SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM NAGAR, KATTANKULATHUR-603203

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

LAB MANUAL

1905507 -MICROPROCESSORS AND MICROCONTROLLERS LABORATORY

V SEMESTER

(Academic Year – 2022-2023-ODD sem)

Prepared By

Mrs. S.Vanila, A.P. (Sel.G) / EEE

1905507 MICROPROCESSORS AND MICROCONTROLLERS LABORATORY **LTPC**
0 0 4 2

SYLLABUS

OBJECTIVES:

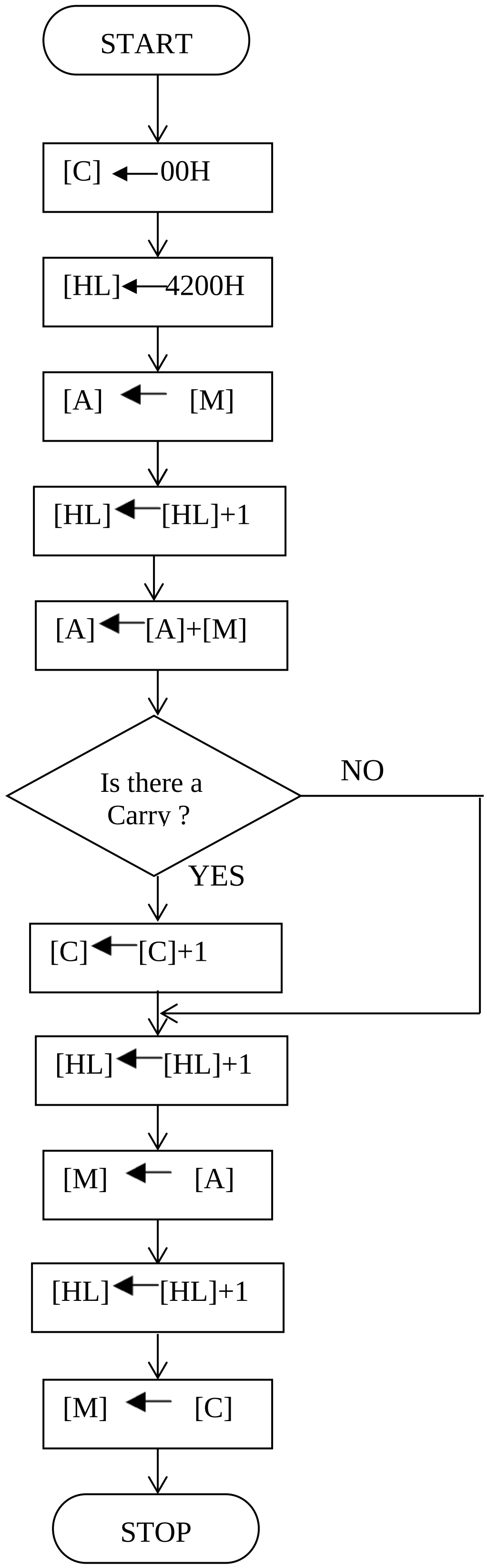
- *To perform simple arithmetic operations using assembly language program and study the addressing modes & instruction set of 8085 & 8051.*
- *To develop skills in simple program writing in assembly languages.*
- *To write an assembly language program to convert Analog input to Digital output and Digital input to Analog output.*
- *To perform interfacing experiments with μP8085 and μC8051.*
- *To study various digital integrated circuits used in simple system configuration.*

LIST OF EXPERIMENTS

	Programming exercises / Experiments with μP8085:
1.	Simple arithmetic operations: Multi precision addition / subtraction/multiplication / division.
2.	Programming with control instructions: Increment / Decrement, Ascending Descending order, Maximum / Minimum of numbers, Rotate instructions, Hex /ASCII / BCD code conversions.
3.	Interface Experiments: A/D Interfacing. D/A Interfacing. Traffic light controller.
4.	Stepper motor controller interface.
	Programming exercises / Experiments with μC8051:
5.	Simple arithmetic operations with 8051: Multi precision addition / subtraction /multiplication / division.
6.	Programming with control instructions: Increment / Decrement, Ascending /Descending order, Maximum / Minimum of numbers, Rotate instructions, Hex /ASCII / BCD code conversions.
7.	Interface Experiments: A/D Interfacing. D/A Interfacing. Traffic light controller
8.	Stepper motor controller interface.
	Experiments with Digital ICs:
9.	Study of Basic Digital IC's.(Verification of truth table for AND, OR, EXOR, NOT, NOR, NAND, JK FF, RS,FF, D FF)
10.	Implementation of Boolean Functions, Adder/ Subtractor circuits; Realizing given function with minimum number of gates by minimization methods.
11.	Study of binary / BCD counters, modulo-n counters.
12.	Design and implementation of Synchronous sequential counters.
13.	Programming ARM architecture with software tools.

CONTENTS

FLOW CHART:



START
[C] ← 00H
[HL] ← 4200H
[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

1(A) 8-BIT DATA ADDITION

AIM:

To add two 8 bit numbers stored at consecutive memory locations and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	MVI	C, 00	Clear C reg.
4101					
4102			LXI	H, 4200	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	
4200	40	4202	60
4201	20	4203	00

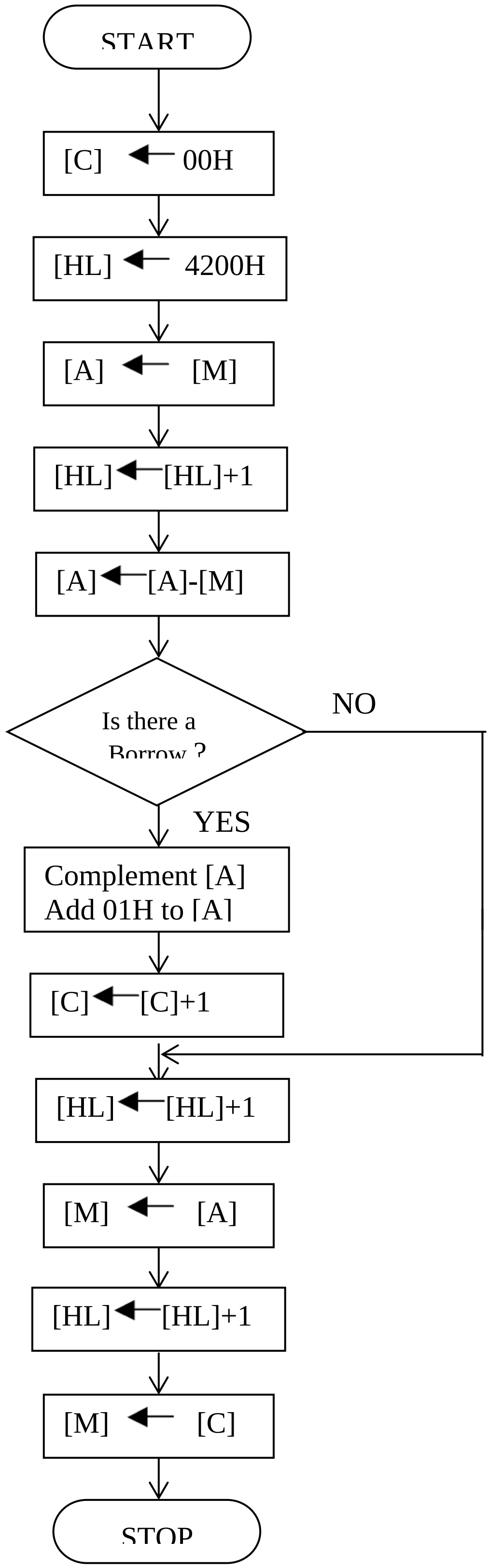
RESULT:

Thus the two 8 bit numbers stored at 4200 &4201 are added and the result is stored at 4202 & 4203.

VIVA QUESTIONS:

1. What is the function of LXI H, 4000 H instruction?
2. How you can store a data in a memory location?
3. What is the meaning of INX
3. How you can read a data from a memory location?
4. What are flags available in 8085 ?
5. What is the function of RESET key of a 8085 microprocessor kit
6. What is the function of JNC instruction?
7. What is the difference between conditional and unconditional jump instruction?
8. What is multi byte?

FLOW CHART:



START
[C] ← 00H
[HL] ← 4200H
[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

1(B) 8-BIT DATA SUBTRACTION

AIM:

To subtract two 8 bit numbers stored at consecutive memory locations and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit, key board

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	MVI	C, 00	Clear C reg.
4102					
4102			LXI	H, 4200	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	
4200	04	4202	03
4201	01	4203	00

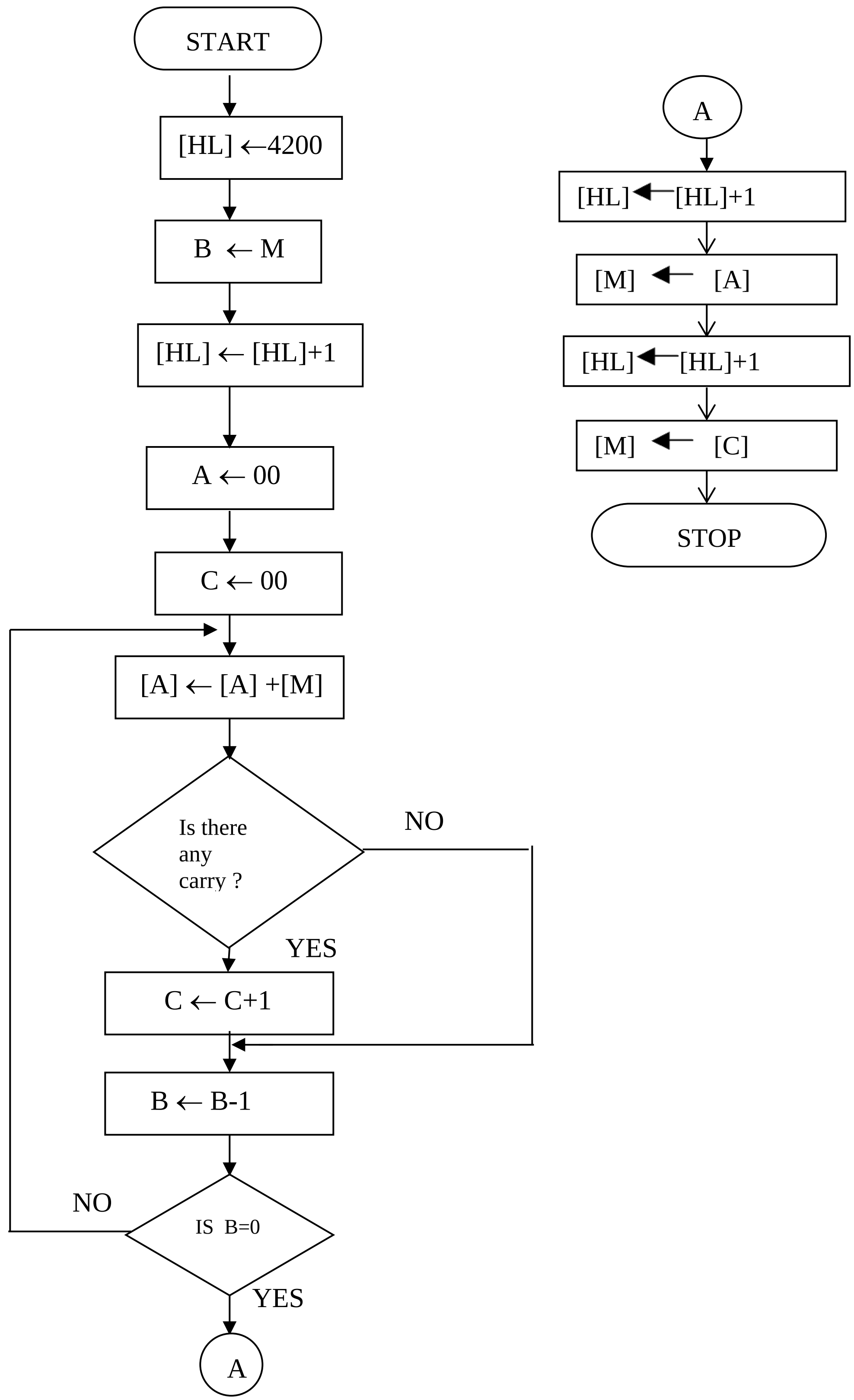
RESULT:

Thus the 8 bit numbers stored at 4200 &4201 are subtracted and the result is stored at 4202 & 4203.

VIVA QUESTIONS:

1. What is meant by ADI instruction
2. What is an instruction?
3. What is Mnemonic?
4. What is the purpose of CMA instruction?
5. What is the function of stack pointer?
6. Why ADI 01H is used in two's complement of an 8-bit number.
7. How many memory locations can be addressed by a microprocessor with 14 address lines?

FLOW CHART:



START
[HL] ←4200
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

1(C) 8-BIT DATA MULTIPLICATION

AIM:

To multiply two 8 bit numbers stored at consecutive memory locations and also to verify the result .

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

ALGORITHM:

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	LXI	H, 4200	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:

INPUT		OUTPUT	
4200	02	4202	08
4201	04	4203	00

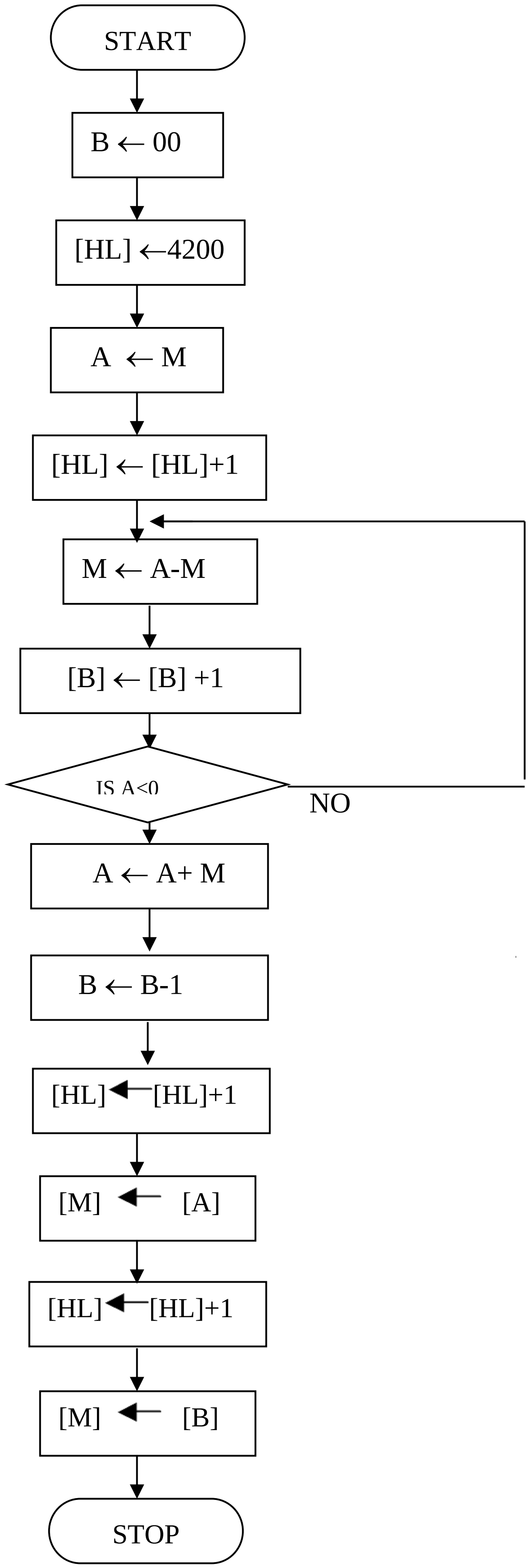
RESULT:

Thus the 8-bit multiplication was done in 8085µp using repeated addition method and also the result is verified.

VIVA QUESTION:

1. Define two's complement of an 8-bit numbers.
2. What is meant by instruction ADC M?
3. What is the use of the instruction MOV A,M
4. What is the function of program counter?
5. Mention the types of 8085 instruction set.
6. How will you perform multiplication using ADD instruction?
7. Describe about DAD B instruction.
8. What is the purpose of the instruction MOV M,A

FLOWCHART:


START
B ← 00
[HL] ←4200
A ← M
[HL] ← [HL]+1
M ← A-M
[B] ← [B] +1
IS A<0
NO
A ← A+ M
B ← B-1
[HL] ← [HL]+1
[M] ← [A]
[HL] ← [HL]+1
[M] ← [B]
STOP

1(D) 8-BIT DIVISION

AIM:

To divide two 8-bit numbers stored in memory and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

ALGORITHM:

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			MVI	B,00	Clear B reg for quotient
4101					
4102			LXI	H,4200	Initialize HL reg. to 4200H
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:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	06	4202	00
4201	02	4203	03

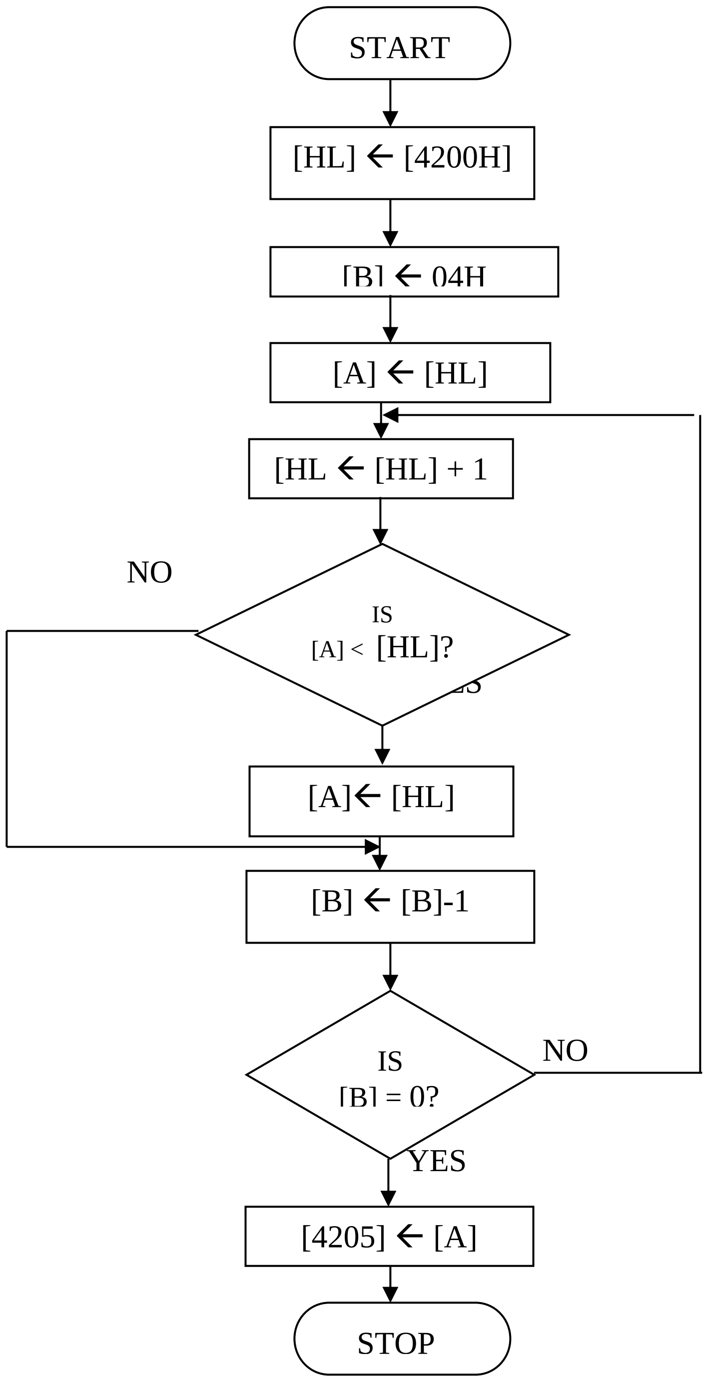
RESULT:

Thus an ALP was written for 8-bit division and also the result is also verified.

VIVA QUESTIONS:

1. What SUB M instruction will do?
2. Describe SBB M instruction
3. Express the use of SUI with an example
4. Where SBI can be used?
5. Give the purpose of the instruction LDAX D
6. How will you perform Division using ADD instruction ?
7. What is the need of ALE signal in 8085?
8. What are the addressing modes of 8085?
9. List the interrupt signals of 8085?

FLOW CHART:



START
[HL] ← [4200H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
NO
IS
[A] < [HL]?
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[4205] ← [A]
STOP

2(A) LARGEST ELEMENT IN AN ARRAY

AIM:

To find the largest element in an array of data stored in memory and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LXI	H,4200	Initialize HL reg. to 8100H
4101					
4102					
4103			MVI	B,04	Initialize B reg with no. of comparisons(n-1)
4104					
4105			MOV	A,M	Transfer first data to acc.
4106		LOOP1	INX	H	Increment HL reg. to point next memory location
4107			CMP	M	Compare M & A
4108			JNC	LOOP	If A is greater than M then go to loop
4109					
410A					
410B			MOV	A,M	Transfer data from M to A reg
410C		LOOP	DCR	B	Decrement B reg
410D			JNZ	LOOP1	If B is not Zero go to loop1
410E					
410F					
4110			STA	4205	Store the result in a memory location.
4111					
4112					
4113			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	01	4205	07
4201	06		
4202	03		
4203	07		
4204	02		

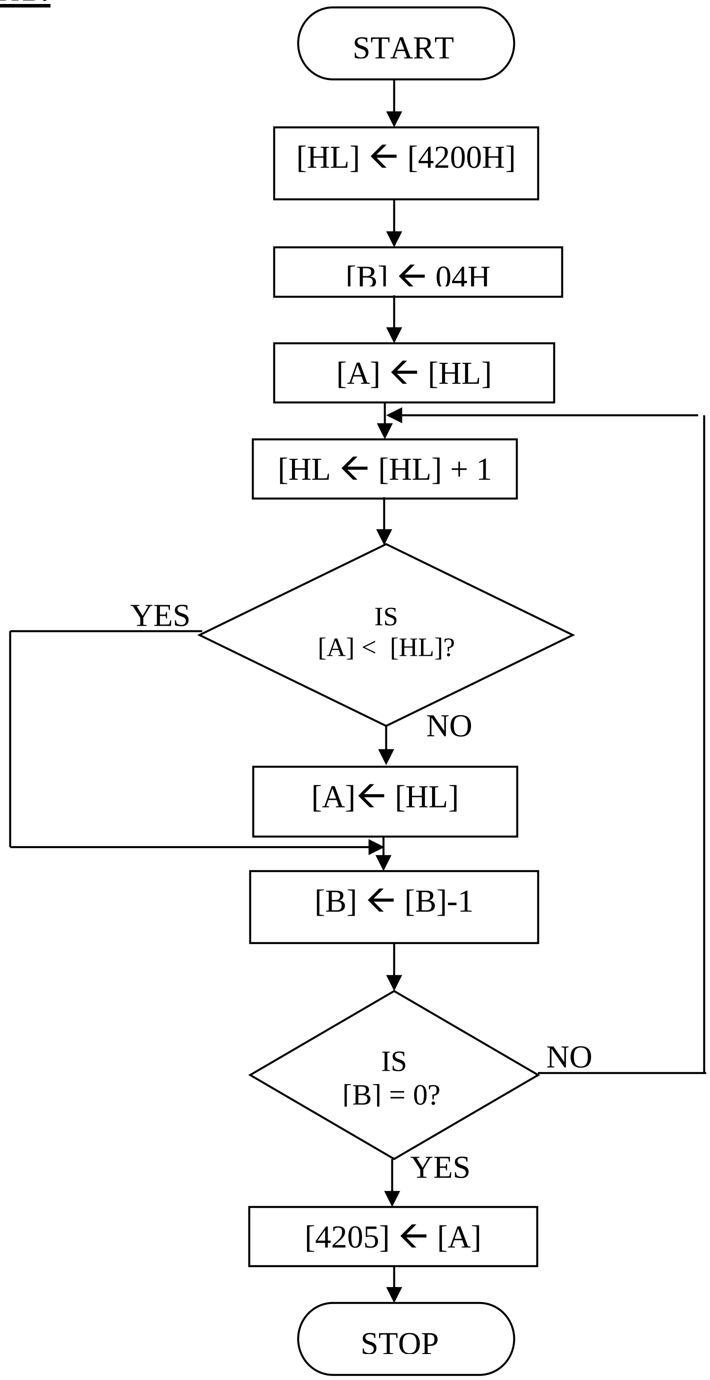
RESULT:

Thus the largest number in the given array is found and it is stored at location 4205.

VIVA QUESTIONS:

1. What is meant by the instruction CMP M
2. What the instruction JNZ will do
3. State the logic behind the finding of largest element
4. List out the similarities b/w the CALL-RET and PUSH-POP instructions?
5. What is the need of ALE signal in 8085?
6. What are the addressing modes of 8085?

FLOW CHART:


START
[HL] ← [4200H]
[B] ← 04H
[A] ← [HL]
[HL ← [HL] + 1
IS
[A] < [HL]?
YES
NO
[A]← [HL]
[B] ← [B]-1
IS
[B] = 0?
NO
YES
[4205] ← [A]
STOP

2(B) SMALLEST ELEMENT IN AN ARRAY

AIM:

To find the smallest element in an array of data stored in memory and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LXI	H,4200	Initialize HL reg. to 8100H
4101					
4102					
4103			MVI	B,04	Initialize B reg with no. of comparisons(n-1)
4104					
4105			MOV	A,M	Transfer first data to acc.
4106		LOOP1	INX	H	Increment HL reg. to point next memory location
4107			CMP	M	Compare M & A
4108			JC	LOOP	If A is lesser than M then go to loop
4109					
410A					
410B			MOV	A,M	Transfer data from M to A reg
410C		LOOP	DCR	B	Decrement B reg
410D			JNZ	LOOP1	If B is not Zero go to loop1
410E					
410F					
4110			STA	4205	Store the result in a memory location.
4111					
4112					
4113			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	01	4205	01
4201	06		
4202	03		
4203	07		
4204	02		

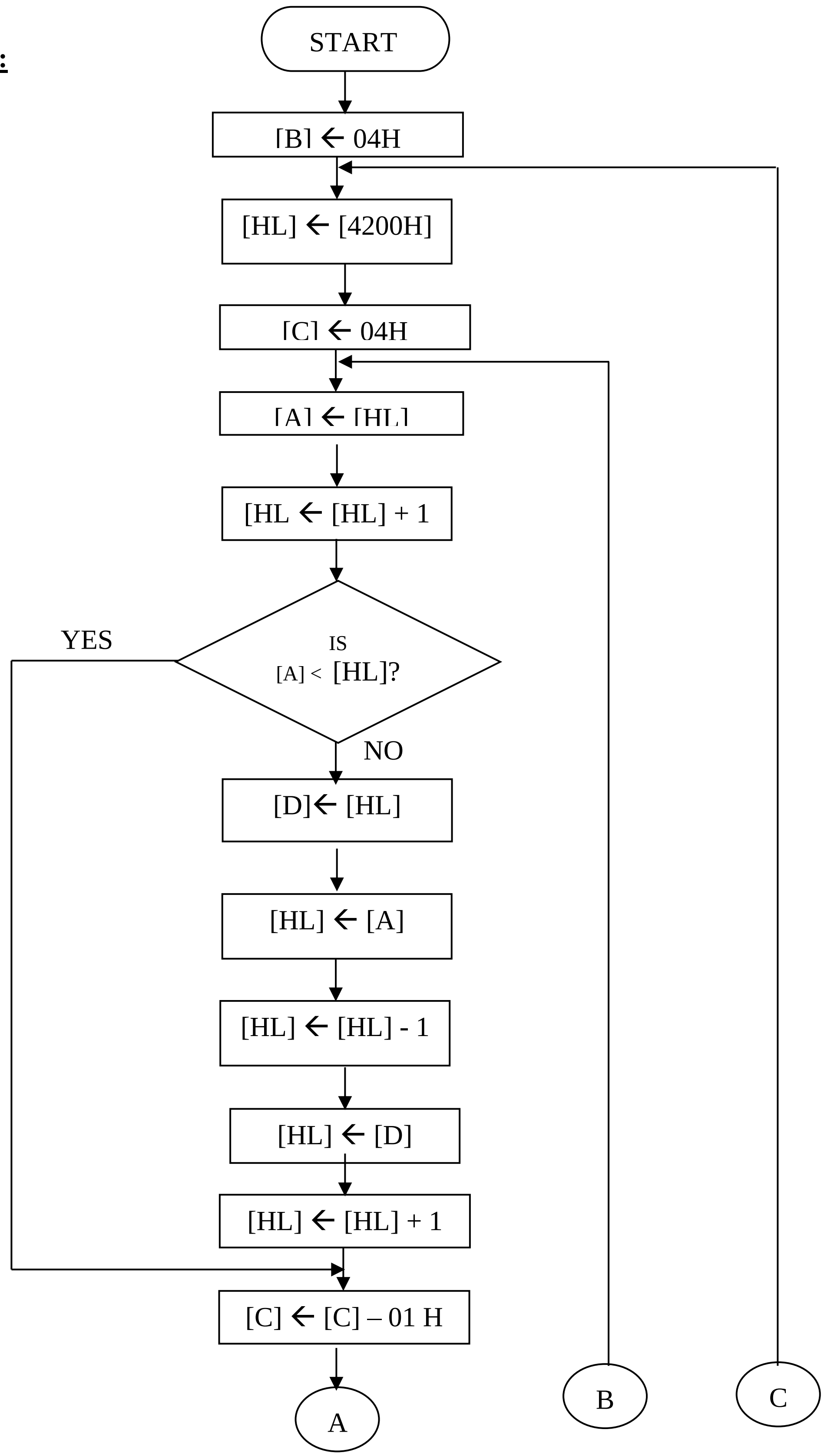
RESULT:

Thus the smallest number in the given array is found and it is stored at location 4205.

