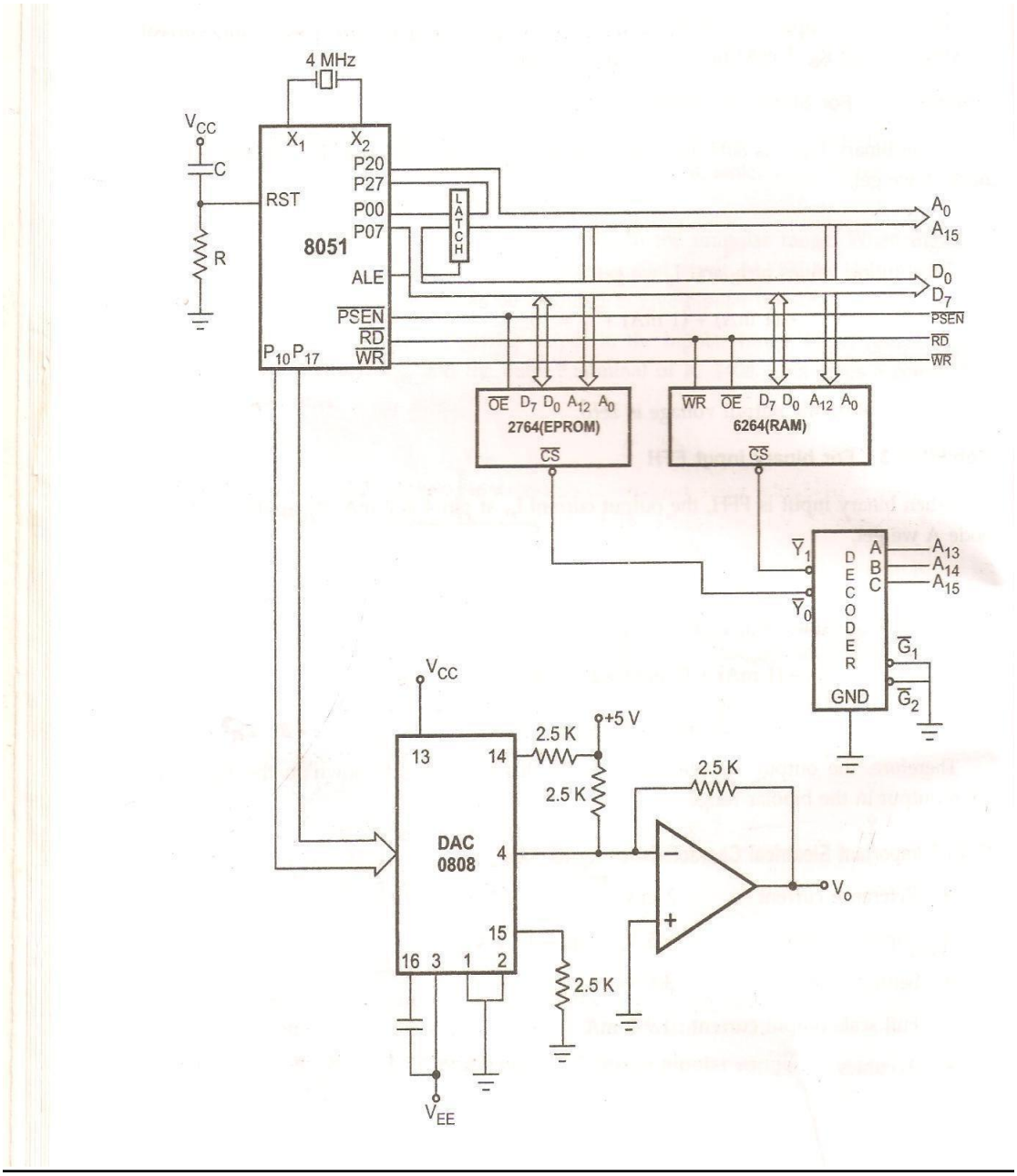


FLOWCHART

TRIANGULAR



DAC INTERFACING



OBSERVATION:

WAVE FORMS	AMPLITUDE	TIME PERIOD
SQUARE WAVEFORM		
SAW TOOTH WAVEFORM		
TRIANGULAR WAVEFORM		

RESULT:

Thus the square, triangular and saw tooth waveform were generated by interfacing DAC with 8051 trainer kit.

(i). Interfacing ADC With 8051

AIM:

To Write an Assembly Language Program to Convert an analog signal into digital using ADC interfacing in 8051 microcontroller kit.

APPARATUS REQUIRED:

- 8051Trainer Kit
- ADC Interface Board

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
4100			MOV DPTR,#FF C8	CHANNEL 0 SELECTION
4103			MOV A,#10	AND ALE LOW
4105			MOVX @DPTR,A	
4106			MOV A,#18	ALE HIGH
4108			MOVX @DPTR,A	
4109			MOV A,#10	
410B			MOV@DPTR,A	
410C		HERE	SJMP HERE	

PROCEDURE

1. Place jumper j2 in B position
2. Place jumper j5 in A position
3. Enter and execute the program
4. Vary the analog input and give the SOC by pressing the SOC switch.
5. See the corresponding digital value in the led display.

OBSERVATION:

ANALOG VOLTAGE	DIGITAL DATA ON LED DISPLAY	HEX CODE

RESULT:

Thus the conversion of analog signal into a digital value was done by using ADC interfacing with 8051 trainer kit.

(ii) STUDY ON INTERFACE WITH DC MOTOR

AIM:

To study and interface with DC Motor in 8051 microcontroller kit.

APPARATUS REQUIRED:

1. **DC Motor**
2. **8051 microcontroller kit**

Program:

To control speed of the motor.

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
		PORT	EQU 0C0H	
4100		START	MVI A,0FFH	Max. data to DAC
4102			OUT 0C0H	DAC port
4104				
4106			HLT	

PROCEDURE:

1. Run the DC motor at full speed by latching 'FF' to the DAC.
2. Make the Gate of Channel 0 To '0' Logic.
3. Initialize The Counter Value. Calculate The Difference In Count.
4. Make the gate logic1 for 1 second.
5. Read the counter value. Calculate the difference in count.
6. Display RPM

RESULT:

Thus the study and interface of the DC motor with 8051 microcontroller kit were studied and verified.