VIVA QUESTION:

1. What is meant by instruction JC ?
2. Tell about the instruction SHLD .
3. Summarize the instruction STAX B.
4. State the logic behind the finding of smallest element .
5. Why address bus is unidirectional?
6. List few instructions to clear accumulator?
7. What is the function of NOP instruction?

FLOWCHART:


START
[B] ← 04H
[HL] ← [4200H]
[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
B
C

2(C) ASCENDING ORDER

AIM:

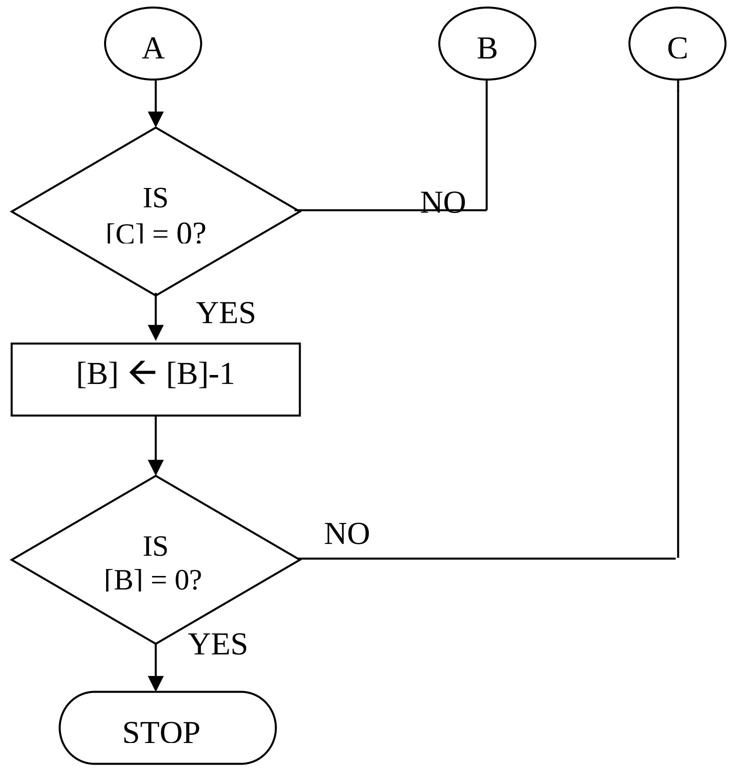
To sort the given numbers in the ascending order using 8085 microprocessor.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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


A
B
C
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	01	4200	01
4201	06	4201	02
4202	03	4202	03
4203	07	4203	06
4204	02	4204	07

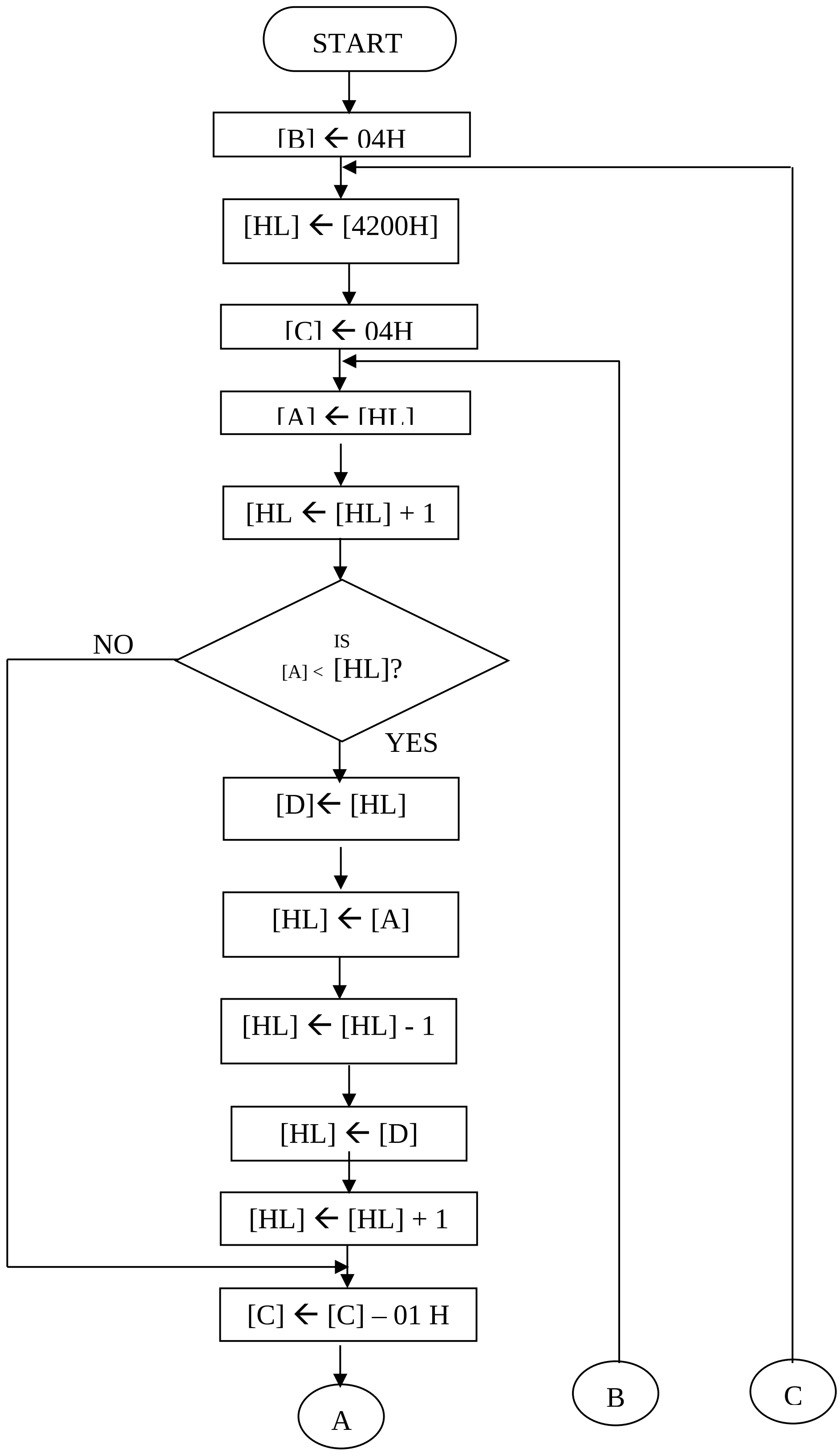
RESULT:

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

VIVA QUESTION:

1. Explain INX operation
2. State the logic behind the Sorting an array of data in Descending order
3. What are the advantages of using memory segmentation 8085?
4. What is the macro & when it is used?
5. What is the function of direction flag?
6. What is DMA?
7. Define machine cycle and instruction cycle?

FLOWCHART:


START
[B] ← 04H
[HL] ← [4200H]
[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
B
C

2(D) DESCENDING ORDER

AIM:

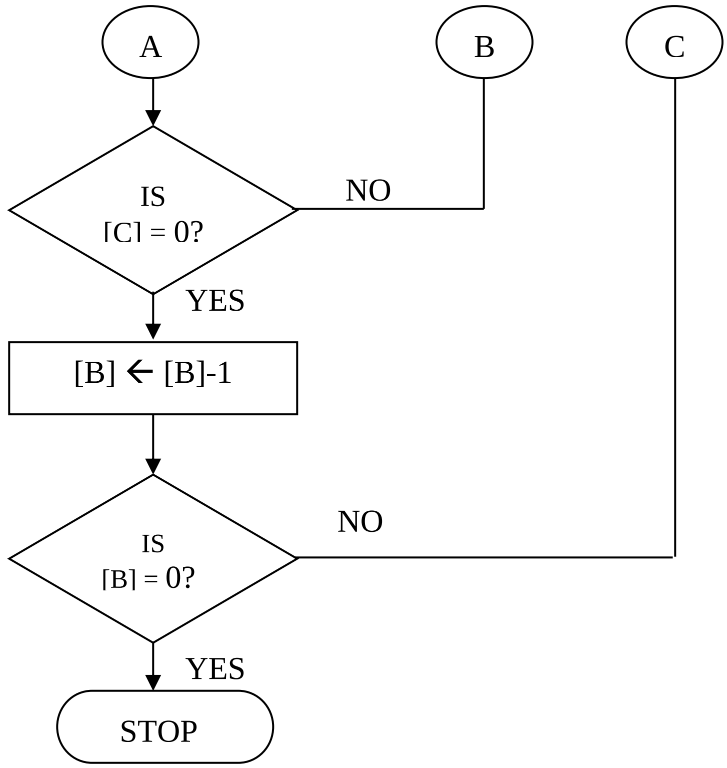
To sort the given numbers in the descending order using 8085 microprocessor.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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


A
B
C
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	01	4200	07
4201	06	4201	06
4202	03	4202	03
4203	07	4203	02
4204	02	4204	01

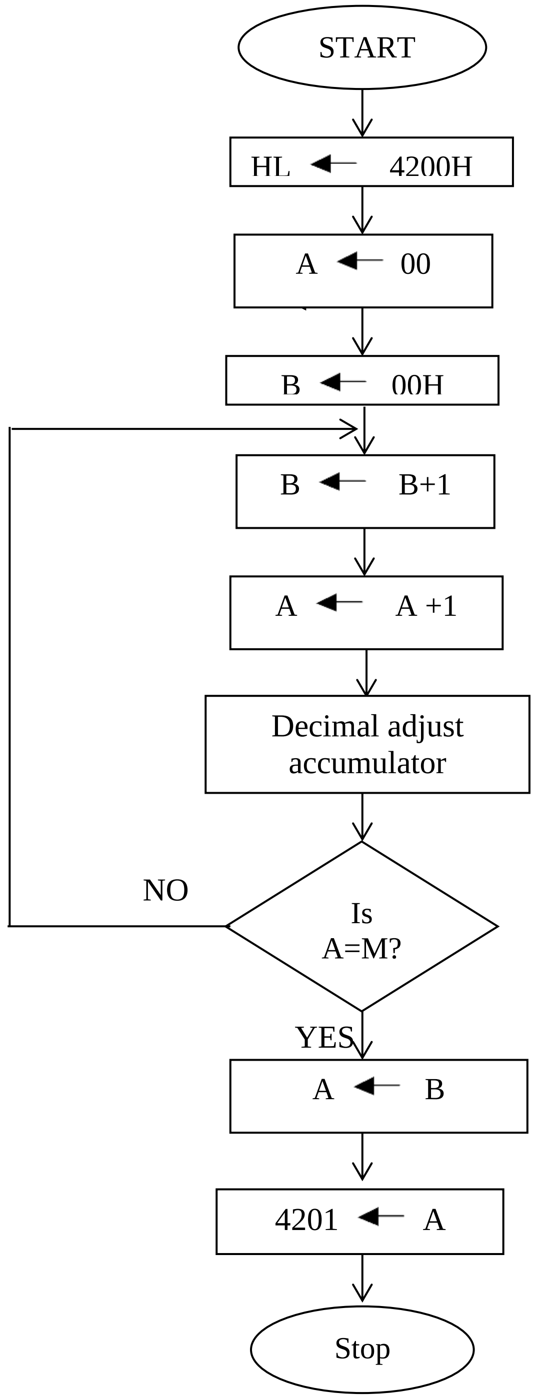
RESULT:

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

VIVA QUESTION:

1. Give out the purpose of the instruction DCX
2. What is meant by CALL instruction?
3. Briefly give out the LHLD instruction
4. State the logic behind the Sorting an array of data in Descending order
5. Name the various flag bits available in 8085 microprocessor?
6. Give the significance of SIM and RIM instructions available in 8085?
7. How do the address and data lines are demultiplexed in 8085?

FLOWCHART:


START
HL ← 4200H
A ← 00
B ← 00H
B ← B+1
A ← A +1
Decimal adjust accumulator
NO
Is A=M?
YES
A ← B
4201 ← A
Stop

2(E) CODE CONVERSION - DECIMAL TO HEXADECIMAL

AIM:

To convert a given decimal number to hexadecimal number.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LXI	H,4200	Initialize HL reg. to 4200H
4101					
4102					
4103			MVI	A,00	Initialize A register.
4104					
4105			MVI	B,00	Initialize B register..
4106					
4107		LOOP	INR	B	Increment B reg.
4108			ADI	01	Increment A reg
4109					
410A			DAA		Decimal Adjust Accumulator
410B			CMP	M	Compare M & A
410C			JNZ	LOOP	If acc and given number are not equal, then go to LOOP
410D					
410E					
410F			MOV	A,B	Transfer B reg to acc.
4110			STA	4201	Store the result in a memory location.
4111					
4112					
4113			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	21	4201	15

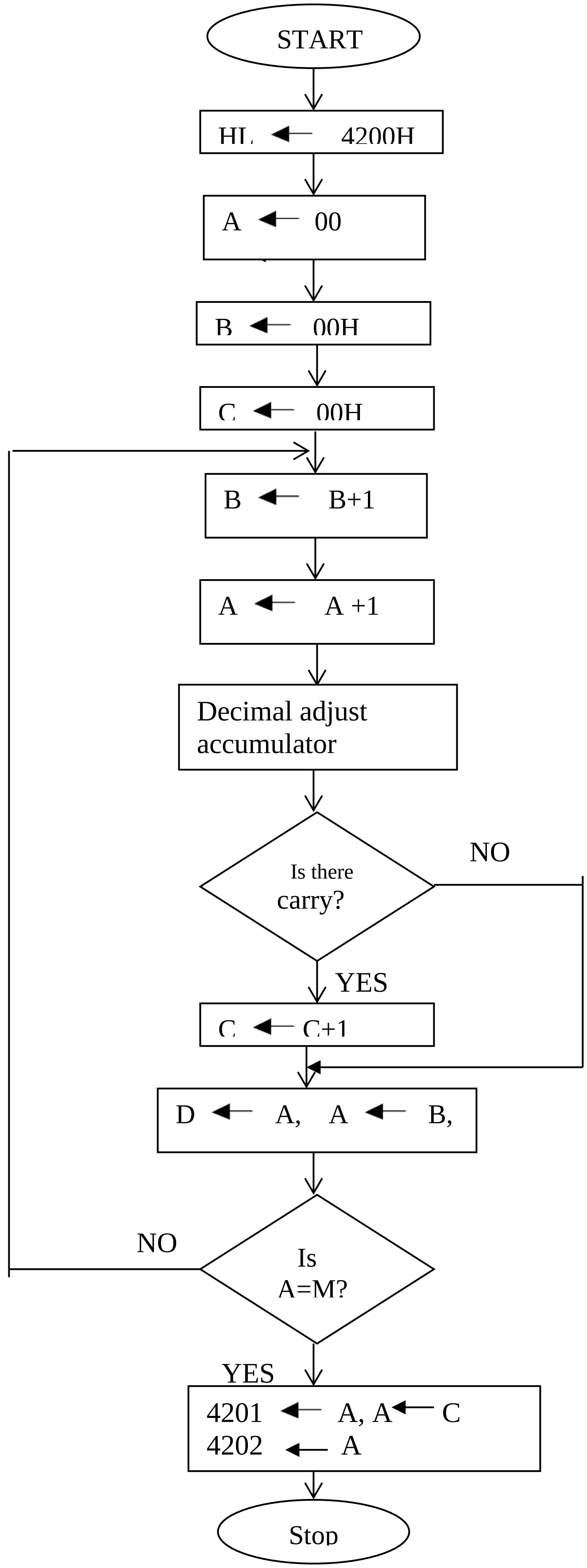
RESULT:

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

VIVA QUESTION:

1. What is meant by ADI instruction?
2. What is the function of DAA instruction?
3. What is the function of XCHG instruction?
4. How you can load 16-bit data in 8500H and 8501H memory locations?
5. What is the difference between LHLD and SHLD instructions?
6. What is physical address?
7. Define OFFSET address.

FLOWCHART:


START
HL ← 4200H
A ← 00
B ← 00H
C ← 00H
B ← B+1
A ← A +1
Decimal adjust accumulator
Is there carry?
NO
YES
C ← C+1
D ← A, A ← B,
Is A=M?
NO
YES
4201 ← A, A← C
4202 ← A
Stop

2(F) CODE CONVERSION - HEXADECIMAL TO DECIMAL

AIM:

To convert a given hexadecimal number to decimal number and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board.

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.

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LXI	H,4200	Initialize HL reg. to 8100H
4103			MVI	A,00	Initialize A register.
4105			MVI	B,00	Initialize B register.
4106					
4107			MVI	C,00	Initialize C register for carry.
4108					
4109		LOOP	INR	B	Increment B reg.
410A			ADI	01	Increment A reg
410B					
410C			DAA		Decimal Adjust Accumulator
410D			JNC	NEXT	If there is no carry go to NEXT.
4110			INR	C	Increment c register.
4111		NEXT	MOV	D,A	Transfer A to D
4112			MOV	A,B	Transfer B to A
4113			CMP	M	Compare M & A
4114			MOV	A,D	Transfer D to A
4115			JNZ	LOOP	If acc and given number are not equal, then go to LOOP
4118			STA	4201	Store the result in a memory location.
411B			MOV	A,C	Transfer C to A
411C			STA	4202	Store the carry in another memory location.
411F			HLT		Stop the program

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	D5	4201	13
		4202	02

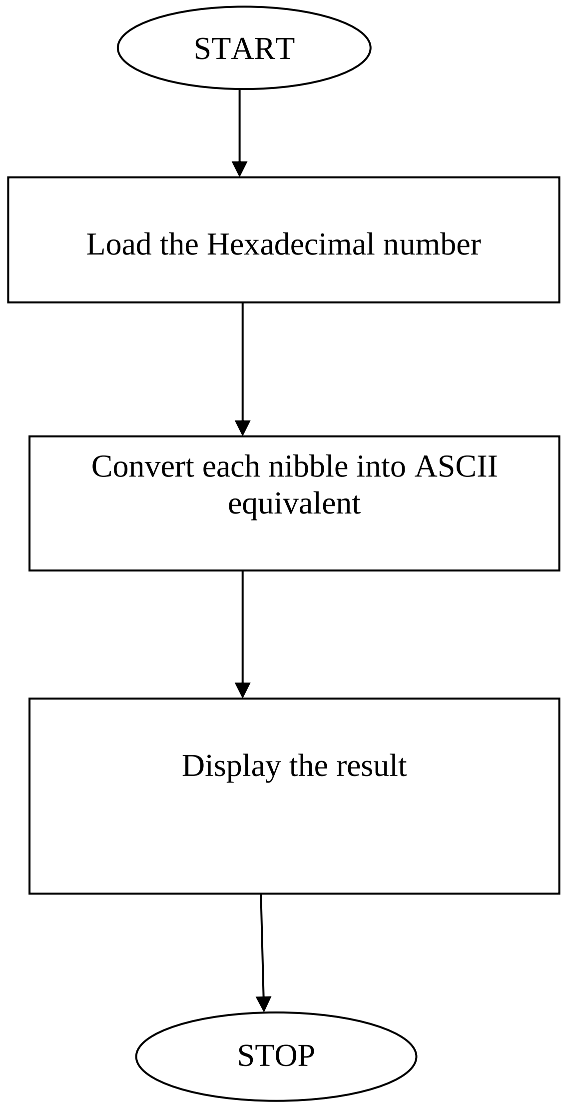
RESULT:

Thus an ALP program for conversion of hexadecimal to decimal was executed and the result is verified.

VIVA QUESTIONS:

1. What is meant by instruction DAA ?
2. Why data bus is bi-directional?
3. Specifies the function of address bus and the direction of address bus?
4. How many memory location can be addressed by a microprocessor with the 14 address lines?
5. List various instructions that can be used to clear accumulator in 8085?
6. When the Ready signal of 8085 is sampled by the processor?
7. List out the similarities b/w the CALL_RET and PUSH_POP instructions?

FLOWCHART:


START
Load the Hexadecimal number
Convert each nibble into ASCII equivalent
Display the result
STOP

2(G). CODE CONVERSION –HEXADECIMAL TO ASCII

Aim

To write an assembly language program to covert the given Hexadecimal number into its ASCII equivalent and to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board.

Algorithm:

Step 1: Load the Hexadecimal number from the location

Step 2: Separate the nibbles

Step 3: Convert each nibble to its ASCII Equivalent.

Step 4: Add the two converted values

Step 5: Display the result

Step 6: Stop

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LDA	4200	Get the data
4101					
4102					
4103			MOV	B,A	
4104			ANI	OF	Mask upper nibble
4105					
4106			CALL	SUB	Get ASCII code for upper nibble
4107					
4108					
4109			STA	4201	Store the value of accumulator
410A					
410B					
410C			MOV	A,B	Mov B reg content to Acc
410D			ANI	F0	Mask lower nibble
410E					
410F			RLC		Rotate left with out carry 4 times
4110			RLC		
4111			RLC		
4112			RLC		
4113			CALL	SUB	Get the ASCII code
4114					
4115					
4116			STA	4202	Store the accumulator
4117					
4118					
4119			HLT		Stop
411A		SUB	CPI	0A	Compare with OA
411B					
411C			JC	SKP	Skip if carry
411D					
411E					
411F			ADI	07	Add 07 to Acc
4120					
4121		SKP	ADI	30	Add 30 to Acc
4122					
4123			RET		Return

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	A5	4201	35
		4202	41

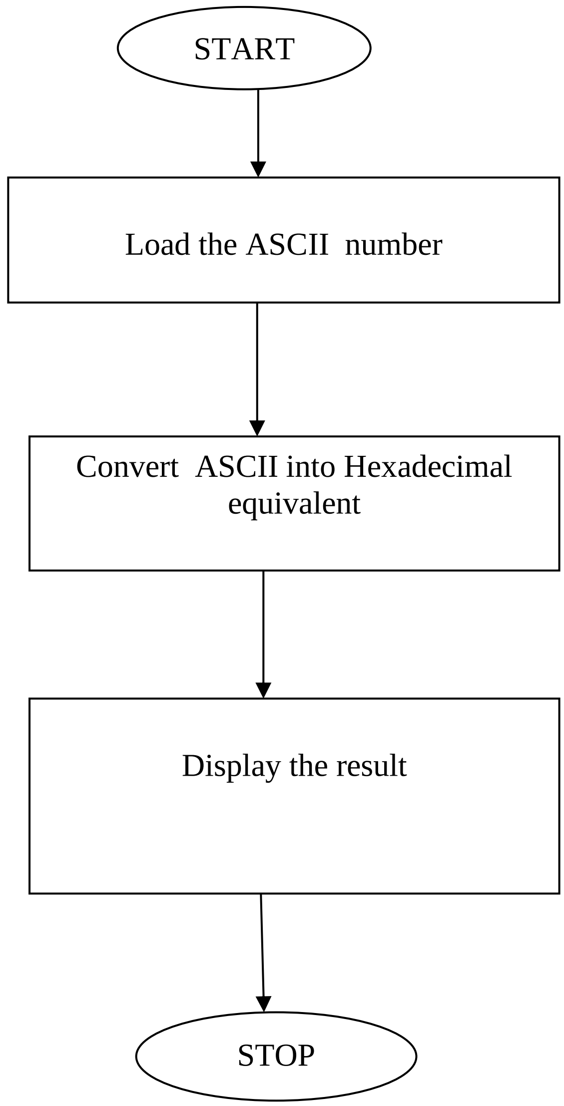
Result :

Thus assembly language program to covert the given Hexadecimal number into its ASCII equivalent is completed and also the result is verified.

VIVA QUESTIONS:

1. What is ASCII number for OAH?
2. What is difference between byte and word?
3. What is the immediate addressing mode?
4. What are data transfer instructions?
5. What is the use of immediate addressing mode?
6. What are branching instructions?
7. What is DMA ?

FLOW CHART:


START
Load the ASCII number
Convert ASCII into Hexadecimal equivalent
Display the result
STOP

2(H). CODE CONVERSION –ASCII TO HEXADECIMAL

AIM :

To write an assembly language program to covert the given ASCII number into its Hexadecimal equivalent and to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board.

ALGORITHM:

Step 1: Load the ASCII number from the location

Step 2: Check for the digit or alphabet

Step 3: Using suitable logic and instructions convert the ASCIII number into Hexadecimal

Step 4: Add the two converted values

Step 5: Display the result

Step 6: Stop

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LDA	4500	Load the memory content to Accumulator
4102					
4102					
4103			SUI	30	Subtract with 30
4104					
4105			CPI	0A	Compare with OA
4106					
4107			JC	SKP	If carry skip
4108					
4109					
410A			SUI	07	Subtract with 07
410B					
410C		SKP	STA	4201	Store Accumulator content
410D					
410E					
410F			HLT		Stop

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	41	4201	0A

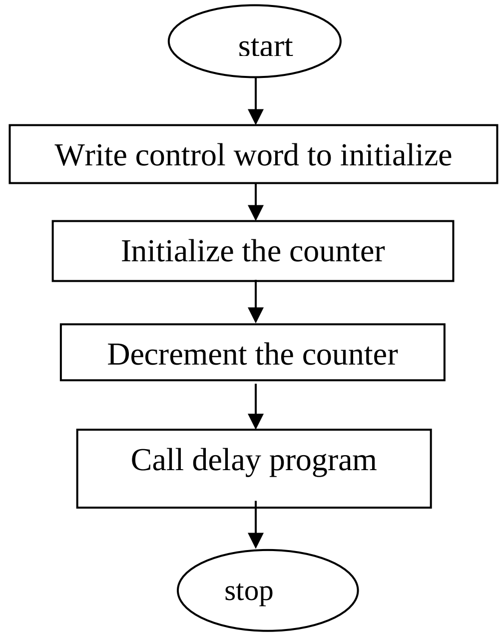
Result :

Thus assembly language program to covert the given ASCII number into its Hexadecimal equivalent is completed and also the result is verified.

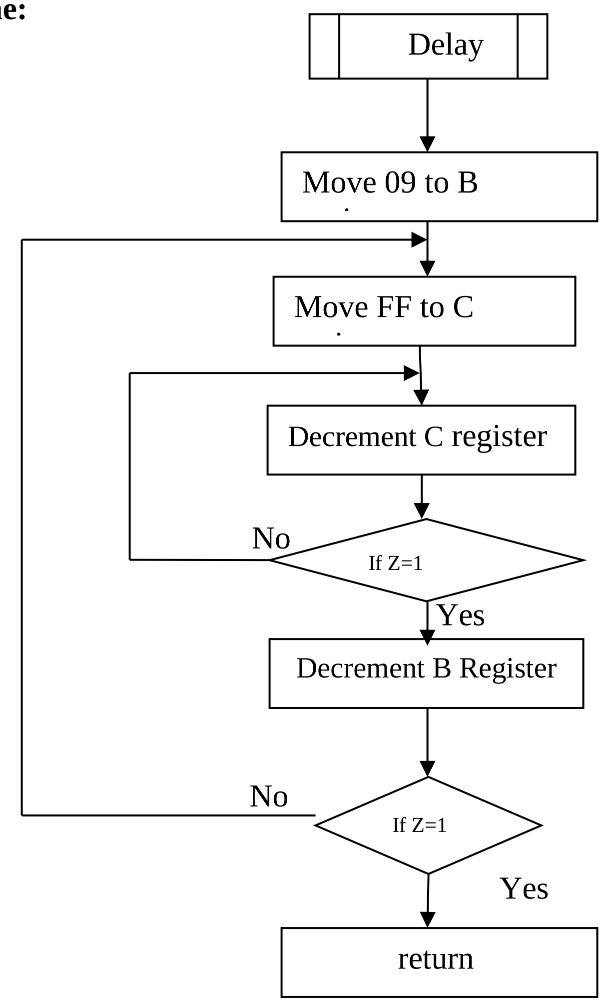
VIVA QUESTIONS:

1. What is the Hexadecimal for $(35)_{\text{ASCII}}$?
2. What is the purpose of branch instructions in 8085 microprocessor?
3. Define one's complement of an 8-bit numbers
4. What is the function of CMA instruction?
5. What is the logic behind the conversion of ASCII number into Hexadecimal number.
6. Give example for Machine control instruction?
7. What is the need of code conversion?

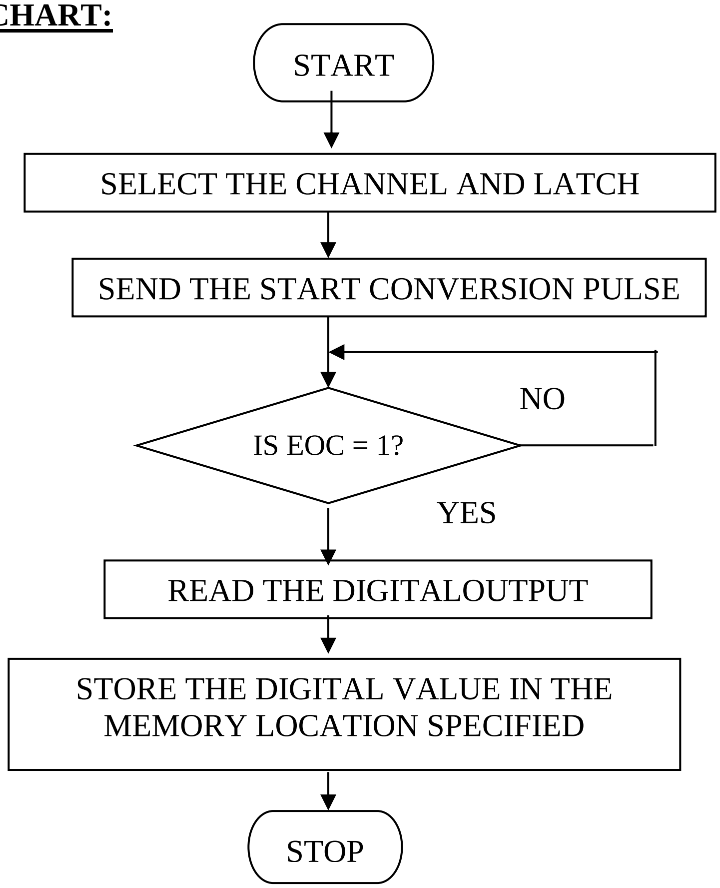
FLOW CHART:



Delay Subroutine:



FLOW CHART:


START
SELECT THE CHANNEL AND LATCH
SEND THE START CONVERSION PULSE
NO
IS EOC = 1?
YES
READ THE DIGITALOUTPUT
STORE THE DIGITAL VALUE IN THE MEMORY LOCATION SPECIFIED
STOP

3A INTERFACING ANALOG TO DIGITAL CONVERTER

AIM:

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

APPARATUS REQUIRED:

SL.NO	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8085	1
2.	Power Supply	+5 V dc,+12 V dc	1
3.	ADC Interface board	-	1

PROBLEM STATEMENT:

The program is executed for various values of analog voltage which are set with the help of a potentiometer. The LED display is verified with the digital value that is stored in a memory location.

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 0809 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 location.

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:

ADDRESSS	LABEL	OPCODE	MNEMONICS	COMMENTS
4100			MVI A,10H	Select channel
4102			OUT C8	Send through output port
4103			MVI A,18H	Load accumulator with value for ALE low
4105			OUT C8	Send through output port
4106			MVI A,01H	Store the value to make SOC high in the accumulator
4108			OUT 00H	Send through output port
4109			XRA A	Introduce delay
410A			XRA A	
410B			XRA A	
410C			MVI A,00	Store the value to make SOC low the accumulator
410E			OUT D0H	Send through output port
410F	L1		IN D8H	Read the EOC signal from port & check for end of conversion
4110			ANI 01	
4112			CPI 01	
4114			JNZ L1	If the conversion is not yet completed, read EOC signal from port again
4117			IN C0H	Read data from port
4118			STA 4150H	Store the data
411B			HLT	Stop

OBSERVATION:

ANALOG VOLTAGE (V)	DIGITAL DATA ON LED DISPLAY	HEX CODE IN MEMORY LOCATION
5	1111 1111	FF
0	0000 0000	00
2.5	1000 0000	80

RESULT:

Thus the ADC was interfaced with 8085and the given analog inputs were converted into its digital equivalent.

VIVA QUESTIONS:

1. What is the name given to time taken by the ADC from the active edge of SOC(start of conversion) pulse till the active edge of EOC(end of conversion) signal ?
2. What are the popular technique that is used in the integration of ADC chips ?
3. The procedure of algorithm for interfacing ADC contain ______________.
4. Which is the ADC among the following?

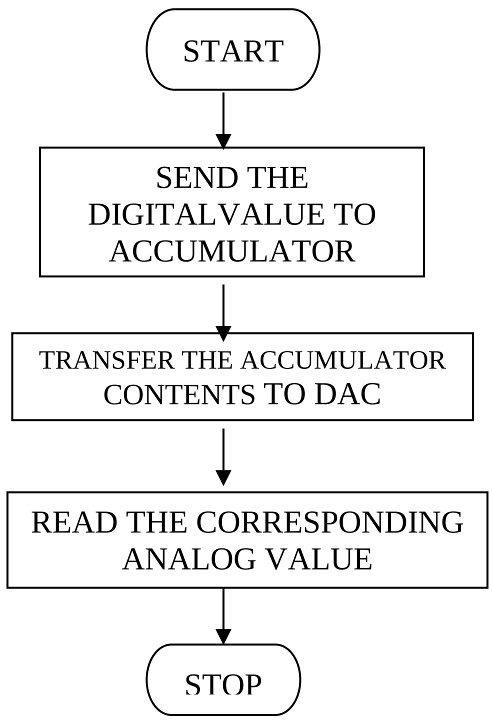
   a) AD 7523 b) 74373 c) 74245 d) ICL7109
5. The conversion delay in successive approximation of an ADC 0808/0809 is _____________.
6. The number of inputs that can be connected at a time to an ADC that is integrated with successive approximation is ___________.
7. ADC 7109 integrated by Dual slope integration technique is used for _____________________
8. Which of the following is not one of the phase of total conversion cycle?
9. Which of the following phase contain feedback loop in it?________________

   a) auto zero phase b) signal integrate phase

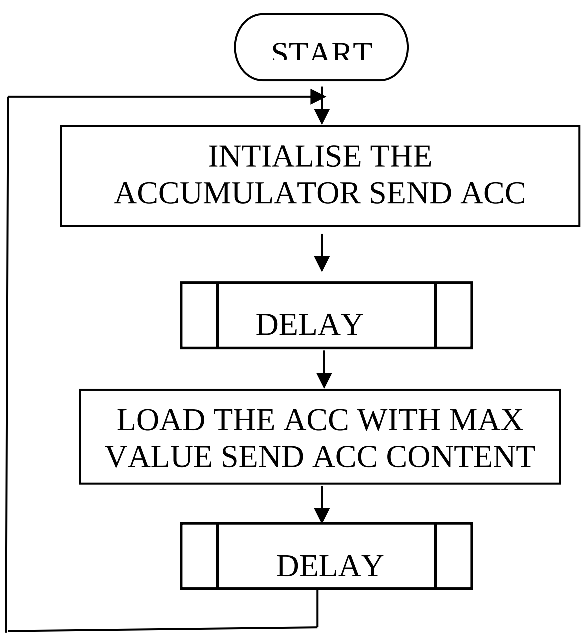
   c) deintegrate phase d) none
10. In the signal integrate phase, the differential input voltage between IN LO(input low) and IN HI(input high) pins is integrated by the internal integrator for a fixed period of ___________.

FLOWCHART:

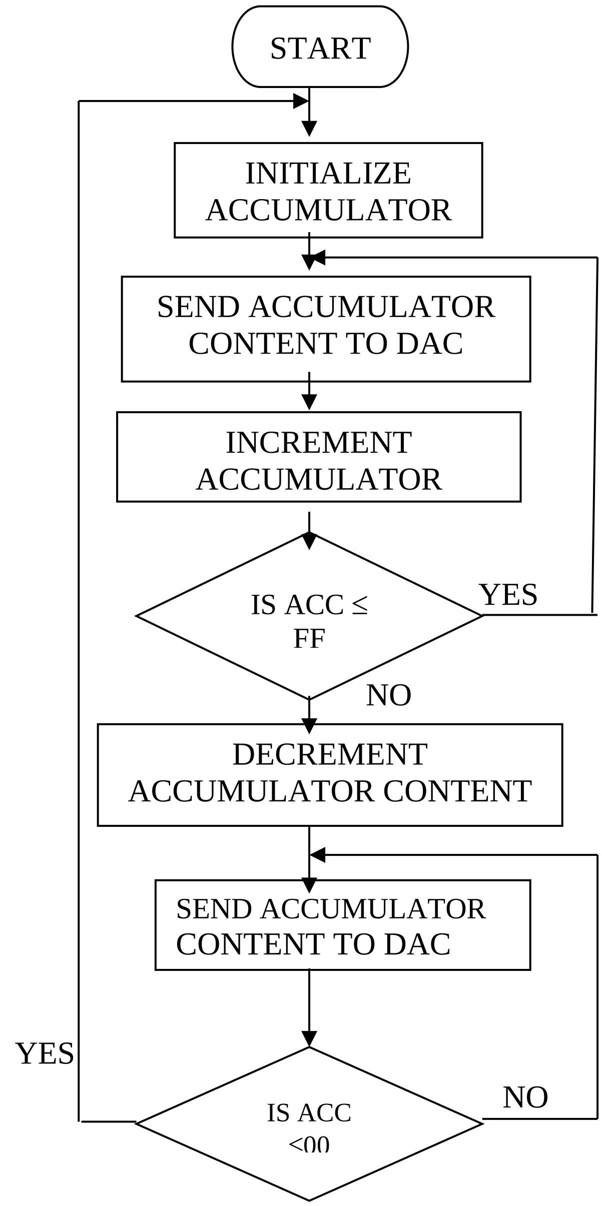
MEASUREMENT OF ANALOG VOLTAGE



SQUARE WAVE FORM



TRIANGULAR WAVE FORM



3B INTERFACING DIGITAL TO ANALOG CONVERTER

AIM:

1. To write an assembly language program for digital to analog conversion
2. To convert digital inputs into analog outputs & to generate different waveforms

APPARATUS REQUIRED:

SL.NO	ITEM	SPECIFICATION	QUANTITY
1.	**Microprocessor kit**	8086 Vi Microsystems	1
2.	**Power Supply**	+5 V, dc,+12 V dc	1
3.	**DAC Interface board**	-	1

PROBLEM STATEMENT:

The program is executed for various digital values and equivalent analog voltages are measured and also the waveforms are measured at the output ports using CRO.

THEORY:

Since 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 (approximately). The digital data input and the corresponding output voltages are presented in the table. The basic idea behind the generation of waveforms is the continuous generation of analog output of DAC. With 00 (Hex) as input to DAC2 the analog output is –5v. Similarly with FF H as input, the output is +5v. Outputting digital data 00 and FF at regular intervals, to DAC2, results in a square wave of amplitude 5v.Output digital data from 00 to FF in constant steps of 01 to DAC2. Repeat this sequence again and again. As a result a saw-tooth wave will be generated at DAC2 output. Output digital data from 00 to FF in constant steps of 01 to DAC2.Output digital data from FF to 00 in constant steps of 01 to DAC2. Repeat this sequence again and again. As a result a triangular wave will be generated at DAC2 output.

ALGORITHM:

Measurement of analog voltage:

1. Send the digital value of DAC.
2. Read the corresponding analog value of its output.

Waveform generation:

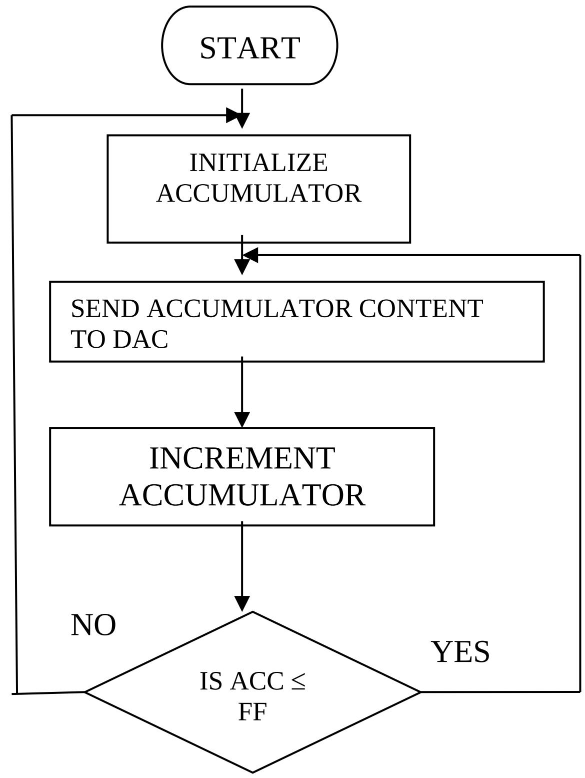
Square Waveform:

1. Send low value (00) to the DAC.
2. Introduce suitable delay.
3. Send high value to DAC.
4. Introduce delay.
5. Repeat the above procedure.

Saw-tooth waveform:

1. Load low value (00) to accumulator.
2. Send this value to DAC.
3. Increment the accumulator.
4. Repeat step (2) and (3) until accumulator value reaches FF.
5. Repeat the above procedure from step 1.

SAWTOOTH WAVEFORM



<u>PROGRAM:</u> Measurement of Analog Voltage

PROGRAM	COMMENTS
MOV A,7FH	Load digital value 00 in accumulator
OUT C0	Send through output port
HLT	Stop

OBSERVATION: Measurement of Analog Voltage

DIGITAL DATA	ANALOG VOLTAGE
FF	**5V**
00	**0V**

<u>Triangular waveform:</u>

1. Load the low value (00) in accumulator.
2. Send this accumulator content to DAC.
3. Increment the accumulator.
4. Repeat step 2 and 3 until the accumulator reaches FF, decrement the accumulator and
   send the accumulator contents to DAC.
5. Decrementing and sending the accumulator contents to DAC.
6. The above procedure is repeated from step (1)

<u>PROGRAM:</u> Square Wave

ADDRESSS	LABEL	PROGRAM	COMMENTS
4100	START	MVI A,00H	Load 00 in accumulator
4102		OUT C8	Send through output port
4103		CALL DELAY	Give a delay
4105		MVI A,0FH	Load 0F in accumulator
4107		OUT C8	Send through output port
4108		CALL DELAY	Give a delay
4109		JMP START	Go to starting location
410A	DELAY	MVI B,05	Load count value 05 in B register
410B	L1	MVI C,0F	Load count value 0F in B register
410C	L2	DCR C	Decrement C register
410E		JNZ L2	Return to loop2
410F		DCR B	Decrement B register
4110		JNZ L1	Return to loop1
4112		RET	Return to main program

<u>PROGRAM:</u> Saw tooth Wave

ADDRESSS	LABEL	PROGRAM	COMMENTS
4100	START	MVI A,00H	Load 00 in accumulator
4102	L1	OUT C0	Send through output port
4103		INR A	Increment contents of accumulator
4104		JNZ L1	Send through output port until it reaches FF
4107		JMP START	Go to starting location

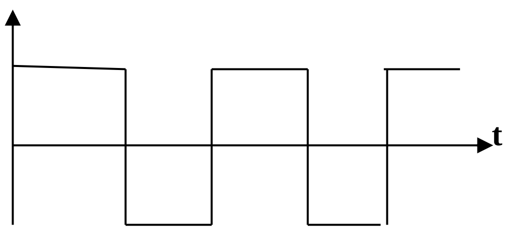
<u>PROGRAM:</u> Triangular Wave

ADDRESSS	LABEL	PROGRAM	COMMENTS
4100	START	MVI L,00H	Load 00 in accumulator
4102	L1	MOV A ,L	Move contents of L to A
4103		OUT C8	Send through output port
4104		INR C	Increment contents of accumulator
4105		JNZ L1	Send through output port until it reaches FF
4108		MVI C,FFH	Load FF in accumulator

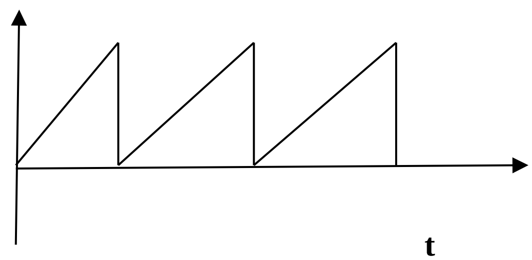
4109	L2	MOV A,L	Move contents of L to A
410A		OUT C8	Send through output port
410B		DCR C	Decrement contents of accumulator
410C		JNZ L2	Send through output port until it reaches 00
410F		JMP START	Go to starting location

MODEL GRAPH:

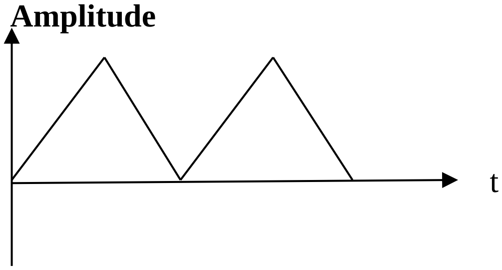
Square Waveform
Amplitude



Saw-tooth waveform
Amplitude



Triangular waveform
Amplitude



RESULT OF WAVEFORM GENERATION:

WAVEFORMS	AMPLITUDE	TIMEPERIOD
Square Waveform		
Saw-tooth waveform		
Triangular waveform		

RESULT:

Thus digital to analog conversion is done and different waveforms such as square wave, sawtooth wave and triangular wave are generated by interfacing DAC with 8085

VIVA QUESTIONS:

1. DAC (Digital to Analog Converter) finds application in (digitally controlled gains,motor speed controls, programmable gain amplifiers)
2. To save the DAC from negative transients the device connected between OUT1 and OUT2 of AD 7523 is _______________
3. An operational amplifier connected to the output of AD 7523 is used (to convert current output to output voltage , to provide additional driving capability, as current-to-voltage converter)
4. The DAC 0800 has a settling time of (100 milliseconds).
5. What is meant by the instruction OUT C8
6. Give examples for various DAC ICs?

3C . TRAFFIC LIGHT CONTROLLER - INTERFACING PPI 8255 WITH 8085

AIM:

To design traffic light controller using 8085 microprocessor through programmable peripheral interface 8255.

APPARATUS REQUIRED:

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

Traffic light controller interface board.

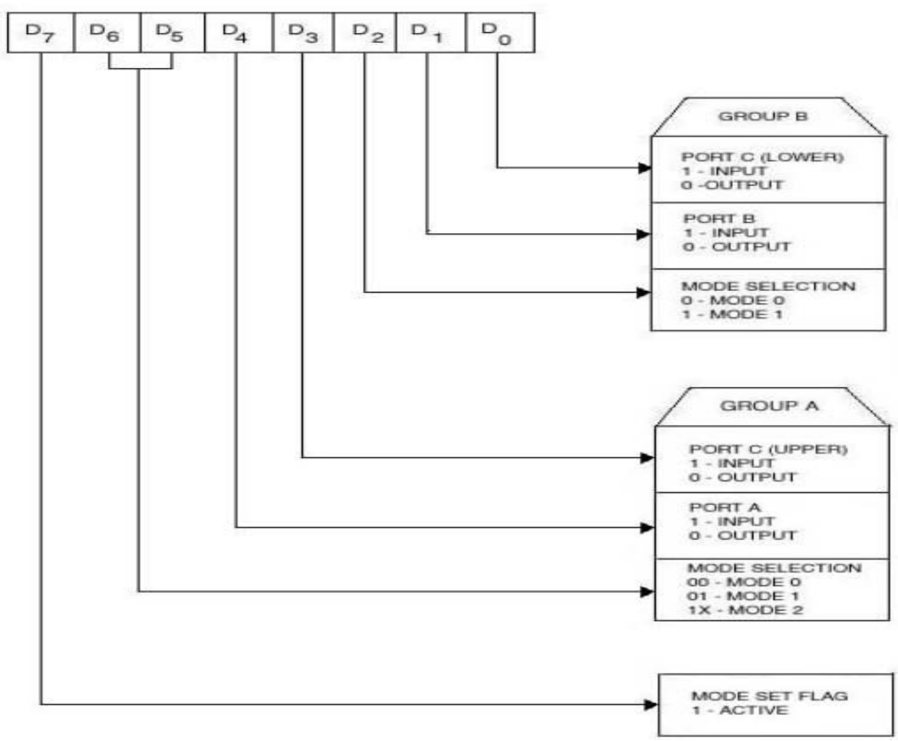
I/O MODES:

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

Control Word:



PROGRAM:

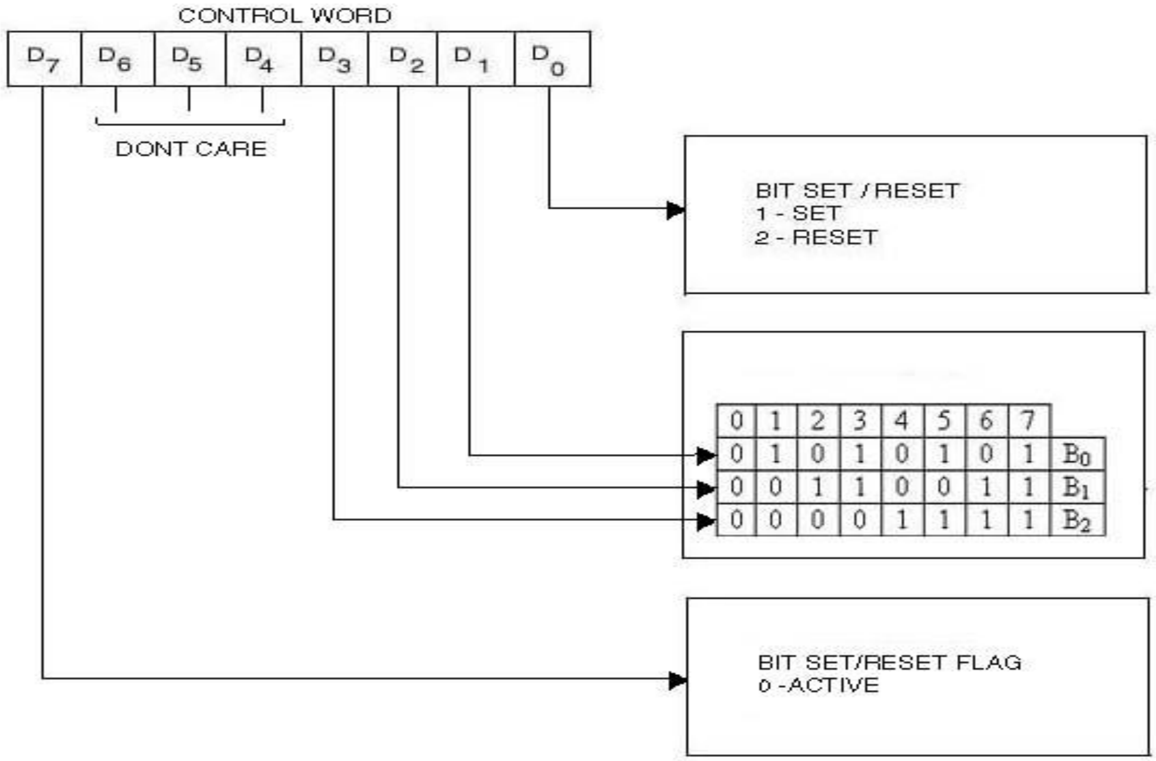
ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
4100			LXI	H, Data	Load the data in HL register pair
4103			MVI	C,04	Move 04 to c register
4105			MOV	A,M	Move M to A
4106			OUT	CNT	Out tocontrol register
4108			INX	H	Increment HL register pair
4109		LOOP1	MOV	A,M	Move M to A
410A			OUT	CPRT	Send control status word
410C			INX	H	Increment h register
410D			MOV	A,M	Move M to A
410E			OUT	BPRT	Send control status word
4110			INX	H	Increment h register
4111			MOV	A,M	Move M to A
4112			OUT	APRT	Send control status word
4114			CALL	DELAY	Call subroutine
4117			INX	H	Increment h register
4118			DCR	C	Decrement C register
4119			JNZ	LOOP1	Jump on nozero to loop1
411C			JMP	START	Jump to start
411F		DELAY	PUSH	B	
4120			MVI	C.0D	Move OD to C register
4122		LOOP3	LXI	D,FF,FF	Load Dregister with FF
4125		LOOP2	DCX	D	Decrement Dregister
4126			MOV	A,D	Move D contents to A register
4127			ORA	E	OR the content of A with E
4128			JNZ	LOOP2	Jump on nozero to loop2
412C			JNZ	LOOP3	Jump on nozero to loop3
412F			POP	B	Do pop operation
4130			RET		Return to main program

MODE 1 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

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.

ALGORITHM:-

1. Start.
2. Write the control word to initialize 8255.Obtain the data for each direction and store in the memory.
3. Initialize a counter to indicate the number of directions.
4. Initialize HL Pair to the starting address of the data..
5. Check the result.
6. Decrement the counter and repeat step 3 till counter becomes 0
7. Stop

RESULT:

Thus the design of traffic light controller using 8085 microprocessor through programmable peripheral interface 8255is done and also the output is verified.

VIVA QUESTIONS:

1. When the 82C55 is reset, its I/O ports are all initializes as what?
2. If the programmable counter timer 8254 is set in mode 1 and is to be used to count six events,
   the output will remain at logic 0 for how many number of counts ?
3. The devices that provide the means for a computer to communicate with the user or other computers are referred to as:
4. What is the maximum number of I\o devices which can be interfaced in the memory mapped
   I\O technique?
5. Interaction between a CPU and a peripheral device that takes place during and input output operation is known as what?
6. What is the other name for Programmable peripheral input-output port?
7. All the functions of the ports of 8255 are achieved by programming the bits of an internal register called what?
8. What is the port that is used for the generation of handshake lines in mode 1 or mode 2 ?
9. What is the pin that clears the control word register of 8255 when enabled?
10. In 8255 if A1=0, A0=1 then the input read cycle is performed from where?

4 STEPPER MOTOR INTERFACING WITH 8085

AIM:

To operate stepper motor by interfacing with 8085 microprocessor.

THEORY:

Stepper Motor

A stepper motor is a device that translates electrical pulses into mechanical movement in steps of fixed step angle.

- The stepper motor rotates in steps in response to the applied signals.
- It is mainly used for position control.
- It is used in disk drives, dot matrix printers, plotters and robotics and process control circuits.

Structure

Stepper motors have a permanent magnet called rotor (also called the shaft) surrounded by a stator. The most common stepper motors have four stator windings that are paired with a center-tap. This type of stepper motor is commonly referred to as a four-phase or unipolar stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.

Interfacing

Even a small stepper motor require a current of 400 mA for its operation. But the ports of the microcontroller cannot source this much amount of current. If such a motor is directly connected to the microprocessor/microcontroller ports, the motor may draw large current from the ports and damage it. So a suitable driver circuit is used with the microprocessor/microcontroller to operate the motor.

Motor Driver Circuit (ULN2003)

Stepper motor driver circuits are available readily in the form of ICs. ULN2003 is one such driver IC

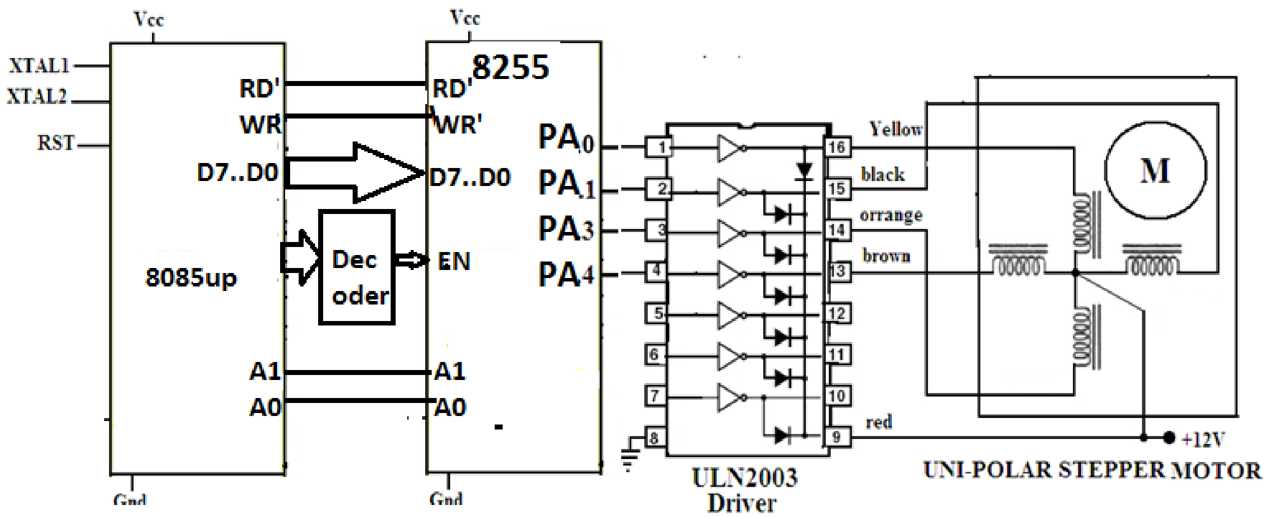
Motor Driver Circuit (ULN2003)

Stepper motor driver circuits are available readily in the form of ICs. ULN2003 is one such driver IC which is a High-Voltage High-Current Darlington transistor array and can give a current of 500mA.This current is sufficient to drive a small stepper motor. Internally, it has protection diodes used to protect the motor from damage due to back emf and large eddy currents. So, this ULN2003 is used as a driver to interface the stepper motor to the microprocessor.

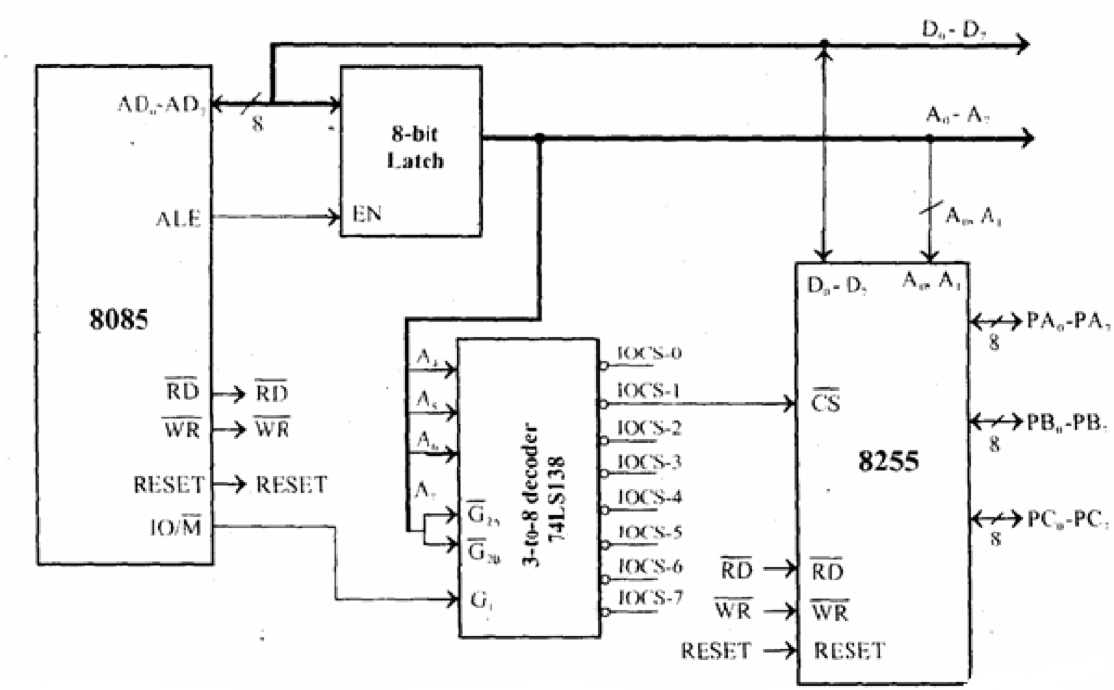
Operation

The important parameter of a stepper motor is the **step angle**. It is the minimum angle through which the motor rotates in response to each **excitation pulse**. In a four phase motor if there are 200 steps in one complete rotation then then the step angle is $360/200 = 1.8^{o}$. So to rotate the stepper motor we have to apply the excitation pulse. For this the controller should send a hexa decimal code through one of its ports. **The hex code mainly depends on the construction of the stepper motor**. So, all the stepper motors do not have the same Hex code for their rotation. (refer the operation manual supplied by the manufacturer.)

For example, let us consider the hex code for a stepper motor to rotate in clockwise direction is 77H , BBH , DDH and EEH. This hex code will be applied to the input terminals of the driver through the assembly language program. To rotate the stepper motor in anti-clockwise direction the same code is applied in the reverse order.



Detailed Connection diagram between 8085 and 8255



PROGRAM :

Address	Label	Mnemonics	Operand	Comments
	Main		MVI A, 80 ;	80H → Control word to configure PA,PB,PC in O/P
			OUT CWR_Address ;	Write control word in CWR of 8255
			MVI A, 77 ;	Code for the Phase 1
			OUT PortA_Address ;	sent to motor via port A of 8255 ;
			CALL DELAY ;	Delay subroutine
			MVI A, BB ;	Code for the Phase II
			OUT PortA_Address ;	sent to motor via port A of 8255
			CALL DELAY ;	Delay subroutine
			MVI A, DD ;	Code for the Phase III
			OUT PortA_Address ;	sent to motor via port A of 8255;
			CALL DELAY ;	Delay subroutine
			MVI A, EE H ;	Code for the Phase 1
			OUT PortA_Address	sent to motor via port A of 8255
			CALL DELAY ;	Delay subroutine
			JMP MAIN	; Keep the motor rotating continuously.
	DELAY		MVI C, FF	; Load C with FF -- Change it for the speed variation
	LOOP1:		MVI D,FF ;	Load D with FF
	LOOP2:		DCR D	
			JNZ LOOP2	
			DCR C	
			JNZ LOOP1	
			RET ;.	Return to main program

PROCEDURE:

- Enter the above program starting from location 4100 and execute the same. The stepper motor rotates.
- By varying the count at C and D register can vary the speed.
- By entering the data in the look-up TABLE in the reverse order can vary direction of rotation.

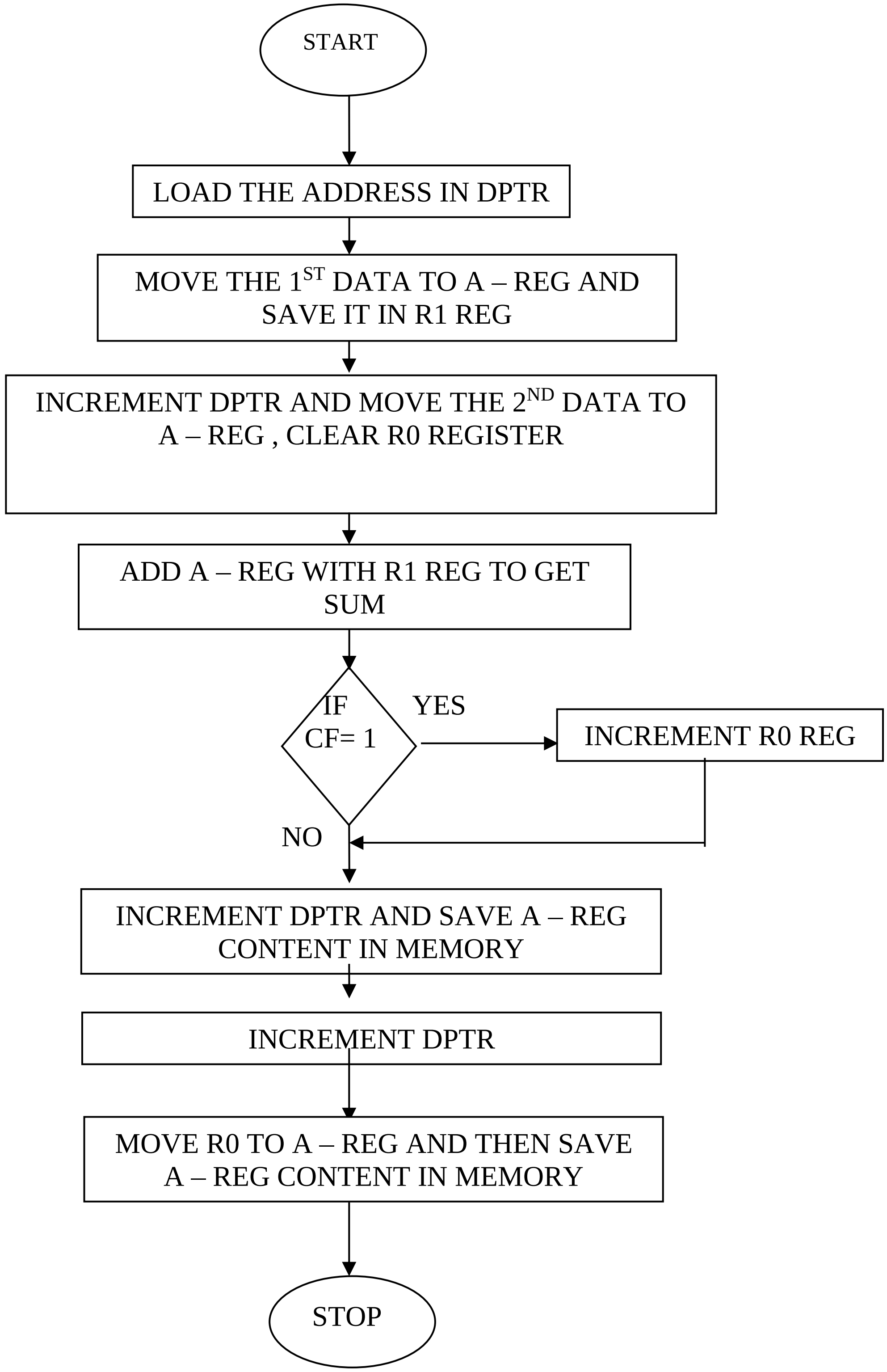
RESULT:

Thus a stepper motor was interfaced with 8085 and run in forward and reverse directions at various speeds.

VIVA QUESTIONS:

1. What are the application of stepper motor?
2. What is meant by step angle?
3. What are the methods to control the speed of stepper motor?
4. What is the formula for steps per revolution?
5. How a stepper motor differs from DC motor?

FLOW CHART:


START
LOAD THE ADDRESS IN DPTR
MOVE THE 1ST DATA TO A – REG AND SAVE IT IN R1 REG
INCREMENT DPTR AND MOVE THE 2ND DATA TO A – REG , CLEAR R0 REGISTER
ADD A – REG WITH R1 REG TO GET SUM
IF CF= 1
YES
INCREMENT R0 REG
NO
INCREMENT DPTR AND SAVE A – REG CONTENT IN MEMORY
INCREMENT DPTR
MOVE R0 TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

5(A) 8-BIT ADDITION

AIM:

To write a program to add two 8-bit numbers using 8051 microcontroller and also to verify the result.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

ALGORITHM:

1. Clear Program Status Word.
2. Select Register bank by giving proper values to RS1 & RS0 of PSW.
3. Load accumulator A with any desired 8-bit data.
4. Load the register R_0 with the second 8- bit data.
5. Add these two 8-bit numbers.
6. Store the result.
7. Stop the program.

PROGRAM:

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		CLR	C		Clear CY Flag
4101		MOV	A,# data1		Get the data1 in Accumulator
4103		ADDC	A, # data 2		Add the data1 with data2
4105		MOV	DPTR, # 4500H		Initialize the memory location
4108		MOVX	@ DPTR, A		Store the result in memory location
4109	L1	SJMP	L1		Stop the program

OBSERVATION:

OUTPUT	
MEMORY LOCATION	**DATA**
Data1: 08,data2: 07 4500	0F

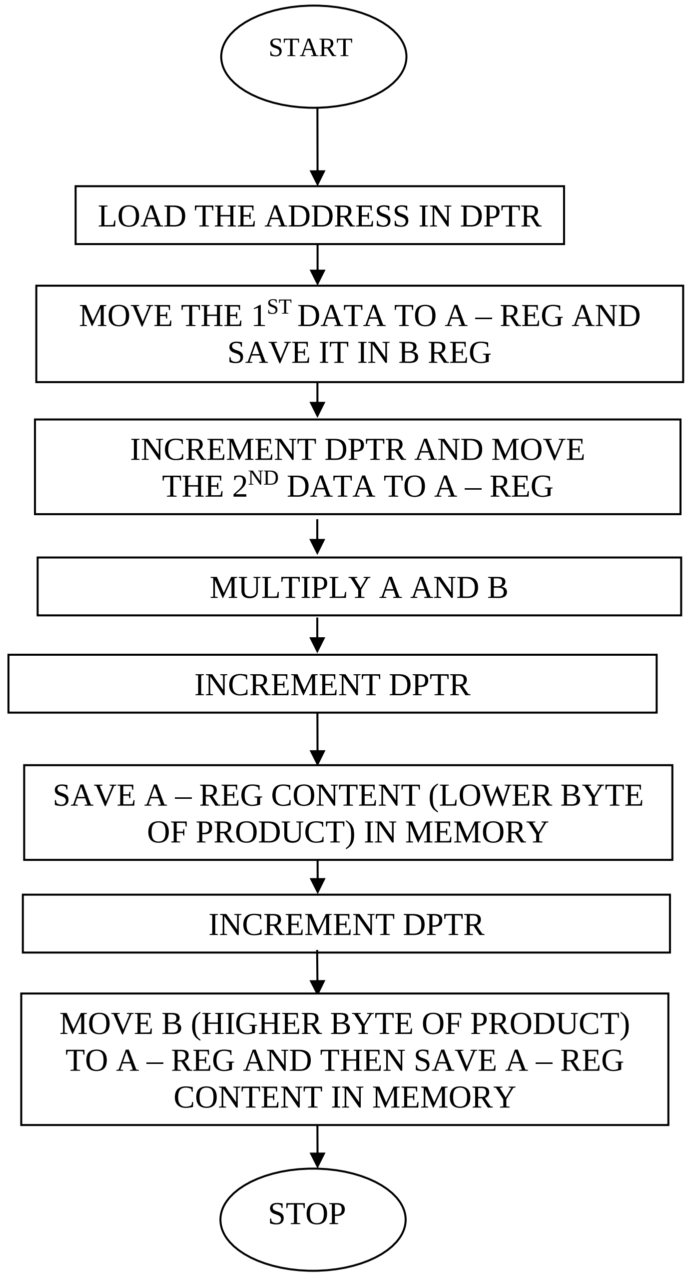
RESULT:

Thus the 8051 ALP for addition of two 8 bit numbers is executed and the result is verified.

VIVA QUESTIONS:

1. Which type of addressing mode is MOV A,# data1 ?
2. Explain SJMP ?
4. Explain ADDC A,# data1 ?
5. If RS1=1, RS0=0, then the register bank selected is (register bank 2) ?
6. What are the various ways to clear the carry flag?

FLOWCHART:


START
LOAD THE ADDRESS IN DPTR
MOVE THE 1ST DATA TO A – REG AND SAVE IT IN B REG
INCREMENT DPTR AND MOVE THE 2ND DATA TO A – REG
MULTIPLY A AND B
INCREMENT DPTR
SAVE A – REG CONTENT (LOWER BYTE OF PRODUCT) IN MEMORY
INCREMENT DPTR
MOVE B (HIGHER BYTE OF PRODUCT) TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

5(B) 8-BIT SUBTRACTION

AIM:

To perform subtraction of two 8 bit data using the 8051 microcontroller and store the result in memory.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

ALGORITHM:

1. Clear the carry flag.
2. Initialize the register for borrow.
3. Get the first operand into the accumulator.
4. Subtract the second operand from the accumulator.
5. If a borrow results increment the carry register.
6. Store the result in memory.

PROGRAM:

ADDRESS	LABEL	MNEMONIC	OPERAND	HEXCODE	COMMENTS
4100		CLR	C		Clear CY flag
4101		MOV	A, # data1		Store data1 in accumulator
4103		SUBB	A, # data2		Subtract data2 from data1
4105		MOV	DPTR, # 4500		Initialize memory location
4108		MOVX	@ DPTR, A		Store the difference in memory location
4109	L1	SJMP	L1		Stop

OBSERVATION:

OUTPUT	
MEMORY LOCATION	**DATA**
Data 1,2 : 08,07 4500	01

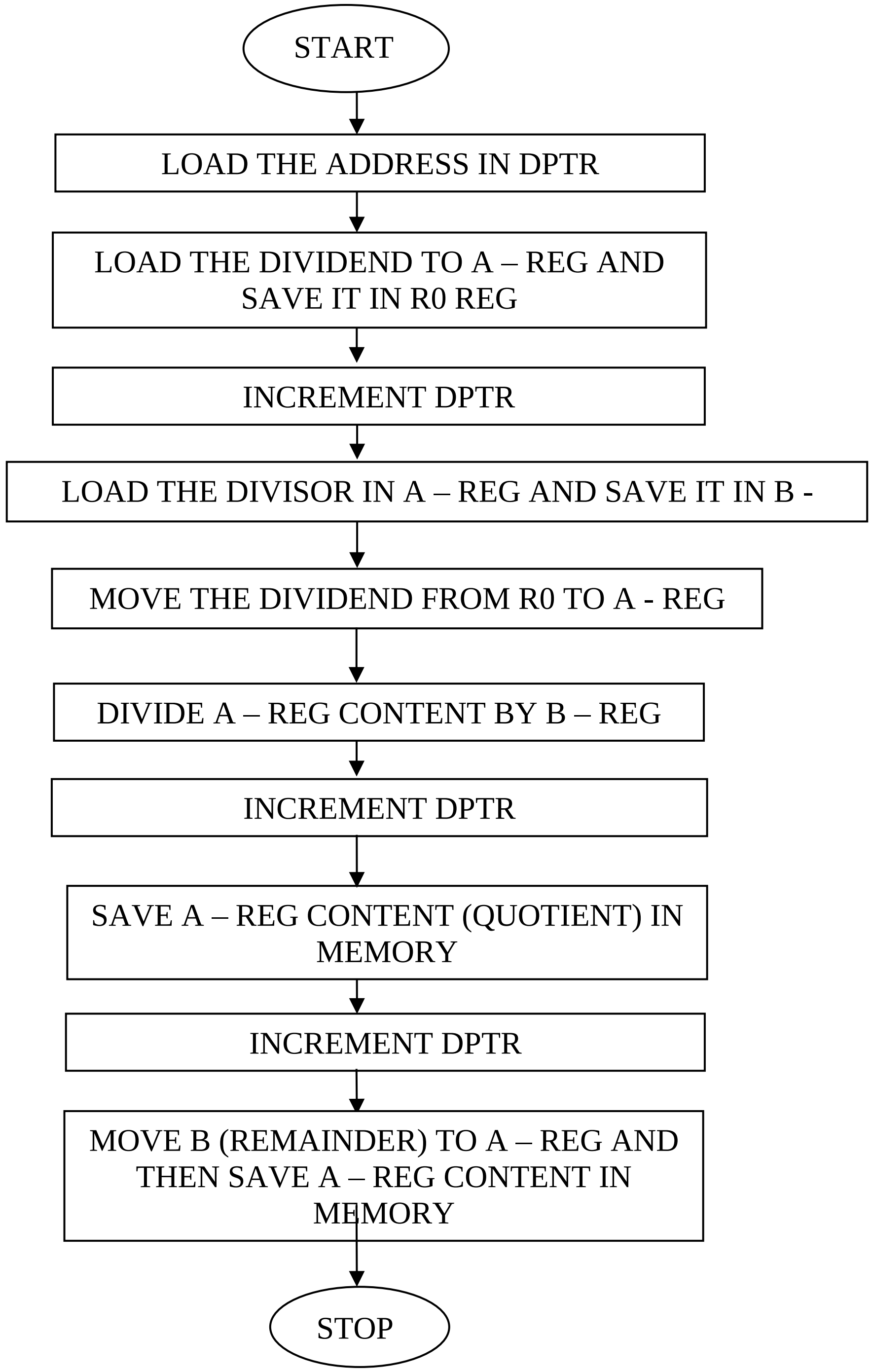
RESULT:

Thus the 8051 ALP for subtraction of two 8 bit numbers is executed and the result is verified.

VIVA QUESTIONS:

1. How SUBB instruction works?
2. The instruction, ADD A, R7 is an example of ____________________ instruction
3. What is meant by PSW ?
4. List out the difference between MOV and MOVX instructions
5. What is the use of DPTR
6. Tell about counter mode in8051.
7. What is the SCON register in 8051?

FLOWCHART:


START
LOAD THE ADDRESS IN DPTR
LOAD THE DIVIDEND TO A – REG AND SAVE IT IN R0 REG
INCREMENT DPTR
LOAD THE DIVISOR IN A – REG AND SAVE IT IN B -
MOVE THE DIVIDEND FROM R0 TO A - REG
DIVIDE A – REG CONTENT BY B – REG
INCREMENT DPTR
SAVE A – REG CONTENT (QUOTIENT) IN MEMORY
INCREMENT DPTR
MOVE B (REMAINDER) TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

5(C) 8-BIT MULTIPLICATION

AIM:

To perform multiplication of two 8 bit data using 8051 microcontroller and to store the result in memory.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

ALGORITHM:

1. Get the multiplier in the accumulator.
2. Get the multiplicand in the B register.
3. Multiply A with B.
4. Store the product in memory.

PROGRAM:

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		MOV	A ,#data1		Store data1 in accumulator
4102		MOV	B, #data2		Store data2 in B reg
4104		MUL	A,B		Multiply both
4106		MOV	DPTR, # 4500H		Initialize memory location
4109		MOVX	@ DPTR, A		Store lower order result
401A		INC	DPTR		Go to next memory location
410B		MOV	A,B		Store higher order result
410D		MOV	@ DPTR, A		
410E	STOP	SJMP	STOP		Stop

Data1:04
Data 2:02

OBSERVATION:

OUTPUT	
MEMORY LOCATION	**DATA**
4500	08
4501	00

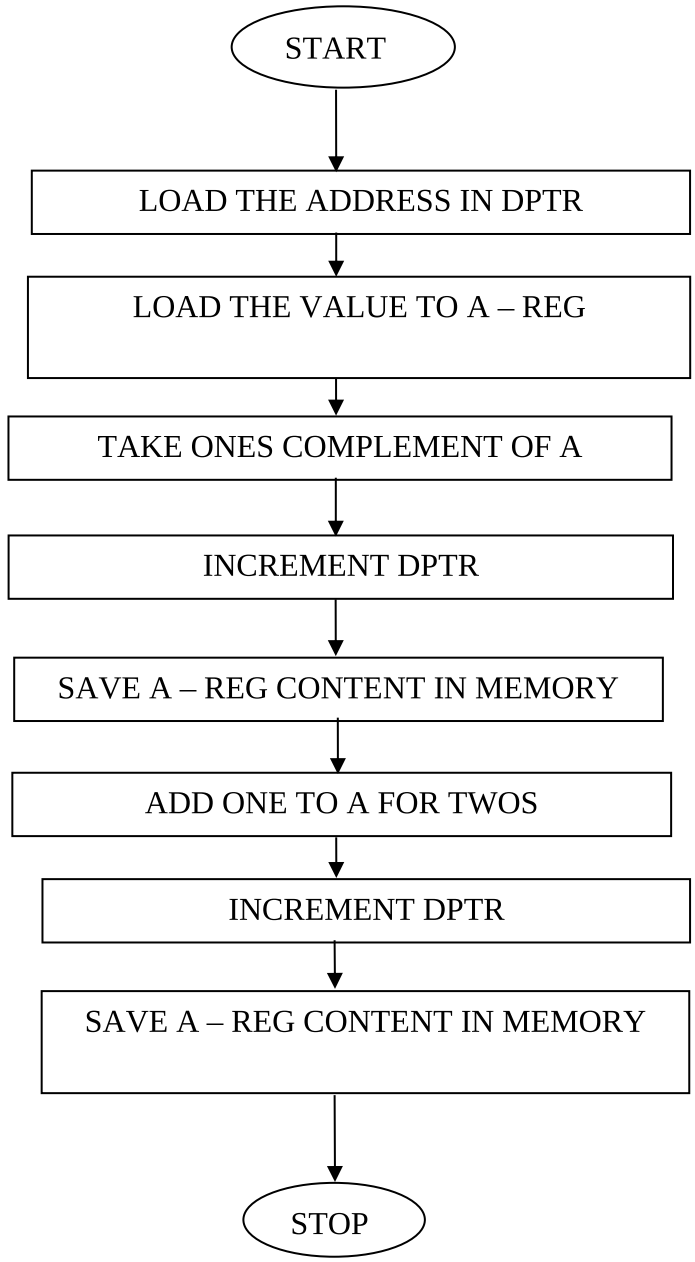
RESULT:

Thus the 8051 ALP for multiplication of two 8 bit numbers is executed and the result is verified.

VIVA QUESTIONS:

1. Give the syntax of multiplication instruction.
2. What is the use of INC DPTR instruction ?
3. What is the use of EA signal in 8051?
4. What is the role of RS0,RS1 bits?
5. What is the use of PSEN pin in 8051?
6. What is the syntax of Division instruction?
7. What is the difference between the SJMP and LJMP?

FLOWCHART:


START
LOAD THE ADDRESS IN DPTR
LOAD THE VALUE TO A – REG
TAKE ONES COMPLEMENT OF A
INCREMENT DPTR
SAVE A – REG CONTENT IN MEMORY
ADD ONE TO A FOR TWOS
INCREMENT DPTR
SAVE A – REG CONTENT IN MEMORY
STOP

5(D) 8-BIT DIVISION

AIM:

To perform division of two 8 bit data using 8051 microcontroller and to store the result in memory.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

ALGORITHM:

1. Get the Dividend in the accumulator.
2. Get the Divisor in the B register.
3. Divide A by B.
4. Store the Quotient and Remainder in memory.

PROGRAM:

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		MOV	A, # data1		Store data1 in accumulator
4102		MOV	B, # data2		Store data2 in B reg
4104		DIV	A,B		Divide
4015		MOV	DPTR, # 4500H		Initialize memory location
4018		MOVX	@ DPTR, A		Store remainder
4109		INC	DPTR		Go to next memory location
410A		MOV	A,B		Store quotient
410C		MOV	@ DPTR, A		
410D	STOP	SJMP	STOP		Stop

Data 1: 08
Data 2 :02

OBSERVATION:

OUTPUT	
MEMORY LOCATION	**DATA**
4500 (remainder)	00
4501 (quotient)	04

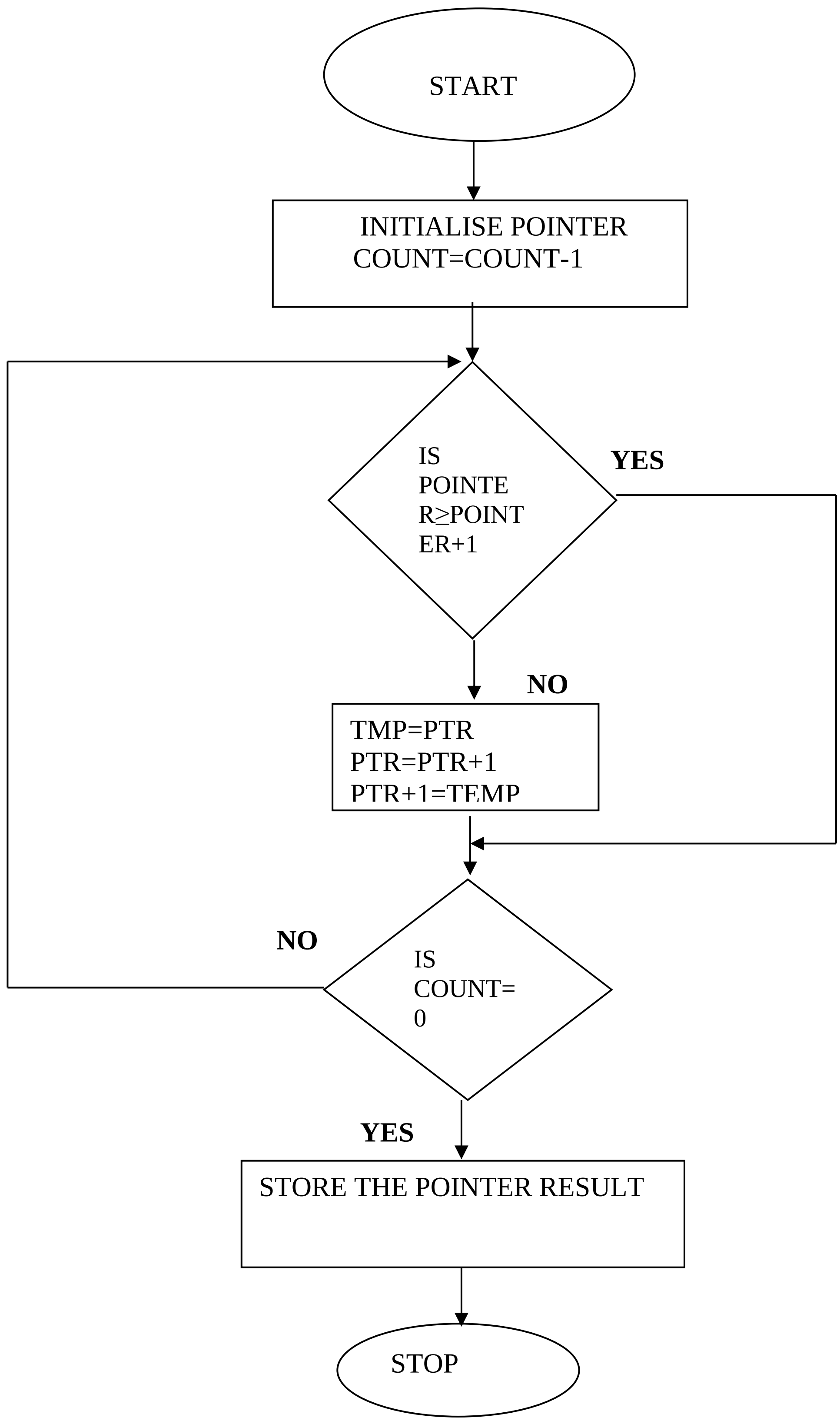
RESULT:

Thus the 8051 ALP for division of two 8 bit numbers is executed and the result is verified.

VIVA QUESTIONS:

1. How division is performed in microcontroller?
2. In which register quotient and remainder is stored?
3. What is SJMP?
4. What is the syntax of Division instruction?
5. What are control and status register?
6. DIV AB is an______________ bit instruction?
7. What is the meant by the instruction DPTR, # 4500H ?

FLOW CHART:


START
INITIALISE POINTER
COUNT=COUNT-1
IS
POINTE
R≥POINT
ER+1
YES
NO
TMP=PTR
PTR=PTR+1
PTR+1=TEMP
NO
IS
COUNT=
0
YES
STORE THE POINTER RESULT
STOP

6(A) LARGEST ELEMENT IN AN ARRAY

AIM:

To write an assembly language program to find the largest element in an array and to execute it using 8051 .

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

.

ALGORITHM

1. Start.
2. Load the array count in a register
3. Get the first two numbers.
4. Compare the numbers and swap them so that the two numbers are in ascending order.
5. Repeat steps 3 and 4 till the array is completed.
6. Repeat the steps 3, 4 and 5 and store the largest number as the result in memory.
7. Stop.

PROGRAM:

MEMORY ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
4100			MOV DPTR,#4200	Load location 4200 to DPTR
4103			MOV 40,#00	Load zero to memory 40H
4106			MOV R5, #07	Move array size to R5
4108		LOOP2	MOVX A,@DPTR	Accumulator is moved to16 bit External Memory address indicated by DPTR
4109			CJNE A,40 LOOP1	Compare A with contents of location 40H and Jump if Not Equal to LOOP1
410C		LOOP3	INC DPTR	Increment DPTR content
410D			DJNZ R5,LOOP2	Decrement Register R5 and Jump if Not Zero to LOOP2
410F			MOV A,40	Move value in location 40H to Accumulator
4111			MOVX @DPTR,A	Accumulator is moved to16 bit External Memory address indicated by DPTR
4112		HLT	SJMP HLT	Stop the execution
4114		LOOP1	JC LOOP3	Jump if Carry Set to LOOP3
4116			MOV 40,A	Move A to location 40H
4118			SJMP LOOP2	Perform short jump to location LOOP2

OUTPUT:

MEMORY ADDRESS	**INPUT VALUES**	**MEMORY ADDRESS**	**OUTPUT VALUES**
4200	07	4207	09
4201	08		
4202	02		
4203	01		
4204	04		
4205	03		
4206	09		

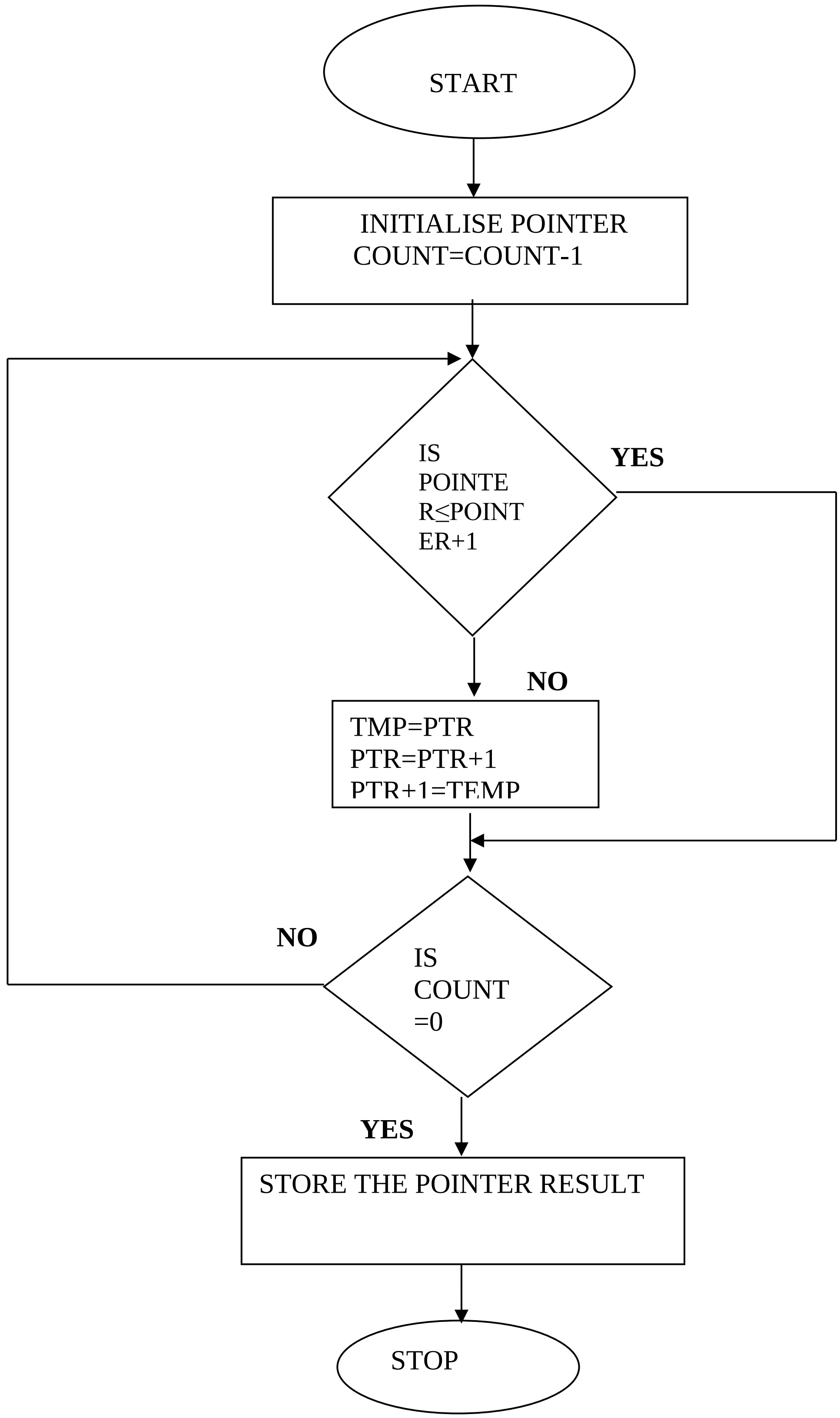
RESULT:

Thus an assembly language program written to find the largest element in an array was executed using 8051 microcontroller and the output was verified.

VIVA QUESTIONS:

1. Explain CJNE A,40 LOOP1
2. The instruction, RLA performs -------------------------------
3. What does the instruction, ADD A, #100 performs?
4. What does the instruction, DJNZ performs?
5. Give example for jump instruction ?
6. What is use of the instruction MOVX @DPTR,A
7. What are one byte instruction in 8051 ?

FLOW CHART:


START
INITIALISE POINTER
COUNT=COUNT-1
IS POINTER≤POINTER+1
YES
NO
TMP=PTR
PTR=PTR+1
PTR+1=TEMP
NO
IS COUNT =0
YES
STORE THE POINTER RESULT
STOP

6(B) SMALLEST ELEMENT IN AN ARRAY

AIM:

To write an assembly language program to find the largest element in an array and to execute it using 8051 microprocessor.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

ALGORITHM

1. Start.
2. Load the array count in a register
3. Get the first two numbers.
4. Compare the numbers and swap them so that the two numbers are in ascending order.
5. Repeat steps 3 and 4 till the array is completed.
6. Repeat the steps 3, 4 and 5 and store the largest number as the result in memory.
7. Stop.

PROGRAM:

MEMORY ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
4100			MOV DPTR,#4200	Load location 4200 to DPTR
4103			MOV 40,#00	Load zero to memory 40H
4106			MOV R5,#07	Move Array size to R5
4108		LOOP2	MOVX A,@DPTR	Accumulator is moved to16 bit External Memory address indicated by DPTR
4109			CJNE A,40 LOOP1	Compare A with contents of location 40H and Jump if Not Equal to LOOP1
410C		LOOP3	INC DPTR	Increment DPTR content
410D			DJNZ R5,LOOP2	Decrement Register R5 and Jump if Not Zero to LOOP2
410F			MOV A,40	Move value in location 40H to Accumulator
4111			MOVX @DPTR,A	Accumulator is moved to16 bit External Memory address indicated by DPTR
4112		HLT	SJMP HLT	Stop the execution
4114		LOOP1	JC LOOP3	Jump if Carry Set to LOOP3
4116			MOV 40,A	Move A to location 40H
4118			SJMP LOOP2	Perform short jump to location LOOP2

OUTPUT:

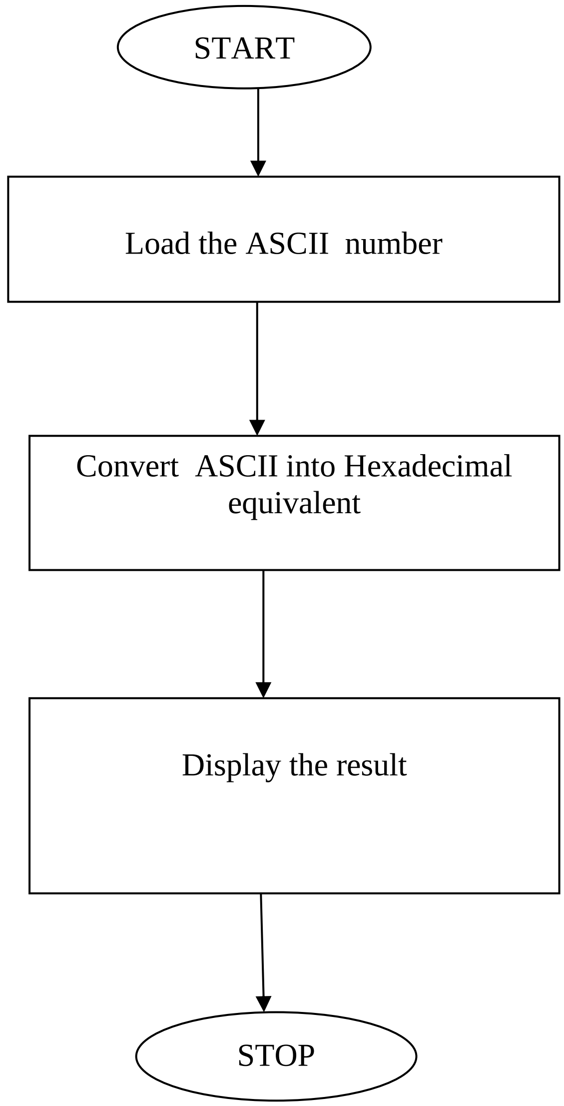
MEMORY ADDRESS	INPUT VALUES	MEMORY ADDRESS	OUTPUT VALUES
4200	07	4207	01
4201	08		
4202	02		
4203	01		
4204	04		
4205	03		
4206	09		

RESULT:

Thus an assembly language program written for finding the smallest element in an array was executed using 8051 microcontroller and the output was verified.

VIVA QUESTIONS:

1. Explain the instruction MOVX DPTR,A
2. How internal RAM is accessed?
3. Which location is used for bit manipulation instruction?
4. What happens upon execution of the instruction MOV 40,A ____________
5. What is the need of the instruction INC DPTR?
6. Expand IP and IE ?
7. How many ports are available in 8051?


START
Load the ASCII number
Convert ASCII into Hexadecimal equivalent
Display the result
STOP

6(C) ASCII to Hexadecimal conversion

AIM:

To add two 8 bit numbers stored at consecutive memory locations and also to verify the result.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board

ALGORITHM:

1)Initialize R0 with number which is required to find equivalent Hexadecimal number code.

2)Compare it with 40H and jump to label1 if it is equal.

3)Compare the carry bit to find which is greater and lesser.

4)If the carry bit is not set(it implies it is greater)jump to label2.

5)If it is lesser subtract the number with 30H.

6)If it is greater subtract the number with 37H.

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
		START	MOV R0,#41H		Move the numbers to be converted
			MOV A,R0		
			CJNE	A,#40H,LABEL1	Compare the no with 40H
		LABEL1	JNC	LABEL2	If the number is greater than 40H jump to LABEL2
			CLR C		
			SUBB	A,#30H	If the number is less than 40 subtract with 30H
			SJMP STOP		
		LABEL2	CLR C		
			SUBB	A,#37H	If the number is less than 40 subtract with 37H
		STOP	END		

OBSERVATION:

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
4200	41	4201	0A

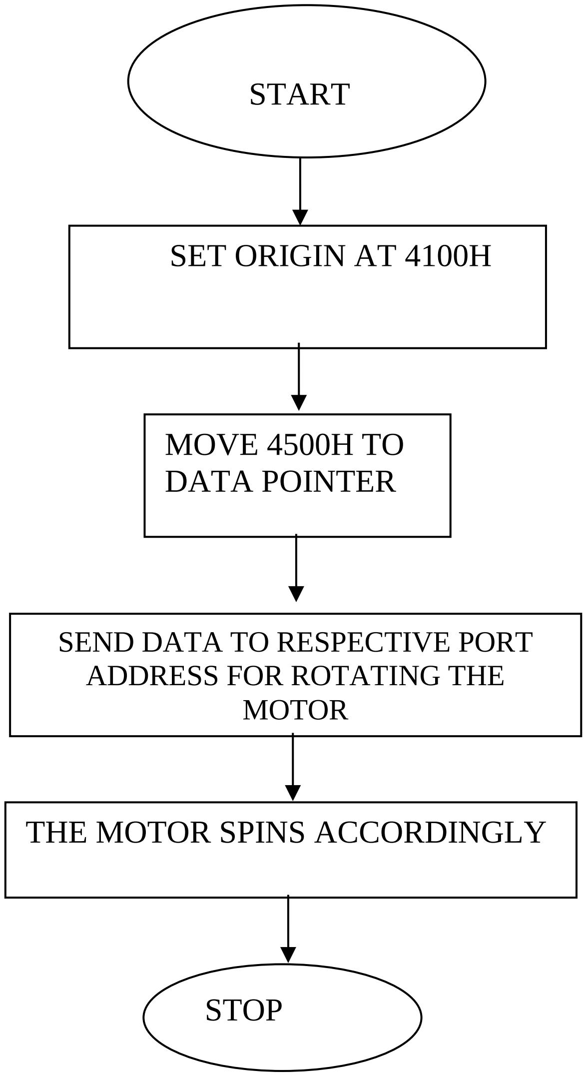
Result :

Thus assembly language program to covert the given ASCII number into its Hexadecimal equivalent is completed and also the result is verified.

VIVA QUESTIONS:

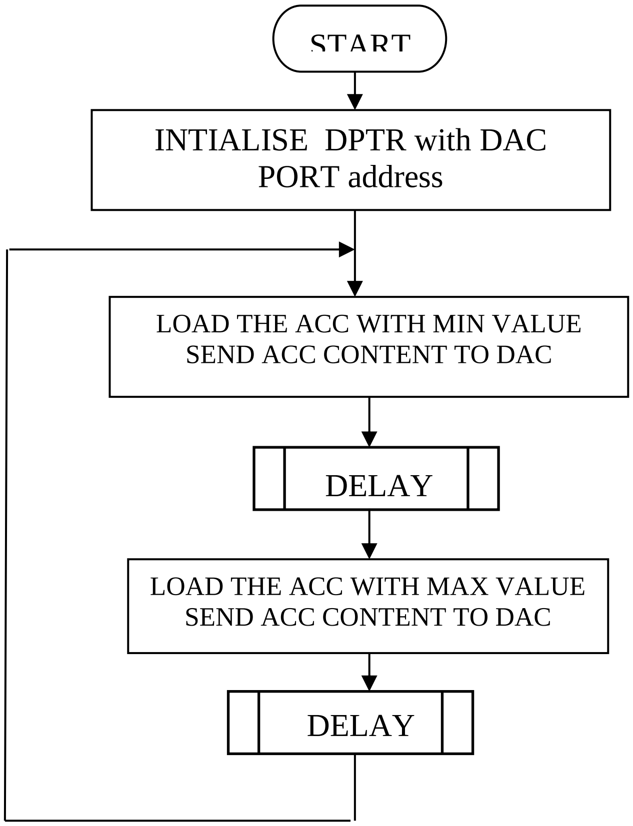
1. What is the Hexadecimal for $(35)_{ASCII}$?
2. What is the purpose of branch instructions in 8085 microprocessor?
3. Define one’s complement of an 8-bit numbers
4. What is the function of CMA instruction?
5. What is the logic behind the conversion of ASCII number into Hexadecimal number.
6. Give example for Machine control instruction?
7. What is the need of code conversion?

FLOW CHART:

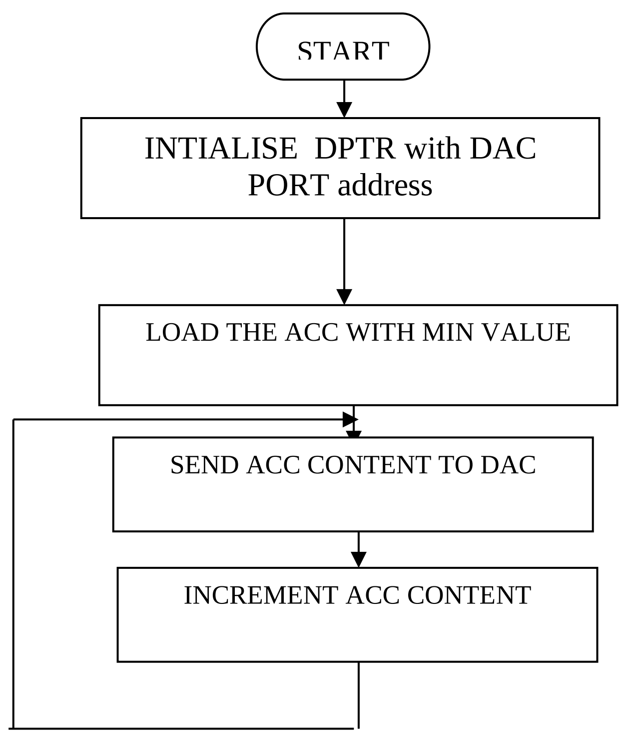

START
SET ORIGIN AT 4100H
MOVE 4500H TO DATA POINTER
SEND DATA TO RESPECTIVE PORT ADDRESS FOR ROTATING THE MOTOR
THE MOTOR SPINS ACCORDINGLY
STOP

FLOWCHART:

SQUARE WAVE FORM:



SAWTOOTH WAVE FORM:



7 INTERFACING DAC WITH 8051

AIM:

To interface DAC with 8051 to demonstrate the generation of square wave, triangular wave and sawtooth wave

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

APPARATUS REQUIRED:

8051 Trainer Kit, DAC interface board

ALGORITHM:

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

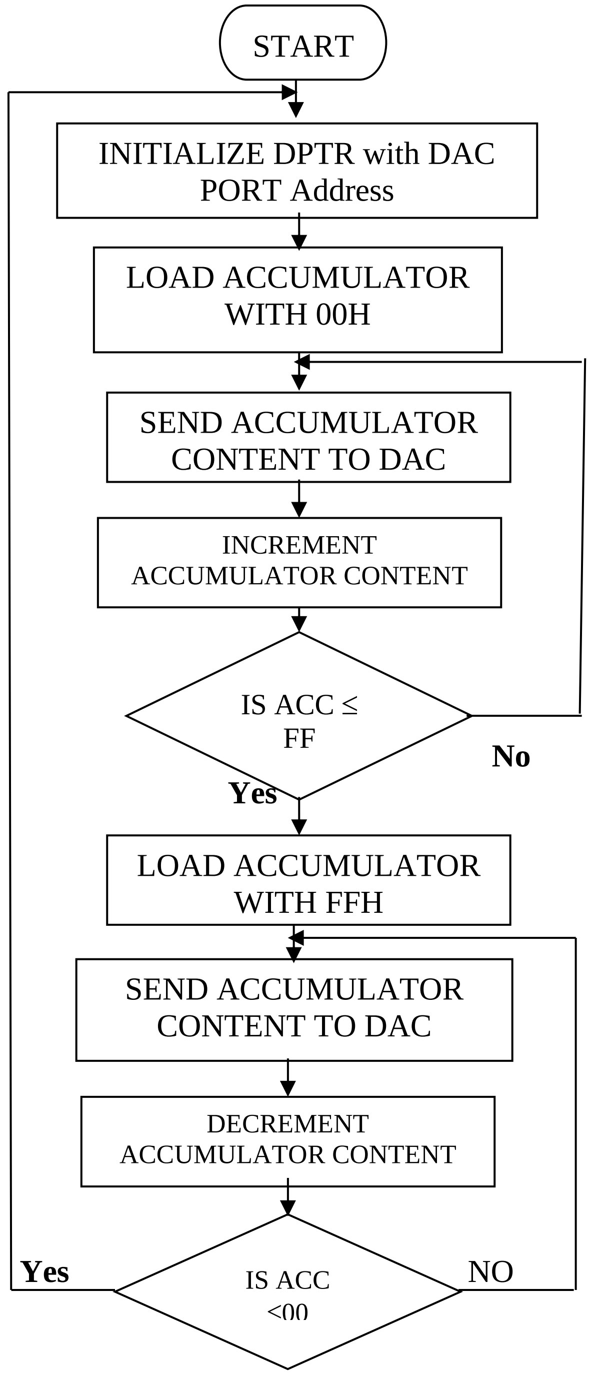
SAWTOOTH WAVE GENERATION:

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

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 step 3.

TRIANGULAR WAVEFORM



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 step2. Else go to step 7.

PROGRAM:

(A) Square Wave Generation

Address	Label	Mnemonics	Opcode	Comments
		ORG 4100H		
		MOV DPTR,PORT		MOV DPTR,PORT ADDRESS OF DAC
4100	START	MOV A,#00		Clear Accumulator
4102		MOVX @DPTR,A		Move A → DPTR
4103		LCALL DELAY		Call delay
4104		MOV A,#FF		Load FF → A
4106		MOVX @DPTR,A		Move A → DPTR
4107		LCALL DELAY		Call delay
410A		LJUMP START		Jump to start
410D	DELAY:	MOV R1,#05		Delay loop
410F	LOOP:	MOV R2,#FF		
4111	HERE:	DJNZ R2,HERE		
4114		DJNZ R1,LOOP		
4117		RET		Return and jump to start
4118		SJMP START		

(B) Saw tooth Wave Generation

Address	Label	Mnemonics	Opcode	Comments
		ORG 4100H		
		MOV DPTR,PORT		MOV DPTR,PORT ADDRESS OF DAC
4100	START	MOV A,#00		Clear Accumulator
4103	LOOP	MOVX @DPTR,A		Move A → DPTR
4105		INC A		Increment A
		SJMP LOOP		Jump to location loop

(C) Triangular Wave Generation

Address	Label	Mnemonics	Opcode	Comments
		ORG 4100H		
		MOV DPTR,PORT		MOV DPTR,PORT ADDRESS OF DAC
4100	START	MOV A,#00		Clear Accumulator
4102	LOOP1	MOVX @DPTR,A		Move A → DPTR
4103		INC A		Increment A
4104		JNZ LOOP1		Jump not zero to location loop1
4107		MOV A,#FF		Load FF → A
4109	LOOP2:	MOVX @DPTR,A		Move A → DPTR
410A		DEC A		Decrement A
410B		JNZ LOOP2		Jump not zero to location loop2
411E		LJMP START		Delay loop

RESULT:

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

VIVA QUESTIONS:

1. Briefly give the principle behind the triangular wave generation
2. What is settling or conversion time in DAC?
3. What are the internal devices of a typical DAC?.
4. What are Program and data memory size in 8051
5. How many 16 bit timers are available in 8051?
6. What is meant by SBUF?

7C . TRAFFIC LIGHT CONTROLLER - INTERFACING WITH 8051

AIM:

To design traffic light controller using 8051 microcontroller

APPARATUS REQUIRED:

8051 kit, DC regulated power supply, Traffic light controller interface board.

PROGRAM:

ADDRESS	OPCODES	LABEL	MNEMONICS	OPERAND	COMMENTS
			ORG	0000h	
			MOV P0	#0D4H	
			ACALL	DELAY1	
			MOV P0	#53H	
			ACALL	DELAY1	
			MOV P0	#04DH	
			ACALL	DELAY1	
			MOV P0	#35H	
			ACALL	DELAY1	
	DELAY1		MOV R2	#42D	
			MOV R1	#40D	
			MOV R0	#30D	
	LOOP1		DJNZ	R0,LOOP1	
	LOOP2		DJNZ	R1,LOOP2	
	LOOP3		DJNZ	R2,LOOP3	
			RET		

ALGORITHM:-

1. Start.
2. Write the data for each direction and load it into the P0
3. Initialize a counter to indicate the number of directions.
4. call the delay program and repeat the process.

RESULT:

Thus the design of traffic light controller using 8085 microprocessor through programmable peripheral interface 8255is done and also the output is verified.

8 STEPPER MOTOR INTERFACING WITH 8051

AIM:

To operate stepper motor by interfacing with 8051 microcontroller.

THEORY:

A motor in which the rotor is able to assume only discrete stationary angular position is a stepper motor. The rotary motion occurs in a step-wise manner from one equilibrium position to the next. Stepper Motors are used very wisely in position control systems like printers, disk drives, process control machine tools, etc.

The basic two-phase stepper motor consists of two pairs of stator poles. Each of the four poles has its own winding. The excitation of any one winding generates a North Pole. A South Pole gets induced at the diametrically opposite side. The rotor magnetic system has two end faces. It is a permanent magnet with one face as South Pole and the other as North Pole.

The Stepper Motor windings A1, A2, B1, B2 are cyclically excited with a DC current to run the motor in clockwise direction. By reversing the phase sequence as A1, B2, A2, B1, anticlockwise stepping can be obtained.

2-PHASE SWITCHING SCHEME:

In this scheme, any two adjacent stator windings are energized. The switching scheme is shown in the table given below. This scheme produces more torque.

ANTICLOCKWISE						**CLOCKWISE**					
STEP	**A1**	**A2**	**B1**	**B2**	**DATA**	**STEP**	**A1**	**A2**	**B1**	**B2**	**DATA**
1	1	0	0	1	9h	1	1	0	1	0	Ah
2	0	1	0	1	5h	2	0	1	1	0	6h
3	0	1	1	0	6h	3	0	1	0	1	5h
4	1	0	1	0	Ah	4	1	0	0	1	9h

ADDRESS DECODING LOGIC:

The 74138 chip is used for generating the address decoding logic to generate the device select pulses, CS1 & CS2 for selecting the IC 74175.The 74175 latches the data bus to the stepper motor driving circuitry.

Stepper Motor requires logic signals of relatively high power. Therefore, the interface circuitry that generates the driving pulses use silicon darlington pair transistors. The inputs for the interface circuit are TTL pulses generated under software control using the Microcontroller Kit. The TTL levels of pulse sequence from the data bus is translated to high voltage output pulses using a buffer 7407 with open collector.

PROGRAM :

Address	Label	Mnemonics	Operand	Comments
		ORG	4100h	
4100	START:	MOV	DPTR, #TABLE	Load the start address of switching scheme data TABLE into Data Pointer (DPTR)
4103		MOV	R0, #04	Load the count in R0
4105	LOOP:	MOVX	A, @DPTR	Load the number in TABLE into A
4106		PUSH	DPH	Push DPTR value to Stack
4108		PUSH	DPL	
410A		MOV	DPTR, #0FFC0h	Load the Motor port address into DPTR
410D		MOVX	@DPTR, A	Send the value in A to stepper Motor port address
410E		MOV	R4, #0FFh	Delay loop to cause a specific amount of time delay before next data item is sent to the Motor
4110	DELAY:	MOV	R5, #0FFh	
4112	DELAY1:	DJNZ	R5, DELAY1	
4114		DJNZ	R4, DELAY	
4116		POP	DPL	POP back DPTR value from Stack
4118		POP	DPH	
411A		INC	DPTR	Increment DPTR to point to next item in the table
411B		DJNZ	R0, LOOP	Decrement R0, if not zero repeat the loop
411D		SJMP	START	Short jump to Start of the program to make the motor rotate continuously
411F	TABLE:	DB	09 05 06 0Ah	Values as per two-phase switching scheme

PROCEDURE:

- Enter the above program starting from location 4100 and execute the same. The stepper motor rotates.
- By varying the count at R4 and R5 can vary the speed.
- By entering the data in the look-up TABLE in the reverse order can vary direction of rotation.

RESULT:

Thus a stepper motor was interfaced with 8051 and run in forward and reverse directions at various speeds.

VIVA QUESTIONS:

1. What are the application of stepper motor?
2. What is meant by step angle?
3. What are the methods to control the speed of stepper motor?
4. What is the formula for steps per revolution?
5. What is the use of DB instruction?
6. What is the use of PUSH and POP operation ?
7. How a stepper motor differs from DC motor?

Ex,No: **9. STUDY OF BASIC DIGITAL ICs AND FLIPFLOPS**

Date:

AIM:

To verify the truth table of basic digital ICs of AND, OR, NOT, NAND, NOR, EX-OR gates.

APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit		1
2.	AND gate	IC 7408	1
3.	OR gate	IC 7432	1
4.	NOT gate	IC 7404	1
5.	NAND gate	IC 7400	1
6.	NOR gate	IC 7402	1
7.	EX-OR gate	IC 7486	1
8.	Connecting wires	As required	

THEORY:

a. AND gate:

An AND gate is the physical realization of logical multiplication operation. It is an electronic circuit which generates an output signal of '1' only if all the input signals are '1'.

b. OR gate:

An OR gate is the physical realization of the logical addition operation. It is an electronic circuit which generates an output signal of '1' if any of the input signal is '1'.

c. NOT gate:

A NOT gate is the physical realization of the complementation operation. It is an electronic circuit which generates an output signal which is the reverse of the input signal. A NOT gate is also known as an inverter because it inverts the input.

d. NAND gate:

A NAND gate is a complemented AND gate. The output of the NAND gate will be '0' if all the input signals are '1' and will be '1' if any one of the input signal is '0'.

e. NOR gate:

A NOR gate is a complemented OR gate. The output of the OR gate will be '1' if all the inputs are ‘0’ and will be ‘0’ if any one of the input signal is ‘1’.

f. EX-OR gate:

An Ex-OR gate performs the following Boolean function,

$$A \oplus B = (A B') + (A'B)$$

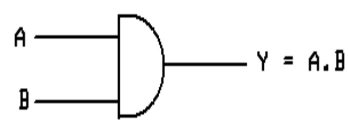
It is similar to OR gate but excludes the combination of both A and B being equal to one. The exclusive OR is a function that give an output signal ‘0’ when the two input signals are equal either ‘0’ or ‘1’.

PROCEDURE:

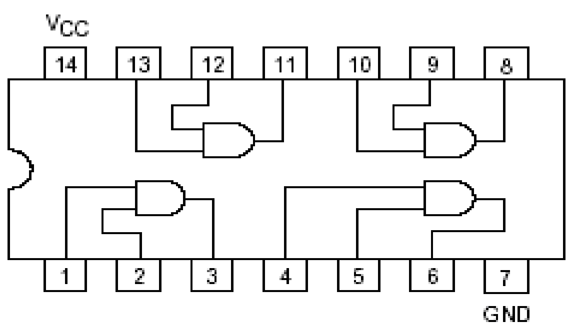
1. Connections are given as per the circuit diagram
1. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
2. Apply the inputs and verify the truth table for all gates.

ANDGATE

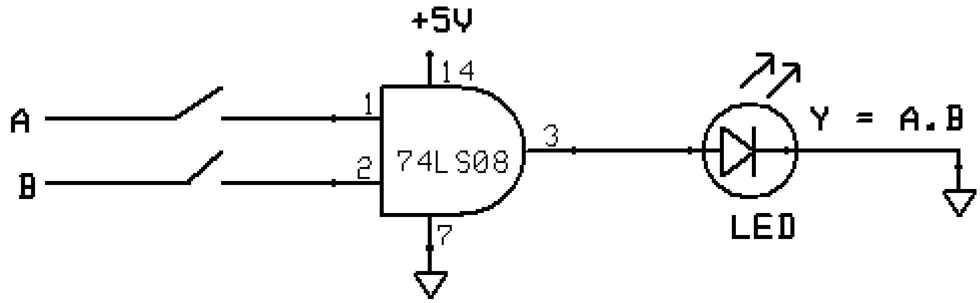
LOGIC DIAGRAM:



PIN DIAGRAM OF IC 7408:



CIRCUIT DIAGRAM:

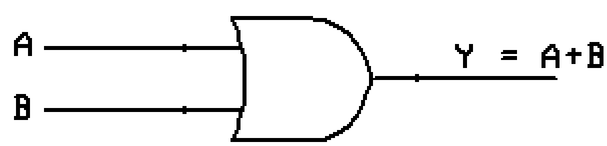


TRUTH TABLE:

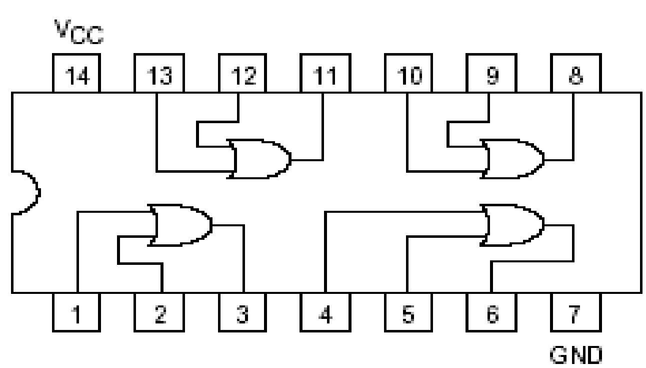
S.No	INPUT		OUTPUT
	A	B	Y = A. B
1.	0	0	0
2.	0	1	0
3.	1	0	0
4.	1	1	1

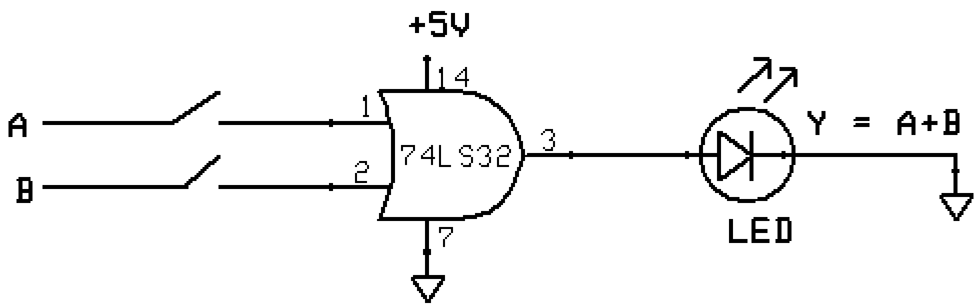
OR GATE

LOGIC DIAGRAM:



PIN DIAGRAM OF IC 7432 :



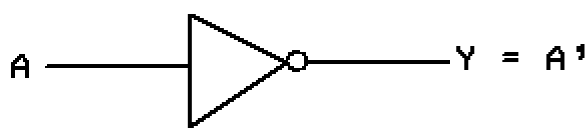


TRUTH TABLE:

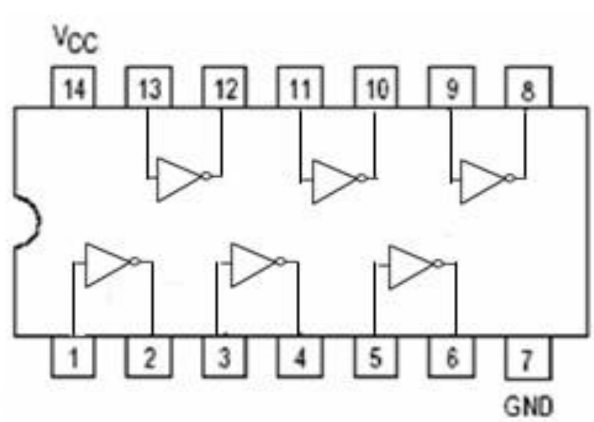
S.No	INPUT		OUTPUT
	A	B	Y = A + B
1.	0	0	0
2.	0	1	1
3.	1	0	1
4.	1	1	1

NOT GATE

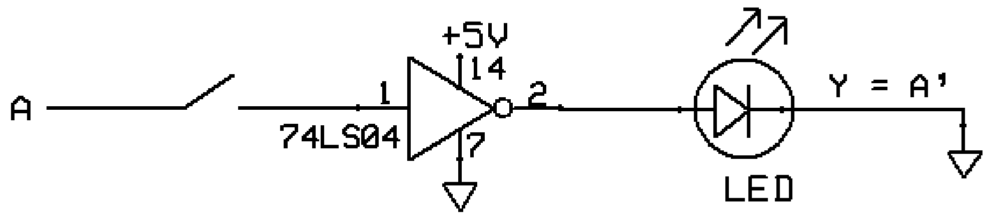
LOGIC DIAGRAM:



PIN DIAGRAM OF IC 7404 :



CIRCUIT DIAGRAM

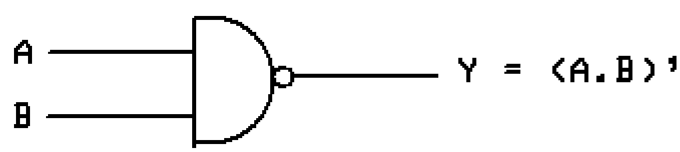


TRUTH TABLE:

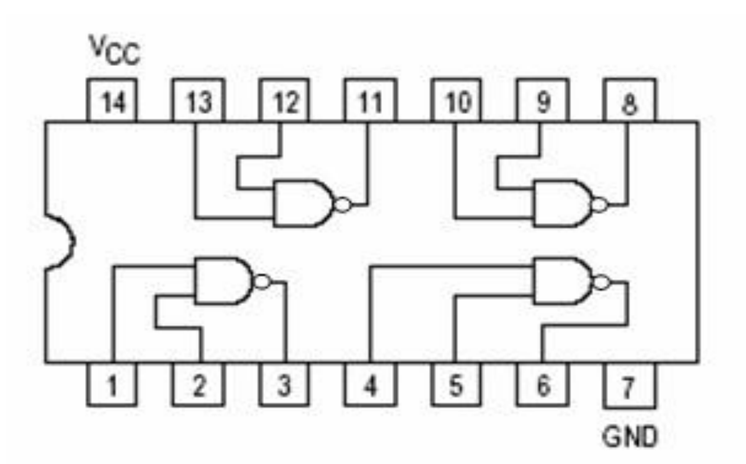
S.No	INPUT	OUTPUT
	A	Y = A'
1.	0	1
2.	1	0

<u>NANDGATE</u>

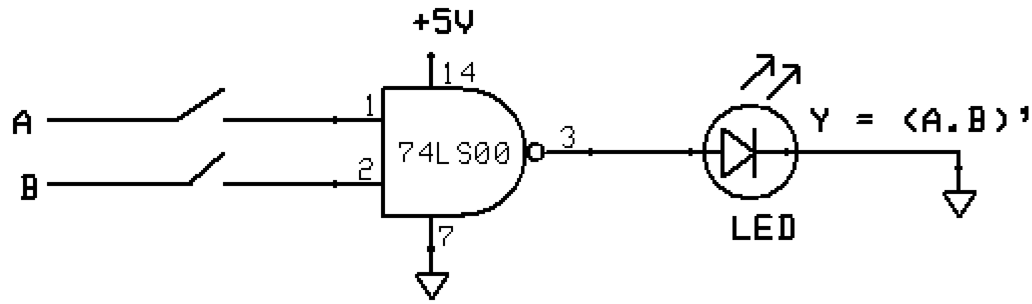
LOGIC DIAGRAM:



PIN DIAGRAM OF IC 7400 :



CIRCUIT DIARAM:

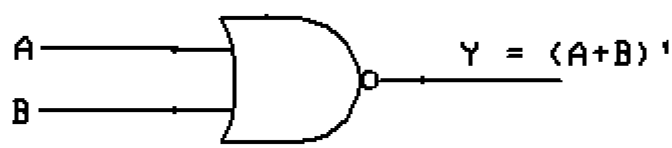


TRUTH TABLE:

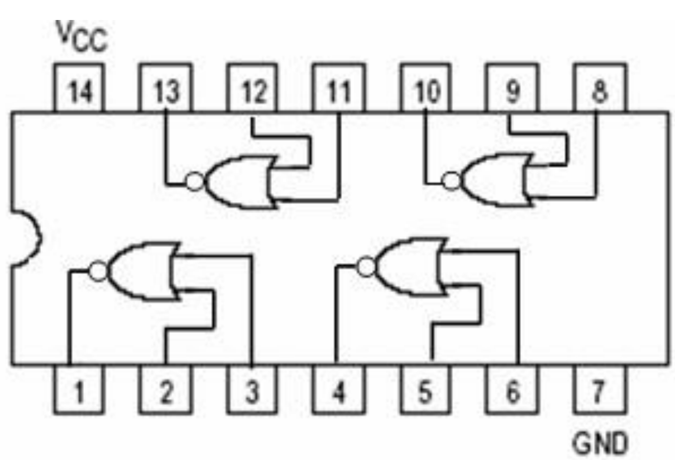
S.No	INPUT		OUTPUT
	A	B	Y = (A. B)’
1.	0	0	1
2.	0	1	1
3.	1	0	1
4.	1	1	0

<u>NORGATE</u>

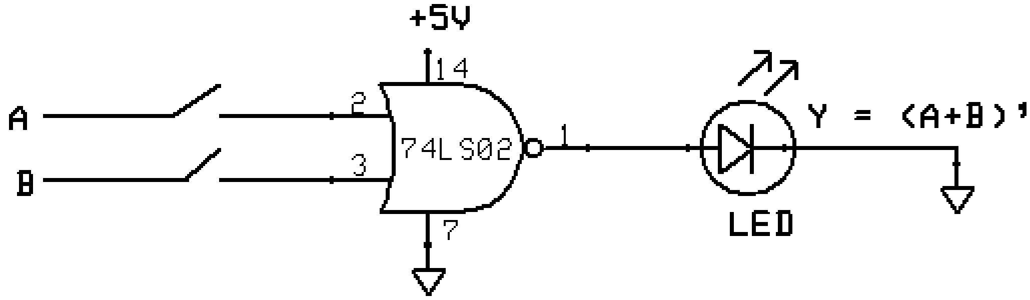
LOGIC DIAGRAM:



PIN DIAGRAM OF IC 7402 :



CIRCUIT DIAGRAM:

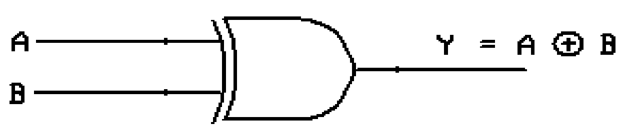


TRUTH TABLE:

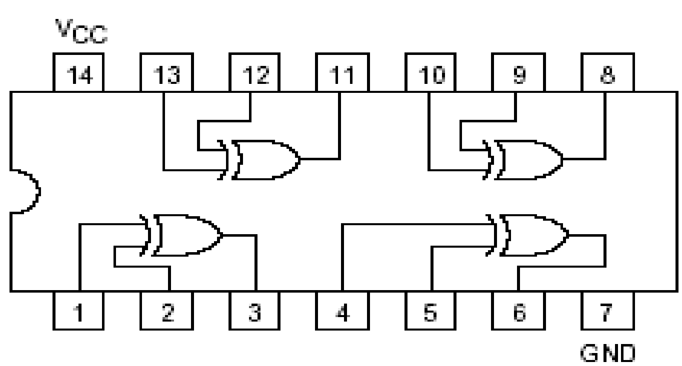
S.No	INPUT		OUTPUT
	A	B	Y = (A + B)’
1.	0	0	1
2.	0	1	0
3.	1	0	0
4.	1	1	0

EX-ORGATE

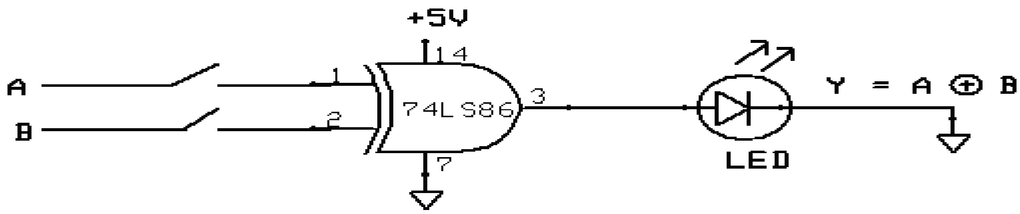
LOGIC DIAGRAM



PIN DIAGRAM OF IC 7486:



CIRCUIT DIAGRAM:



TRUTH TABLE:

S.No	INPUT		OUTPUT
	A	B	Y = A ⊕ B
1.	0	0	0
2.	0	1	1
3.	1	0	1
4.	1	1	0

DISCUSSION QUESTIONS:

1. What is Integrated Circuit?
2. What is a Logic gate?
3. What are the basic digital logic gates?
4. What are the gates called universal gates?
5. Why NAND and NOR gates are called universal gates?
6. What are the properties of EX-NOR gate?

RESULT:

The truth tables of all the basic digital ICs were verified.

:

10 STUDY OF FLIP FLOPS

Date:

AIM:

To verify the characteristic table of RS, D, JK, and T Flip flops .

APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit		1
2.	NOR gate	IC 7402	
3.	NOT gate	IC 7404	
4.	AND gate (three input)	IC 7411	
5.	NAND gate	IC 7400	
6.	Connecting wires		As required

THEORY:

A Flip Flop is a sequential device that samples its input signals and changes its output states only at times determined by clocking signal. Flip Flops may vary in the number of inputs they possess and the manner in which the inputs affect the binary states.

RS FLIP FLOP:

The clocked RS flip flop consists of NAND gates and the output changes its state with respect to the input on application of clock pulse. When the clock pulse is high the S and R inputs reach the second level NAND gates in their complementary form. The Flip Flop is reset when the R input high and S input is low. The Flip Flop is set when the S input is high and R input is low. When both the inputs are high the output is in an indeterminate state.

D FLIP FLOP:

To eliminate the undesirable condition of indeterminate state in the SR Flip Flop when both inputs are high at the same time, in the D Flip Flop the inputs are never made equal at the same time. This is obtained by making the two inputs complement of each other.

JK FLIP FLOP:

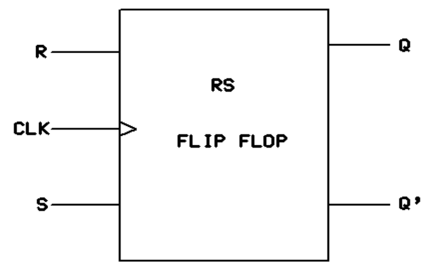
The indeterminate state in the SR Flip-Flop is defined in the JK Flip Flop. JK inputs behave like S and R inputs to set and reset the Flip Flop. The output Q is ANDed with K input and the clock pulse, similarly the output Q’ is ANDed with J input and the Clock pulse. When the clock pulse is zero both the AND gates are disabled and the Q and Q’ output retain their previous values. When the clock pulse is high, the J and K inputs reach the NOR gates. When both the inputs are high the output toggles continuously. This is called Race around condition and this must be avoided.

T FLIP FLOP:

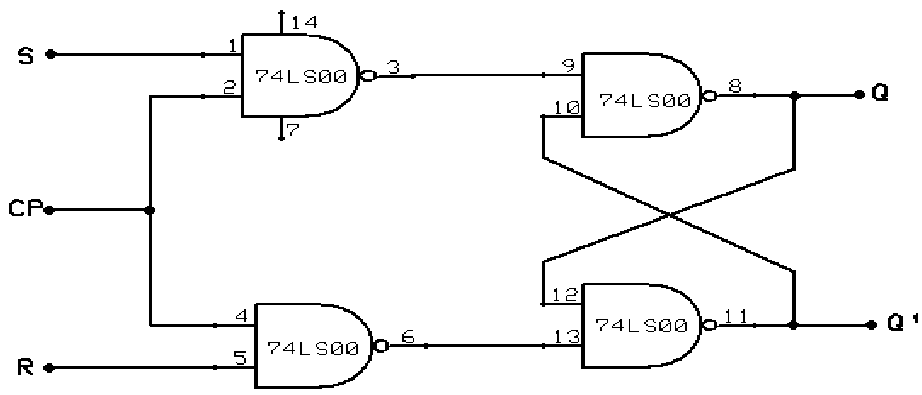
This is a modification of JK Flip Flop, obtained by connecting both inputs J and K inputs together. T Flip Flop is also called Toggle Flip Flop.

RS FLIP FLOP

LOGIC SYMBOL:



CIRCUIT DIAGRAM:

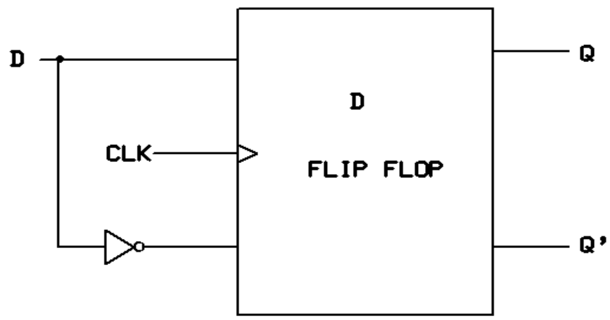


CHARACTERISTIC TABLE:

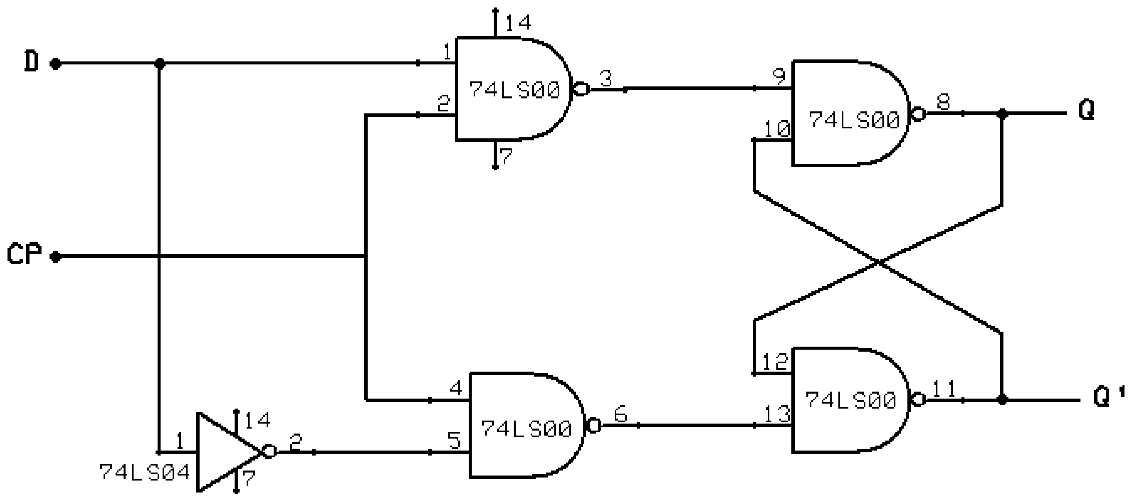
CLOCK PULSE	INPUT		PRESENT STATE (Q)	NEXT STATE(Q+1)	STATUS
	S	R			
1	0	0	0	0	
2	0	0	1	1	
3	0	1	0	0	
4	0	1	1	0	
5	1	0	0	1	
6	1	0	1	1	
7	1	1	0	X	
8	1	1	1	X	

D FLIP FLOP

LOGIC SYMBOL:



CIRCUIT DIAGRAM:

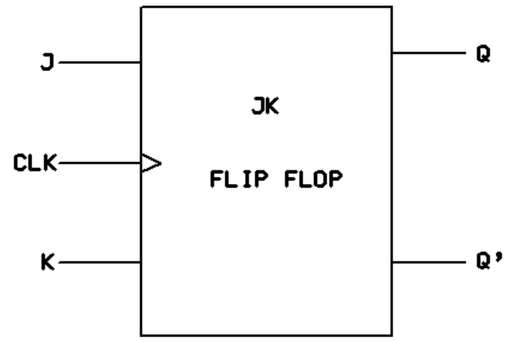


CHARACTERISTIC TABLE:

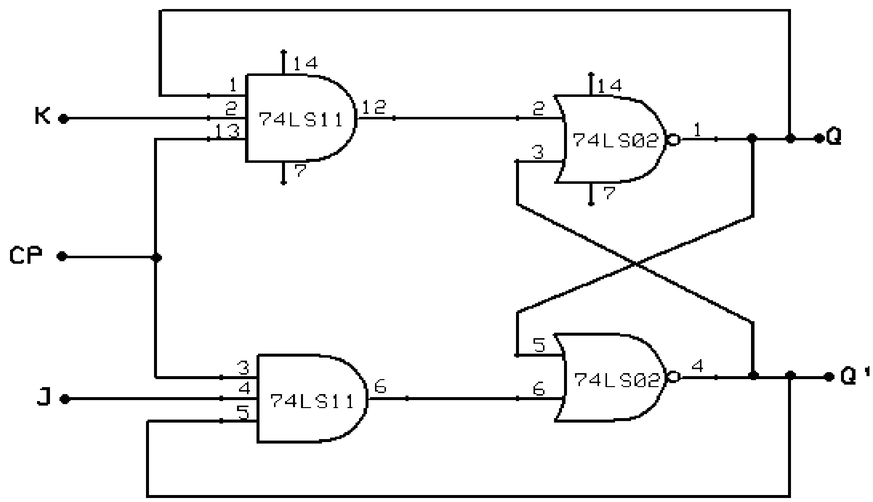
CLOCK PULSE	INPUT D	PRESENT STATE (Q)	NEXT STATE(Q+1)	STATUS
1	0	0	0	
2	0	1	0	
3	1	0	1	
4	1	1	1	

JK FLIP FLOP

LOGIC SYMBOL:



CIRCUIT DIAGRAM:

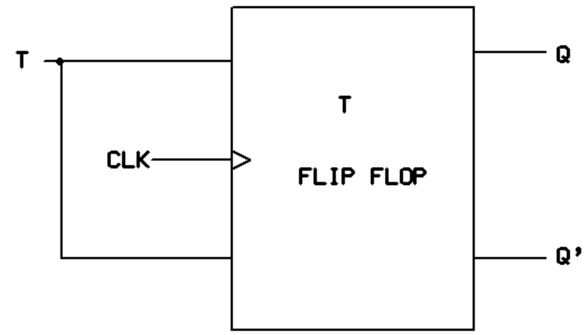


CHARACTERISTIC TABLE:

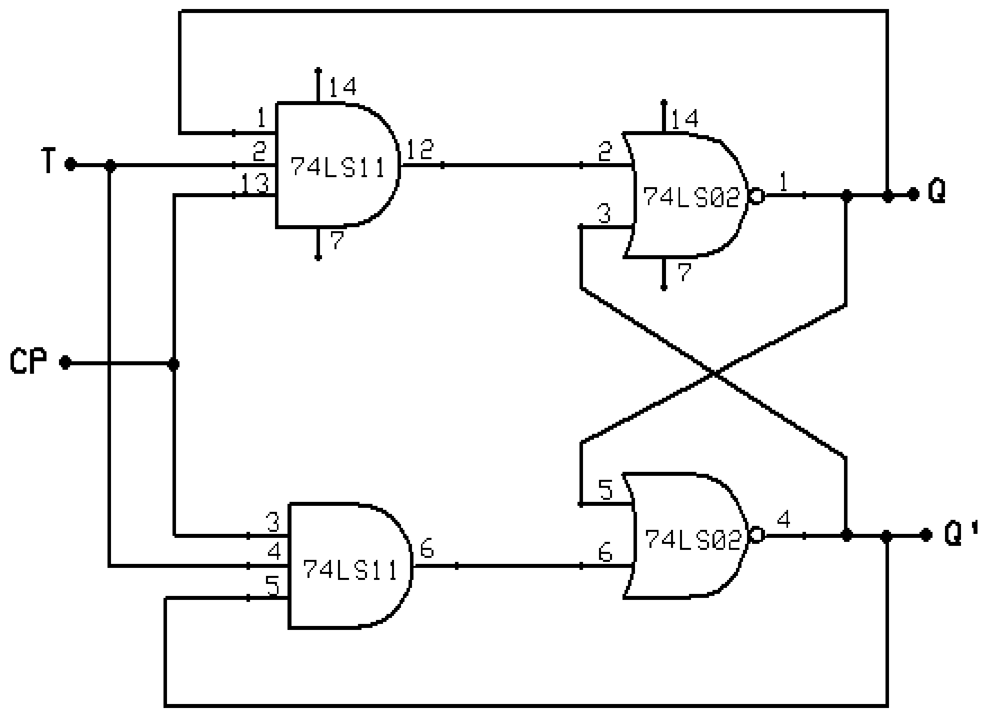
CLOCK PULSE	INPUT		PRESENT STATE (Q)	NEXT STATE(Q+1)	STATUS
	J	K			
1	0	0	0	0	
2	0	0	1	1	
3	0	1	0	0	
4	0	1	1	0	
5	1	0	0	1	
6	1	0	1	1	
7	1	1	0	1	
8	1	1	1	0	

T FLIPFLOP

LOGIC SYMBOL:



CIRCUIT DIAGRAM:



CHARACTERISTIC TABLE:

CLOCK PULSE	INPUT T	PRESENT STATE (Q)	NEXT STATE(Q+1)	STATUS
1	0	0	0	
2	0	1	0	
3	1	0	1	
4	1	1	0	

PROCEDURE:

1. Connections are given as per the circuit diagrams.
2. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
3. Apply the inputs and observe the status of all the flip flops.

DISCUSSION QUESTIONS:

1. Define flip-flop
2. What is race around condition?
3. Explain the flip-flop excitation tables for D flip-flop
4. Explain the flip-flop excitation tables for JK flip-flop
5. What is a master-slave flip-flop?
6. What is edge-triggered flip-flop?
7. What is the operation of D flip-flop?
8. What are the different types of flip-flop?

RESULT:

The Characteristic tables of RS, D, JK, T flip flops were verified.

Ex.No:

10A IMPLEMENTATION OF BOOLEAN FUNCTIONS

Date:

AIM:

To design the logic circuit and verify the truth table of the given Boolean expression, F (A,B,C,D) = Σ (0,1,2,5,8,9,10)

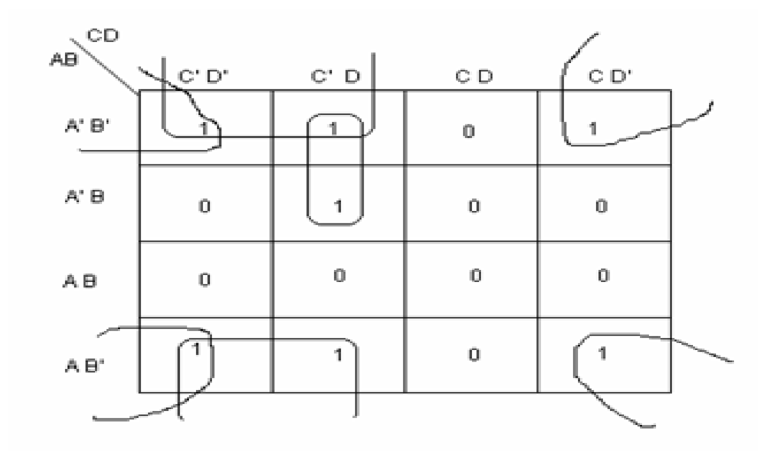
APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit		1
2.	AND gate	IC 7408	
3.	OR gate	IC 7432	
4.	NOT gate	IC 7404	
5.	NAND gate	IC 7400	
6.	NOR gate	IC 7402	
7.	EX-OR gate	IC 7486	
8.	Connecting wires		As required

DESIGN:

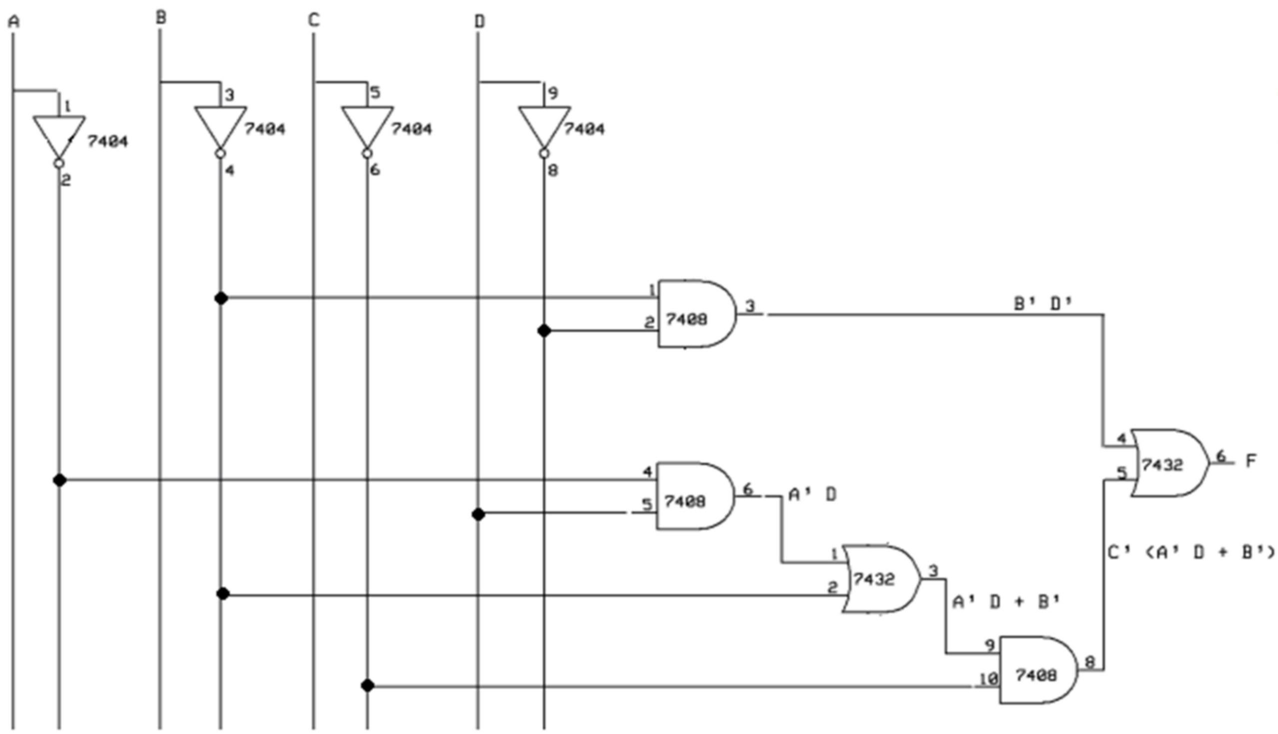
Given , F (A,B,C,D) = Σ (0,1,2,5,8,9,10)

The output function F has four input variables hence a four variable Karnaugh Map is used to obtain a simplified expression for the output as shown,



From the K-Map,
F = B' C' + D' B' + A' C' D
Since we are using only two input logic gates the above expression can be re-written as, F = C' (B' + A' D) + D' B'
Now the logic circuit for the above equation can be drawn.

CIRCUIT DIAGRAM:



TRUTH TABLE:

S.No	INPUT				OUTPUT
	A	B	C	D	F=D'B'+C'(B'+A'D)
1.	0	0	0	0	1
2.	0	0	0	1	1
3.	0	0	1	0	1
4.	0	0	1	1	0
5.	0	1	0	0	0
6.	0	1	0	1	1
7.	0	1	1	0	0
8.	0	1	1	1	0
9.	1	0	0	0	1
10.	1	0	0	1	1
11.	1	0	1	0	1
12.	1	0	1	1	0
13.	1	1	0	0	0
14.	1	1	0	1	0
15.	1	1	1	0	0
16.	1	1	1	1	0

PROCEDURE:

1. Connections are given as per the circuit diagram
2. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
3. Apply the inputs and verify the truth table for the given Boolean expression.

DISCUSSION QUESTIONS:

1. What is variable mapping?
2. Define Demorgans theorem.
3. What do you mean by don’t care functions?
4. State two absorption properties of Boolean function.
5. What are the two methods of Boolean function minimization?

RESULT:

The truth table of the given Boolean expression was verified.

Ex.No:

10B DESIGN AND IMPLEMENTATION OF ADDER/SUBTRACTOR

Date:

AIM:

To design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using logic gates.

APPARATUS REQUIRED:

S. No	Name	Specification	Quantity
1.	IC	7432, 7408, 7486, 7483	1
2.	Digital IC Trainer Kit		1
3.	Patch chords		-

THEORY:

The most basic arithmetic operation is the addition of two binary digits. There are four possible elementary operations, namely,

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10_2$$

The first three operations produce a sum of whose length is one digit, but when the last operation is performed the sum is two digits. The higher significant bit of this result is called a carry and lower significant bit is called the sum.

HALFADDER:

A combinational circuit which performs the addition of two bits is called half adder. The input variables designate the augend and the addend bit, whereas the output variables produce the sum and carry bits.

FULLADDER:

A combinational circuit which performs the arithmetic sum of three input bits is called full adder. The three input bits include two significant bits and a previous carry bit. A full adder circuit can be implemented with two half adders and one OR gate.

HALFADDER

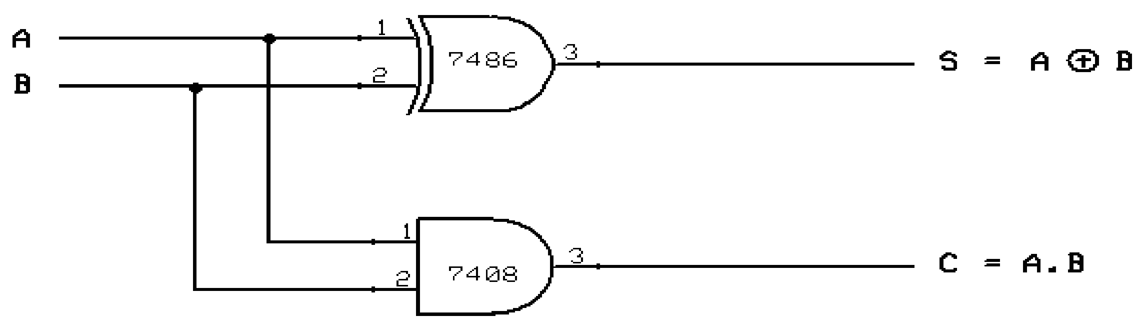
TRUTH TABLE:

S.No	INPUT		OUTPUT	
	A	B	S	C
1.	0	0	0	0
2.	0	1	1	0
3.	1	0	1	0
4.	1	1	0	1

DESIGN:

From the truth table the expression for sum and carry bits of the output can be obtained as, Sum, $S = A \oplus B$; Carry, $C = A . B$

CIRCUIT DIAGRAM:



FULLADDER

TRUTH TABLE:

S.No	INPUT			OUTPUT	
	A	B	C	SUM	CARRY
1.	0	0	0	0	0
2.	0	0	1	1	0
3.	0	1	0	1	0
4.	0	1	1	0	1
5.	1	0	0	1	0
6.	1	0	1	0	1
7.	1	1	0	0	1
8.	1	1	1	1	1

DESIGN:

From the truth table the expression for sum and carry bits of the output can be obtained as,SUM = A’B’C + A’BC’ + AB’C’ + ABC;CARRY = A’BC + AB’C + ABC’ +ABC
Using Karnaugh maps the reduced expression for the output bits can be obtained as,

<u>SUM</u>

A \ BC	B’C’	B’C	BC	BC’
A’	0	1	0	1
A	1	0	1	0

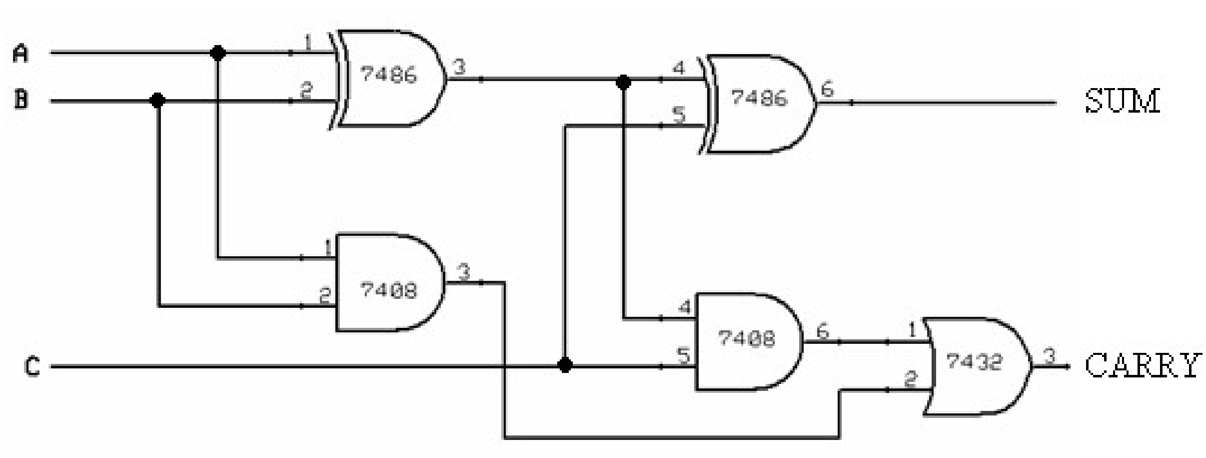
SUM = A’B’C + A’BC’ + AB’C’ + ABC = A $\oplus$ B $\oplus$ C

<u>CARRY</u>

A \ BC	B’C’	B’C	BC	BC’
A’	0	0	1	0
A	0	1	1	1

CARRY = AB + AC + BC

CIRCUIT DIAGRAM:



HALFSUBTRACTOR:

A combinational circuit which performs the subtraction of two bits is called half subtractor. The input variables designate the minuend and the subtrahend bit, whereas the output variables produce the difference and borrow bits.

FULLSUBTRACTOR:

A combinational circuit which performs the subtraction of three input bits is called full subtractor. The three input bits include two significant bits and a previous borrow bit. A full subtractor circuit can be implemented with two half subtractors and one OR gate.

HALFSUBTRACTOR

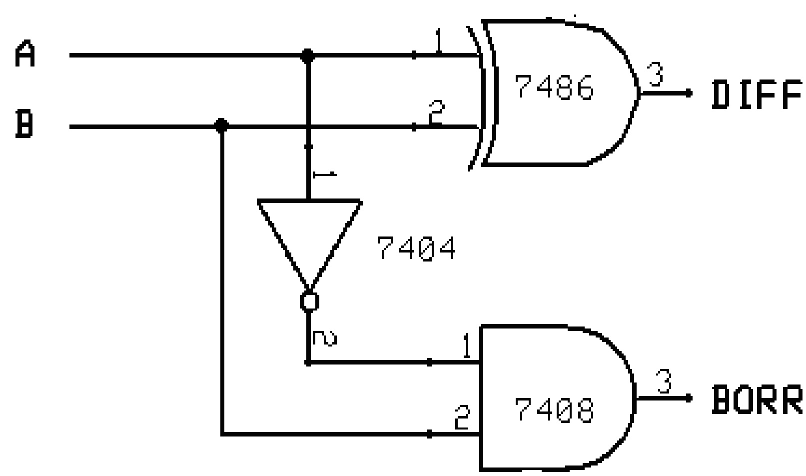
TRUTH TABLE:

S.No	INPUT		OUTPUT	
	A	B	DIFF	BORR
1.	0	0	0	0
2.	0	1	1	1
3.	1	0	1	0
4.	1	1	0	0

DESIGN:

From the truth table the expression for difference and borrow bits of the output can be obtained as, Difference, DIFF = A $\oplus$ B; Borrow, BORR = A' . B

CIRCUIT DIAGRAM:



FULLSUBTRACTOR

TRUTH TABLE:

S.No	INPUT			OUTPUT	
	A	B	C	DIFF	BORR
1.	0	0	0	0	0
2.	0	0	1	1	1
3.	0	1	0	1	1
4.	0	1	1	0	1
5.	1	0	0	1	0
6.	1	0	1	0	0
7.	1	1	0	0	0
8.	1	1	1	1	1

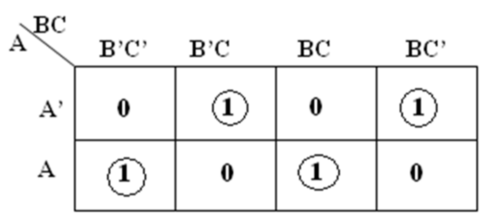
DESIGN:

From the truth table the expression for difference and borrow bits of the output can be obtained as,

Difference, DIFF= A’B’C + A’BC’ + AB’C’ + ABC
Borrow, BORR = A’BC + AB’C + ABC’ +ABC

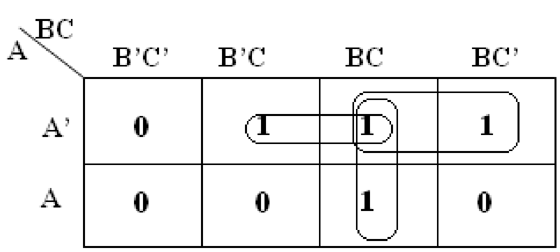
Using Karnaugh maps the reduced expression for the output bits can be obtained as,

DIFFERENCE



A’B’C + A’BC’ + AB’C’ + ABC = A $\oplus$B $\oplus$ C

BORROW

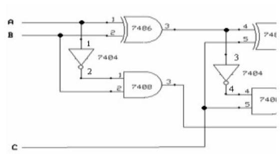


BORROW = A’B + A’C + BC

CIRCUIT DIAGRAM:

PROCEDURE:

- The connections are given as pe



r the circuit diagram.

- Two 4 – bit numbers added or subtracted depend upon the control input and the output is obtained.
- Apply the inputs and verify the truth table for thehalf adder or s subtractor and full adder or subtractor circuits.

DISCUSSION QUESTIONS:

1. What is a combinational circuit?
2. What is different between combinational and sequential circuit?
3. What are the gates involved for binary adder?
4. List the properties of Ex-Nor gate?
5. What is the expression for sum and carry in half and full adder?

RESULT:

Thus the half adder, full adder, half subtractor and full subtractor circuits were designed and their truth table were verified.

Ex. No: Date:

11 CODE CONVERTER

AIM:

To construct and verify the performance of binary to gray and gray to binary.

APPARATUS REQUIRED:

S. No	Name	Specification	Quantity
1.	IC	7404, 7486	1
2.	Digital IC Trainer Kit		1
3.	Patch chords		-

THEORY:

BINARY TO GRAY:

The MSB of the binary code alone remains unchanged in the Gray code. The remaining bits in the gray are obtained by EX-OR ing the corresponding gray code bit and previous bit in the binary code. The gray code is often used in digital systems because it has the advantage that only one bit in the numerical representation changes between successive numbers.

GRAY TO BINARY**:**

The MSB of the Gray code remains unchanged in the binary code the remaining bits are obtained by EX – OR ing the corresponding gray code bit and the previous output binary bit.

PROCEDURE:

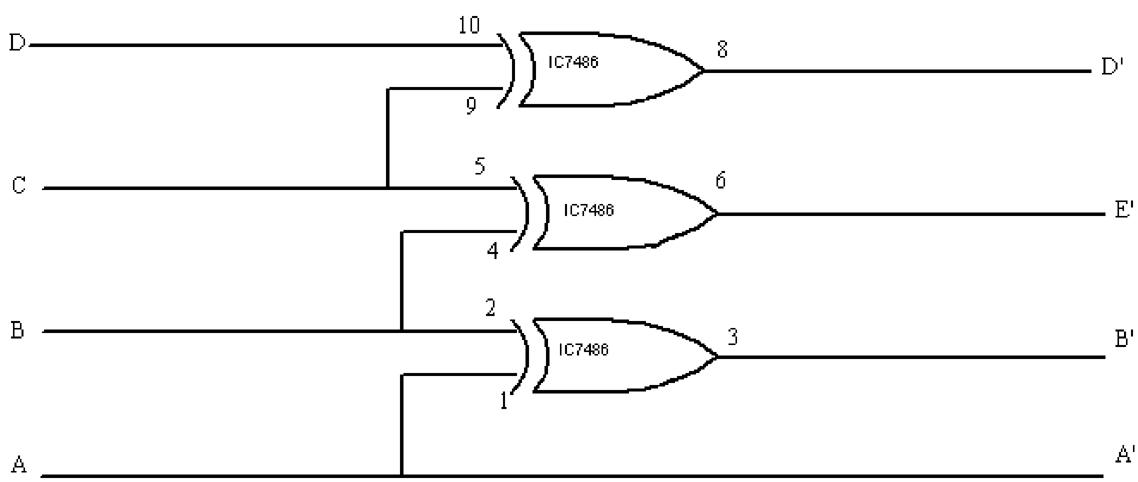
- Connections are given as per the logic diagram.
- The given truth tables are verified.

BINARY TOGRAY:

TRUTH TABLE

Decimal	Binary code				Gray code			
	D	C	B	A	**D’**	C’	B’	A’
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Logic diagram

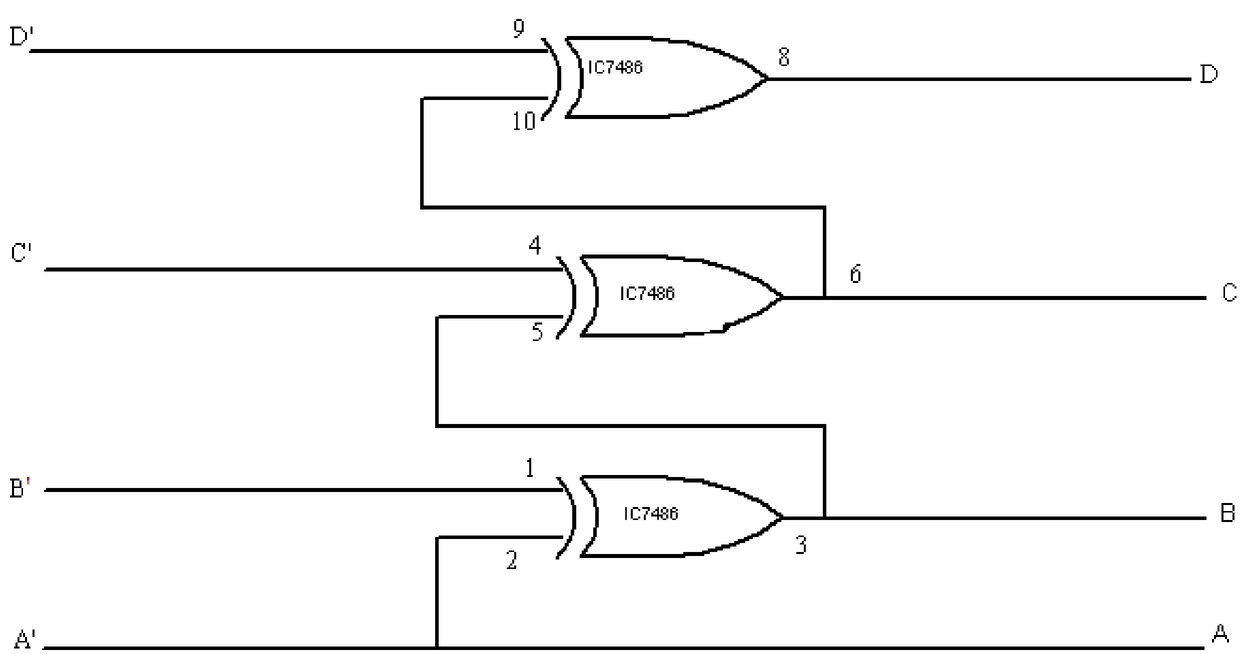


GRAY TO BINARY

TRUTH TABLE

Decimal	Binary code				Gray code			
	D’	C’	B’	A’	**D**	C	B	A
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Logic diagram

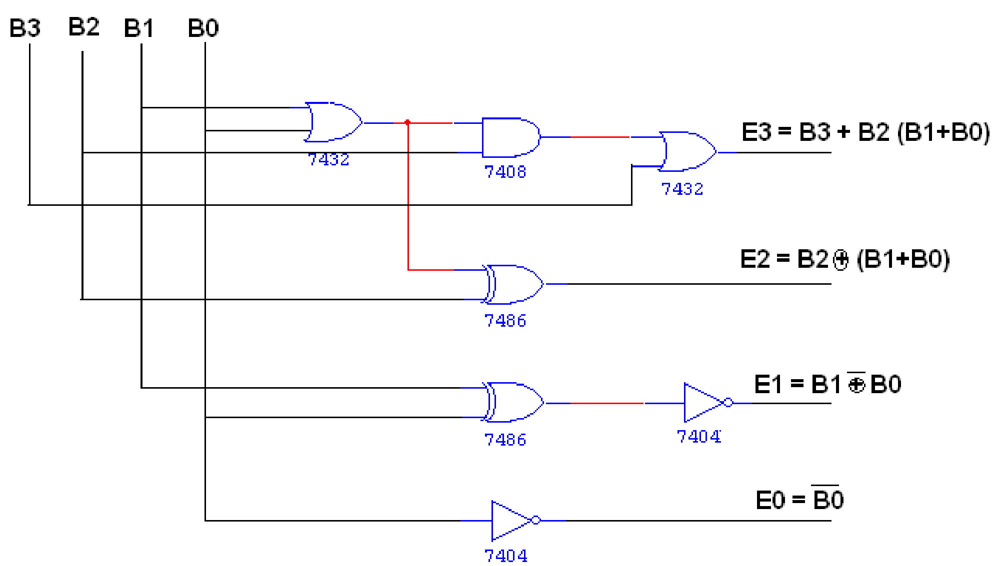


BCD TO EXCESS-3

TRUTH TABLE

BCD Input				Excess-3 Output			
B3	B2	B1	B0	E3	E2	E1	E0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Logic diagram

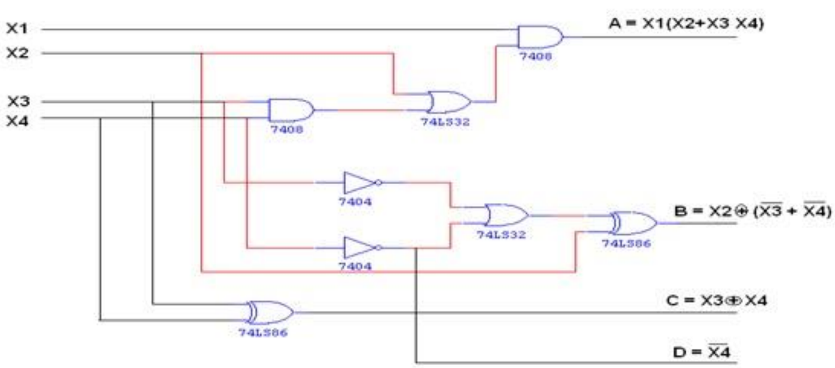


EXCESS-3 TO BCD

TRUTH TABLE

Excess-3 Input				BCD Output			
E3	E2	E1	E0	B3	B2	B1	B0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Logic diagram



DISCUSSION QUESTIONS:

1. List the procedures to convert gray code into binary?
2. Why weighted code is called as reflective codes?
3. What is a sequential code?
4. What is error deducting code?
5. What is ASCII code?

RESULT:

The design of the three bit Binary to Gray code converter & Gray to Binary code converter circuits was done and its truth table was verified.

Ex. No: **1) ENCODER**

Date:

AIM:

To design and implement encoder using IC 74148 (8-3 encoder)

APPARATUS REQUIRED:

S. No	Name	Specification	Quantity
1.	IC	74148	1
2.	Digital IC Trainer Kit		1
3.	Patch chords		-

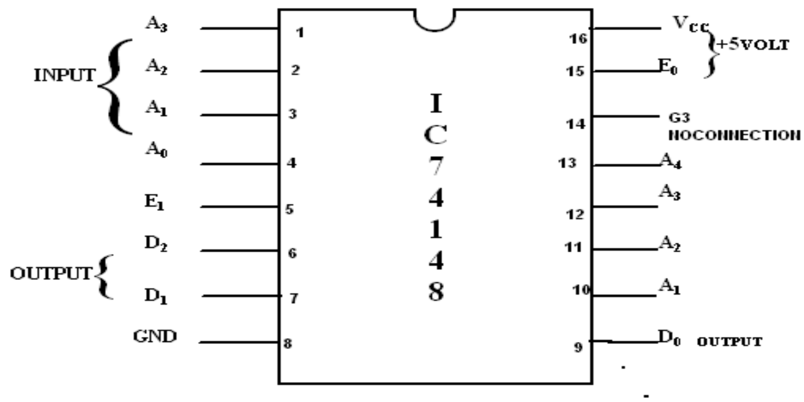
THEORY:

An encoder is digital circuit that has 2^n input lines and n output lines. The output lines generate a binary code corresponding to the input values 8 – 3 encoder circuit has 8 inputs, one for each of the octal digits and three outputs that generate the corresponding binary number. Enable inputs E_1 should be connected to ground and E_o should be connected to V_{CC}

PROCEDURE:

- Connections are given as per the logic diagram.
- The truth table is verified by varying the inputs.

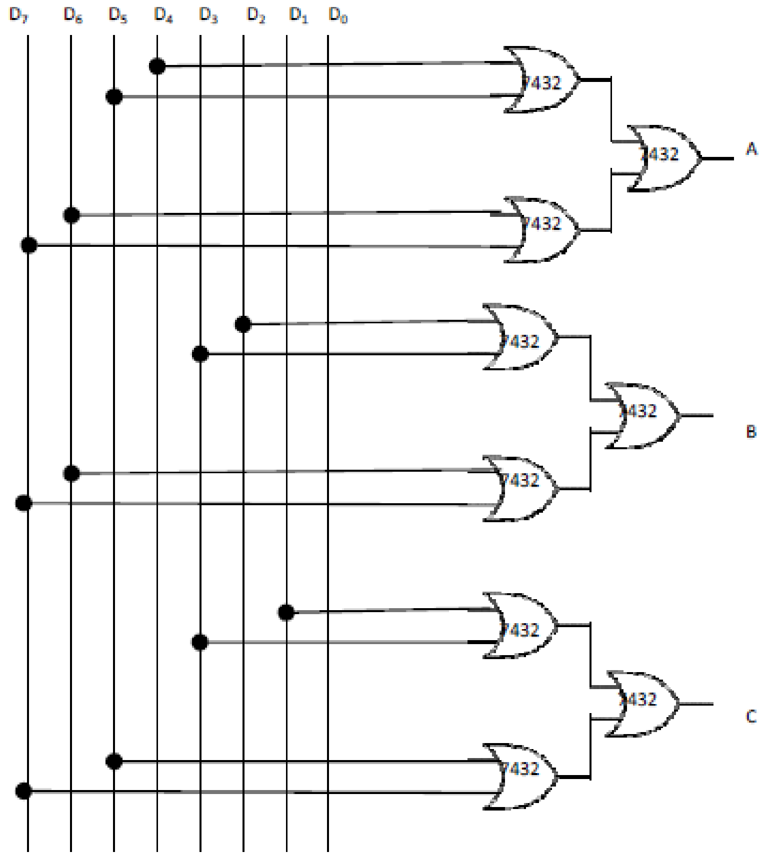
PIN DIAGRAM



TRUTH TABLE

Input								Output		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A	B	C
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

LOGIC DIAGRAM:



ENCODER

AIM:

To design and implement decoder using IC 74155 (3-8 decoder).

APPARATUS REQUIRED:

S. No	Name	Specification	Quantity
1.	IC	74155	1
2.	Digital IC Trainer Kit		1
3.	Patch chords		-

THEORY:

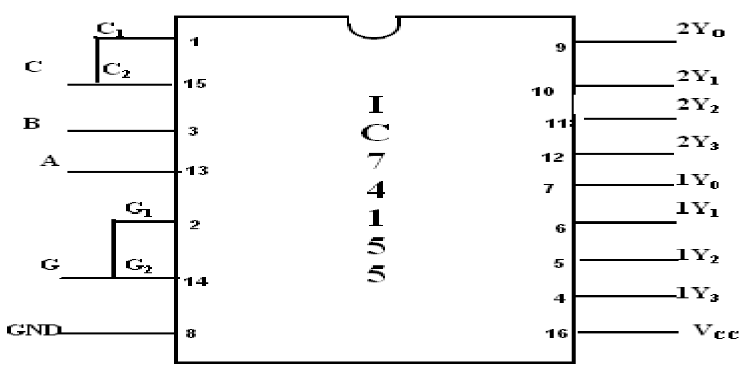
A decoder is a combinational circuit that converts binary information from n input lines to 2^n unique output lines.

In 3-8 line decoder the three inputs are decoded into right outputs in which each output representing one of the minterm of 3 input variables. IC 74155 can be connected as a dual 2*4 decoder or a single 3*8 decoder desired input in C_1 and C_2 must be connected together and used as the C input. G_1 and G_2 should be connected and used as the G (enable) input. G is the enable input and must be equal to 0 for proper operation.

PROCEDURE:

- Connections are given as per the logic diagram.
- The truth table is verified by varying the inputs.

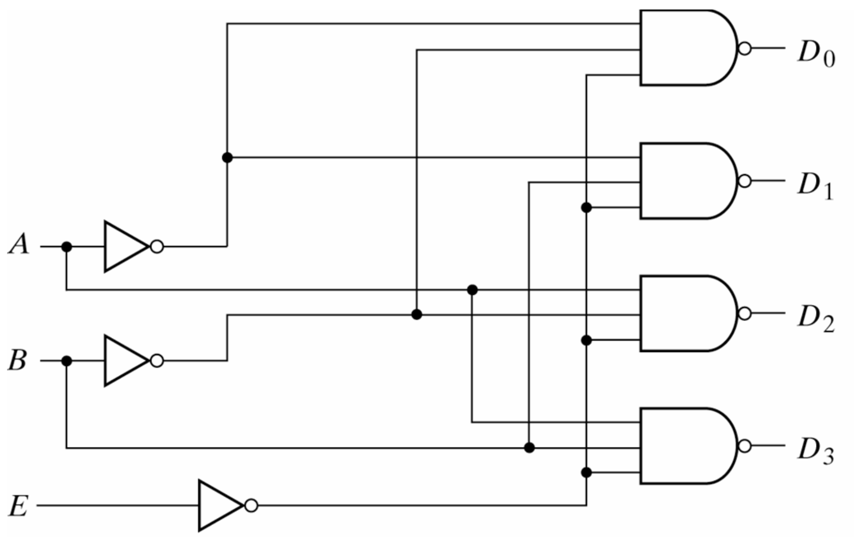
PIN DIAGRAM



TRUTH TABLE

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

LOGIC DIAGRAM:



DISCUSSION QUESTIONS:

1. How the output line will be activated in decoder circuit?
2. What are the necessary steps for implementing higher order decoders?
3. What is the use of code converters?
4. How to convert BCD to Decimal decoder?
5. What is seven segment displays?
6. What is the other name of encoder?
7. What is encoding?
8. What are the applications of encoder?
9. What is BCD encoder?

RESULT:

Thus the encoder and decoder circuits were designed and implemented.

Ex. No:
Date:

12A ASYNCHRONOUS COUNTER

AIM:

To implement and verify the truth table of an asynchronous decade counter.

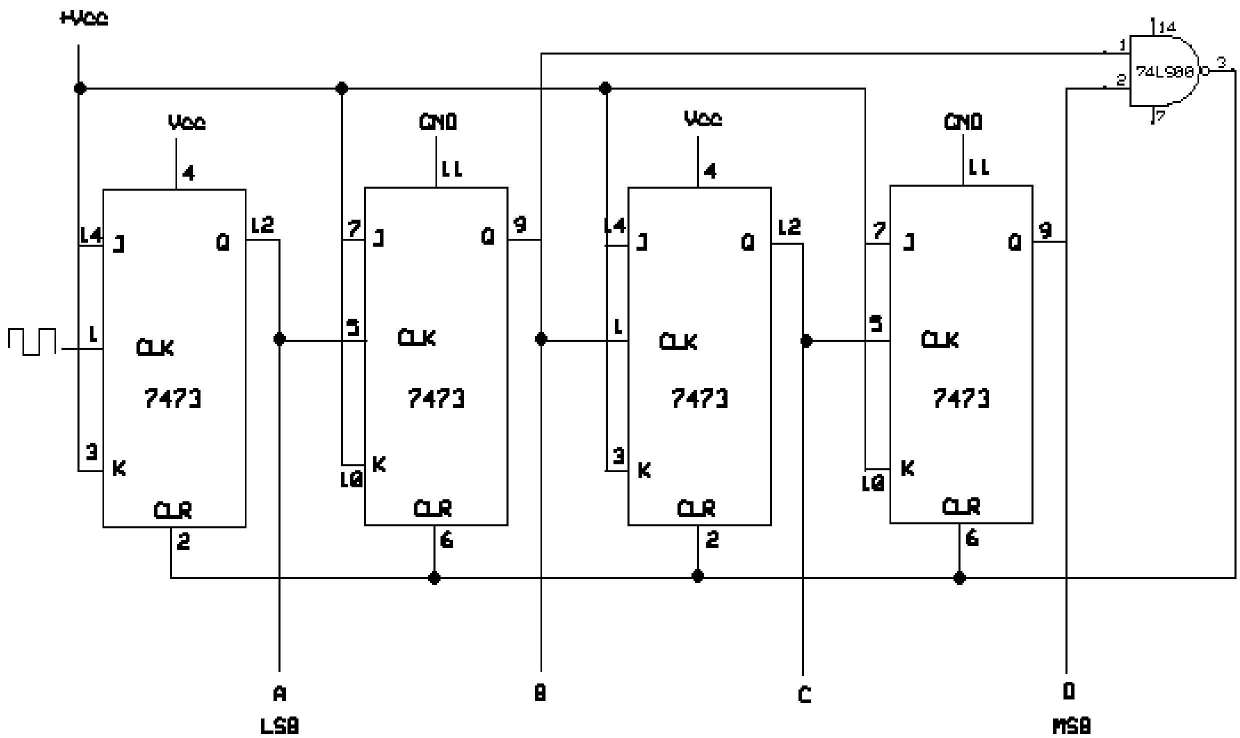
APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit		1
2.	JK Flip Flop	IC 7473	2
4.	NAND gate	IC 7400	1
5.	Connecting wires		As required

THEORY:

Asynchronous decade counter is also called as ripple counter. In a ripple counter the flip flop output transition serves as a source for triggering other flip flops. In other words the clock pulse inputs of all the flip flops are triggered not by the incoming pulses but rather by the transition that occurs in other flip flops. The term asynchronous refers to the events that do not occur at the same time. With respect to the counter operation, asynchronous means that the flip flop within the counter are not made to change states at exactly the same time, they do not because the clock pulses are not connected directly to the clock input of each flip flop in the counter.

CIRCUIT DIAGRAM:



TRUTH TABLE:

S.No	CLOCK PULSE	OUTPUT			
		D(MSB)	C	B	A(LSB)
1	0	0	0	0	0
2	1	0	0	0	1
3	2	0	0	1	0
4	3	0	0	1	1
5	4	0	1	0	0
6	5	0	1	0	1
7	6	0	1	1	0
8	7	0	1	1	1
9	8	1	0	0	0
10	9	1	0	0	1
11	10	0	0	0	0

PROCEDURE:

1. Connections are given as per the circuit diagrams.
2. Apply the input and verify the truth table of the counter.

Ex. No:
Date:

12B SYNCHRONOUS COUNTER

AIM:

To design and implement 4-bit synchronous BCD counter.

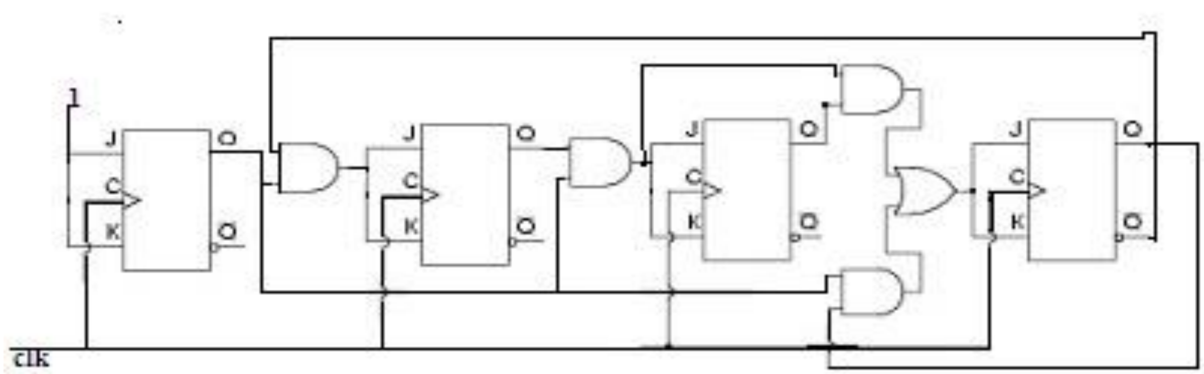
APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit		1
2.	JK Flip Flop	IC 7473	2
3.	AND gate	IC 7408	2
4.	OR gate	IC 7432	1
5	Connecting wires		As required

THEORY:

A counter is a register capable of counting number of clock pulse arriving at the clock input. In synchronous counter all the flip-flops are clocked simultaneously .It is faster in speed because of the propagation delay of the single flip-flop is involved. It is also called as a parallel counter. A BCD synchronous counter can be called as a decade counter or mod-10 counter. It requires 4 flip flops ($10 <= 2^4$). So there are 16 possible states out of which 10 are valid and other 6 are invalid.

CIRCUIT DIAGRAM:



TRUTH TABLE:

Present State				Next State				Excitation Required							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1

PROCEDURE:

1. Connections are given as per the circuit diagrams.
2. Apply the input and verify the truth table of the counter.

DISCUSSION QUESTIONS:

1. Compare synchronous and asynchronous sequential circuits?
2. What is a ripple counter?
3. What is propagation delay in ripple counter?
4. Define MOD counter?
5. What are the applications of counters?
6. State the types of counter?
7. Define bit, byte and word.
8. Define address of a memory.
9. What is a parallel counter?
10. What is the speed of a synchronous counter?

Result:

Thus the synchronous and asynchronous counter circuits were designed and the outputs were verified.

13 STUDY OF ARM EVALUATION SYSTEM

AIM:

To study of ARM processor system and describe the features of architecture.

ARCHITECTURE OF ARM PROCESSOR:

1.1. Features of ARM DEVELOPMENT KIT Processor:

 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory. 128-bit wide interface/accelerator enables high-speed 60 MHz operation. In-System/In-Application Programming (ISP/IAP) via on-chip boot loader software.

 Single flash sector/full chip erase in 400 ms and programming of 256 bytes in 1ms.USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM.The LPC2146/48 provides 8 kB of on-chip RAM accessible to USB by DMA.

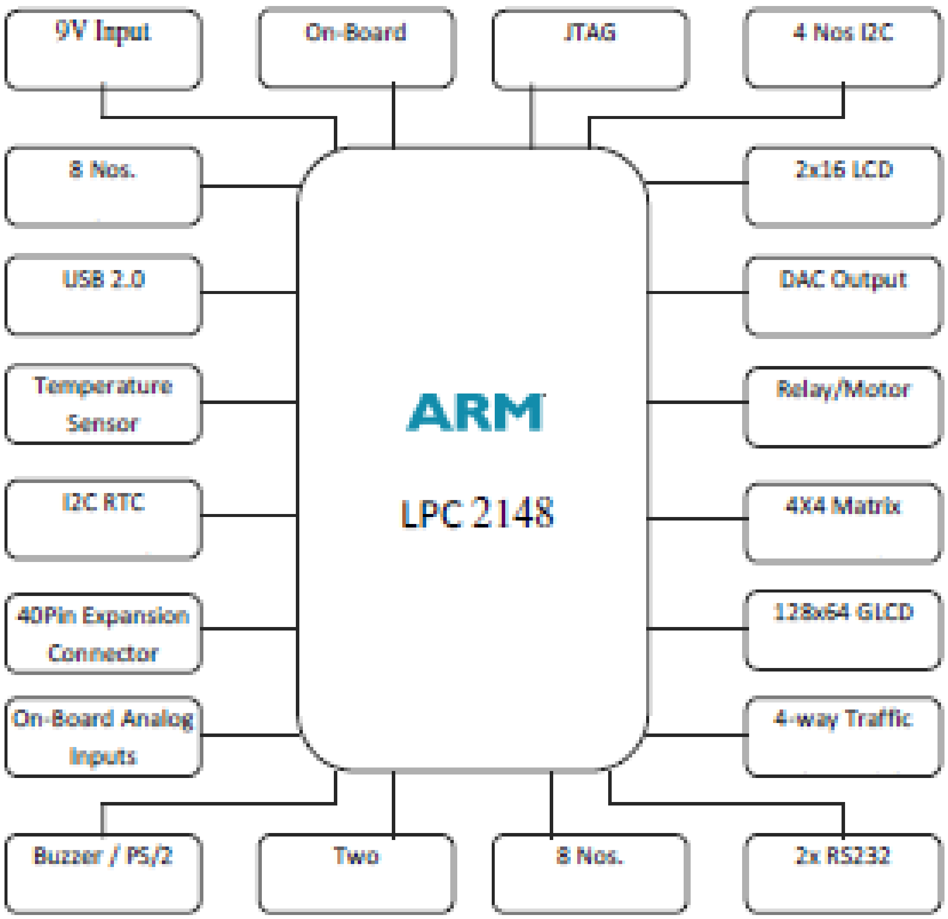
 One or two (LPC2141/42 vs. LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel. Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only).Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.

 Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input. Multiple serial interfaces including two UARTs (16C550), two Fast I2Cbus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.

 Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.Up to 21 external interrupt pins available.

 60MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100μs.On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz.Power saving modes include Idle and Powerdown.

 Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.Processor wake-up from Power-down mode via external interrupt or BOD.Single power supply chip with POR and BOD circuits:CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V



1.2. Power Supply:

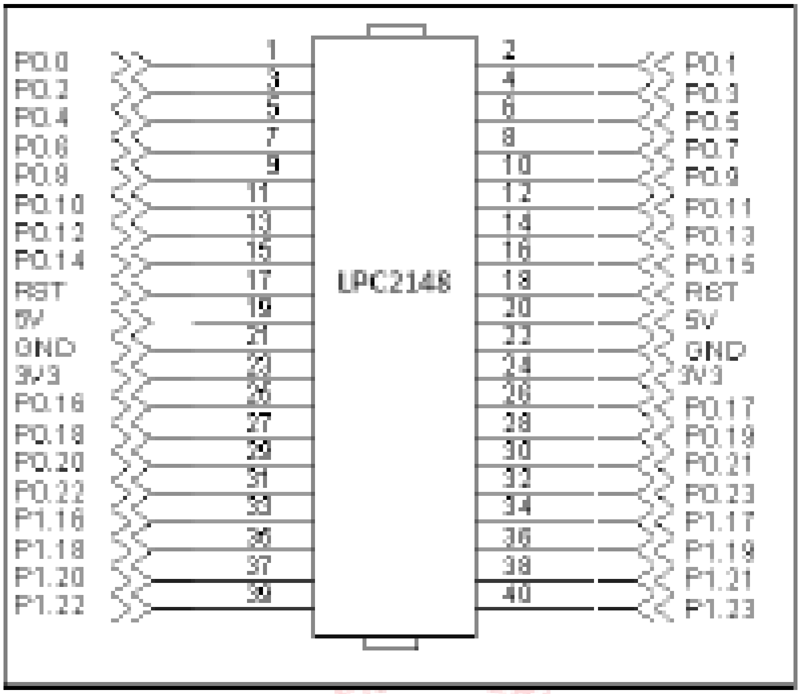
 The external power can be AC or DC, with a voltage between (9V/12V, 1A output) at 230V AC input.The ARM board produces +5V using an LM7805 voltage regulator,which provides supply to the peripherals.

 LM1117 Fixed +3.3V positive regulator used for processor & processor related peripherals.

1.3. Flash Programming Utility

 NXP (Philips)

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology.



1.4. On-board Peripherals:

- 8-Nos. of Point LED’s (Digital Outputs)
- 8-Nos. of Digital Inputs (slide switch)
- 2 Lines X 16 Character LCD Display
- I2C Enabled 4 Digit Seven-segment display
- 128x64 Graphical LCD Display
- 4 X 4 Matrix keypad
- Stepper Motor Interface
- 2 Nos. Relay Interface
- Two UART for serial port communication through PC
- Serial EEPROM
- On-chip Real Time Clock with battery backup
- PS/2 Keyboard interface(Optional)
- Temperature Sensor
- Buzzer(Alarm Interface)
- Traffic Light Module(Optional)

RESULT:

Thus the study of ARM processor has been done and ensured its composition with internal features specifically.

ADDITIONAL EXPERIMENTS

14 PROGRAMS TO VERIFY TIMER AND INTERRUPTS OPERATIONS IN 8051 MICROCONTROLLER

AIM:

To write ALP to generate a square wave of frequency, transfer a data serially from one kit to anotherand to verify the result.

APPARATUS REQUIRED:

8051 microcontroller kit ,key board.

a) Program to generate a square wave of frequency.

Steps to determine the count:

Let the frequency of sqaurewave to be generated be Fs KHz.

And the time period of the squarewave be Ts Sec.

Oscillator Frequency = 11.0592MHz.

One machine cycle = 12 clock periods

Time taken to complete one machine cycle=12*(1/11.0592MHz)= 1.085microsec.

Y(dec) = (Ts/2)/(1.085microsec)

Count(dec) = 65536(dec) – Y(dec)

= Count(hexa)

PROGRAM:

```
            MOV TMOD,#10h       ; To select timer1 & mode1 operation
L1:         MOV TL1,#LOWERORDER BYTE OF THE COUNT
            MOV TH1,#HIGHER ORDER BYTE OF THE COUNT
            SETB TR1            ; to start the timer (TCON.6)
BACK:       JNB TF1,BACK        ; checking the status of timerflag1(TCON.7) for
                                  overflow
            CPL Px.x            ; get the square wave through any of the portpins
                                ; eg. P1.2 (second bit of Port 1)
            CLR TR1             ; stop timer
            CLR TF1             ; clear timer flag for the next cycle
            SJMP L1
```

b) Program to transfer a data serially from one kit to another.

Transmitter:

```
            MOV TMOD,#20H       ; Mode word to select timer1 & mode 2
            MOV TL1,#FDH        ; Initialize timer1 with the count
            MOV TH1,#FFH
            MOV SCON,#50H       ; Control word for serial communication to
                                 to select serial mode1
            SETB TR1            ; Start timer1
            MOV A,#06h
            MOV SBUF,A          ; Transfer the byte to be transmitted to serial
                                 Buffer register.
LOOP:       JNB TI, LOOP        ; checking the status of Transmit interrupt
                                flag
            CLR TI
HERE:       SJMP HERE
```

Receiver:

```
            MOV TMOD,#20H
            MOV TL1,#FDH
            MOV TH1,#FFH
            MOV SCON,#50H
            SETB TR1
LOOP:       JNB RI,LOOP
            MOV A,SBUF
            MOV DPTR,#4500H
            MOVX @DPTR,A
            CLR RI
HERE:       SJMP HERE
```

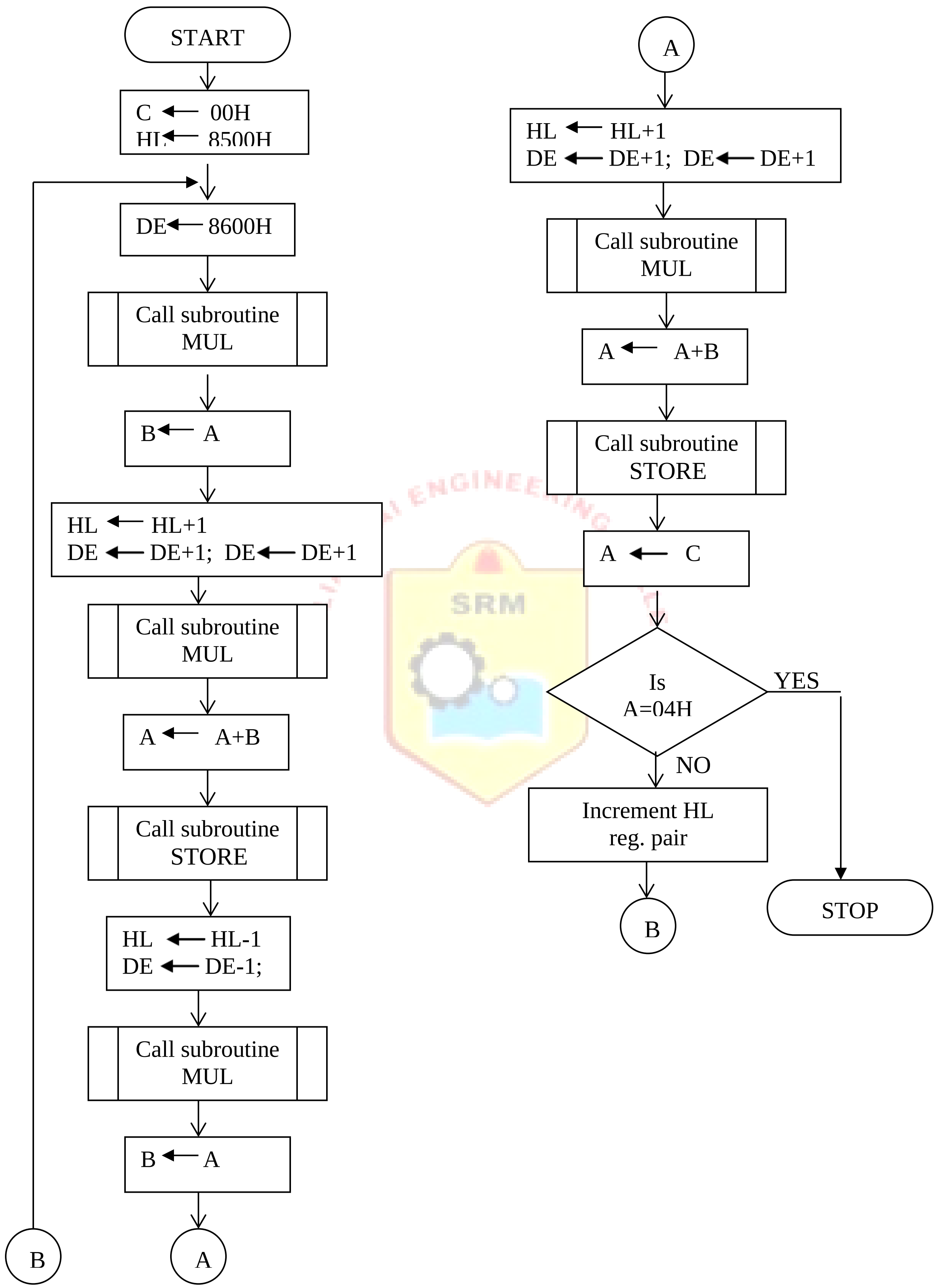
Result:

Thus ALP to generate a square wave of frequency, transfer a data serially from one kit to another and also the result is verified.

VIVA QUESTIONS:

1. What is the use of INT0,INT1?
2. What is the special function of the pin ALE/PROG
3. What is meant by memory interfacing?
4. What is meant by memory mapped IO and IO mapped IO?
5. What are the timer modes are available in 8051?
6. What are interrupt control register?
7. What is the function of IP register?

FLOW CHART:


START
C ← 00H
HL ← 8500H
DE ← 8600H
Call subroutine MUL
B ← A
HL ← HL+1
DE ← DE+1; DE ← DE+1
Call subroutine MUL
A ← A+B
Call subroutine STORE
HL ← HL-1
DE ← DE-1;
Call subroutine MUL
B ← A
B
A
A
HL ← HL+1
DE ← DE+1; DE ← DE+1
Call subroutine MUL
A ← A+B
Call subroutine STORE
A ← C
Is A=04H
YES
NO
Increment HL reg. pair
B
STOP

15 2 X 2 MATRIX MULTIPLICATION

AIM:

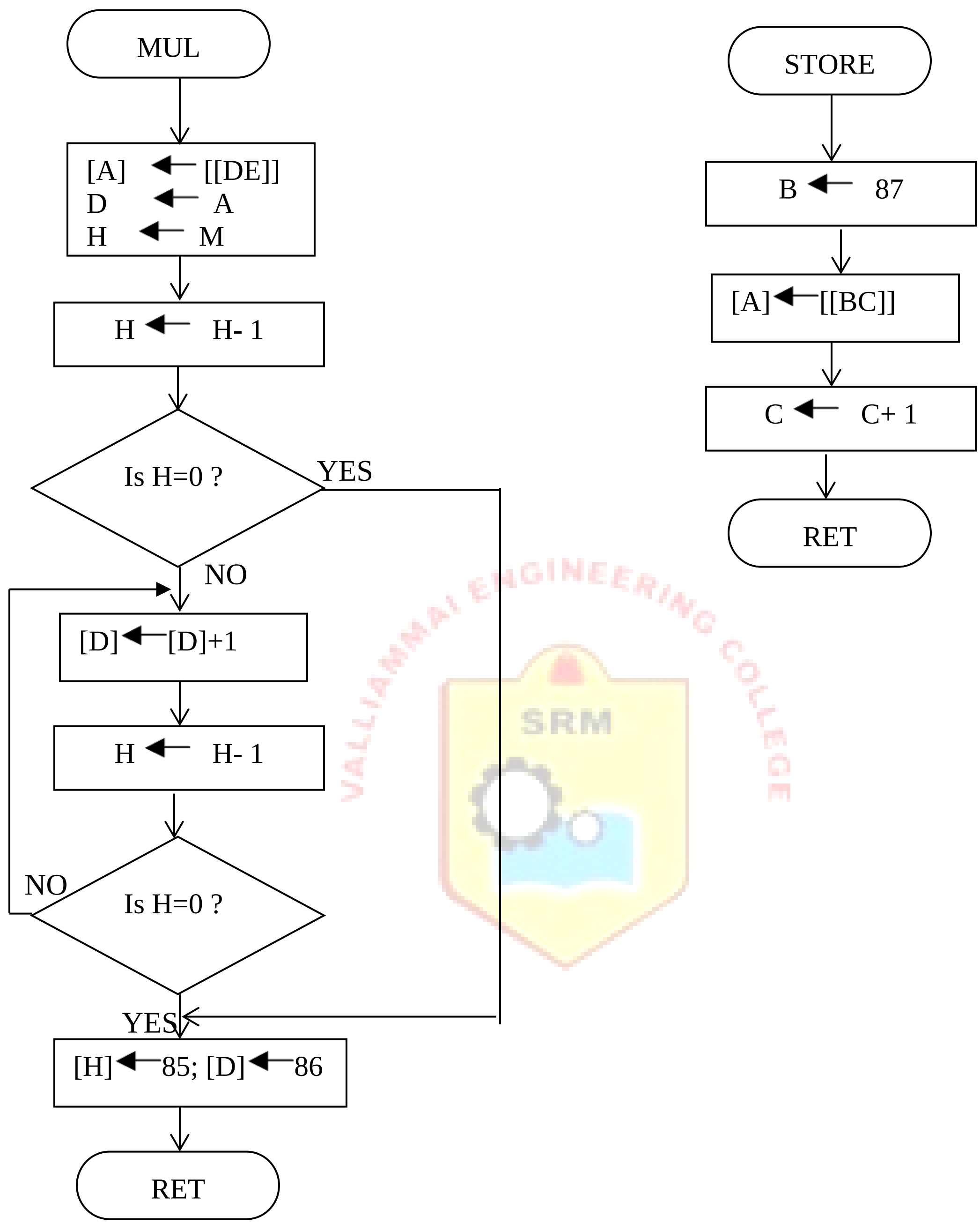
To perform the 2 x 2 matrix multiplication using 8085 microprocessor.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board.

ALGORITHM:

1. Load the 2 input matrices in the separate address and initialize the HL and the DE register pair with the starting address respectively.
2. Call a subroutine for performing the multiplication of one element of a matrix with the other element of the other matrix.
3. Call a subroutine to store the resultant values in a separate matrix.
4. Halt


MUL
[A] ← [[DE]]
D ← A
H ← M
H ← H- 1
Is H=0 ?
YES
NO
[D]←[D]+1
H ← H- 1
Is H=0 ?
NO
YES
[H]←85; [D]←86
RET
STORE
B ← 87
[A]←[[BC]]
C ← C+ 1
RET

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONCS	OPERAND	COMMENT
8100			MVI	C, 00	Clear C reg.
8101					
8102			LXI	H, 8500	Initialize HL reg. to 4500
8103					
8104					
8105		LOOP2	LXI	D, 8600	Load DE register pair
8106					
8107					
8108			CALL	MUL	Call subroutine MUL
8109					
810A					
810B			MOV	B,A	Move A to B reg.
810C			INX	H	Increment HL register pair .
810D			INX	D	Increment DE register pair
810E			INX	D	Increment DE register pair
810F			CALL	MUL	Call subroutine MUL
8110					
8111					
8112			ADD	B	Add [B] with [A]
8113			CALL	STORE	Call subroutine STORE
8114					
8115					
8116			DCX	H	Decrement HL register pair
8117			DCX	D	Decrement DE register pair
8118			CALL	MUL	Call subroutine MUL
8119					
811A					
811B			MOV	B,A	Transfer A reg content to B reg.
811C			INX	H	Increment HL register pair
811D			INX	D	Increment DE register pair
811E			INX	D	Increment DE register pair
811F			CALL	MUL	Call subroutine MUL
8120					
8121					
8122			ADD	B	Add A with B
8123			CALL	STORE	Call subroutine MUL
8124					
8125					
8126			MOV	A,C	Transfer C register content to Acc.

8127			CPI	04	Compare with 04 to check whether all elements are multiplied.
8128					
8129			JZ	LOOP1	If completed, go to loop1
812A					
812B					
812C			INX	H	Increment HL register Pair.
812D			JMP	LOOP2	Jump to LOOP2.
812E					
812F					
8130		LOOP1	HLT		Stop the program.
8131		MUL	LDAX	D	Load acc from the memory location pointed by DE pair.
8132			MOV	D,A	Transfer acc content to D register.
8133			MOV	H,M	Transfer from memory to H register.
8134			DCR	H	Decrement H register.
8135			JZ	LOOP3	If H is zero go to LOOP3.
8136					
8137					
8138		LOOP4	ADD	D	Add Acc with D reg
8139			DCR	H	Decrement H register.
813A			JNZ	LOOP4	If H is not zero go to LOOP4.
813B					
813C					
813D		LOOP3	MVI	H,85	Transfer 85 TO H register.
813E					
813F			MVI	D,86	Transfer 86 to D register.
8140					
8141			RET		Return to main program.
8142		STORE	MVI	B,87	Transfer 87 to B register.
8143					
8144			STAX	B	Load A from memory location pointed by BC pair.
8145			INR	C	Increment C register.
8146			RET		Return to main program.

OBSERVATION:

INPUT				OUTPUT	
4500	01	4600	01	4700	07
4501	02	4601	02	4701	08
4502	03	4602	03	4702	0B
4503	03	4603	03	4703	0F

RESULT:

Thus the 2 x 2 matrix multiplication is performed and the result is stored at 4700,4701 , 4702 & 4703.

VIVA QUESTIONS:

1. How many loops needed to perform matrix multiplication ?
2. What is the condition for two matrix is multipliable ?
3. What is the use of the instruction RET ?
4. What are conditional jump and unconditional jump instructions?
5. What is the next line will execute after Call instruction?
6. Compare Call and DJNZ instructions?
7. If there is no RET statement after CALL instruction whether the program will come to end or not?