

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)
SRM NAGAR, KATTANKULATHUR-603203

**DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION
ENGINEERING**

LAB MANUAL



EI3468-MICROPROCESSORS & MICROCONTROLLERS LAB

IV SEMESTER
(Academic Year 2025-2026 Even Semester)

R2023

Prepared By

Dr.M.JOE MARSHALL, Asst. Prof. (Sr.G)/EIE

Mr. P. TAMILMANI, Asst. Prof. (Sel.G)/EIE

Dr. R. ISSAN RAJ, Asst. Prof. (Sr.G)/EIE

SYLLABUS

COURSE OBJECTIVES:

- To provide training on programming of microprocessors
- To provide training on programming of microcontrollers
- To provide training on interfacing peripherals with microprocessors.
- To provide training on interfacing peripherals with microcontrollers
- To provide training on interfacing I/O devices with arduino / raspberry pi development boards

LIST OF EXPERIMENTS

8-bit Microprocessor

1. Simple arithmetic operations: addition / subtraction / multiplication / division.
2. Programming with control instructions: a. Ascending / Descending order, Maximum / Minimum of numbers. b. Programs using Rotate instructions. c. Hex / ASCII / BCD code conversions.
3. Interface Experiments: with 8085 a. A/D Interfacing & D/A Interfacing.
4. Traffic light controller.
5. I/O Port / Serial communication
6. Read a key, interface display

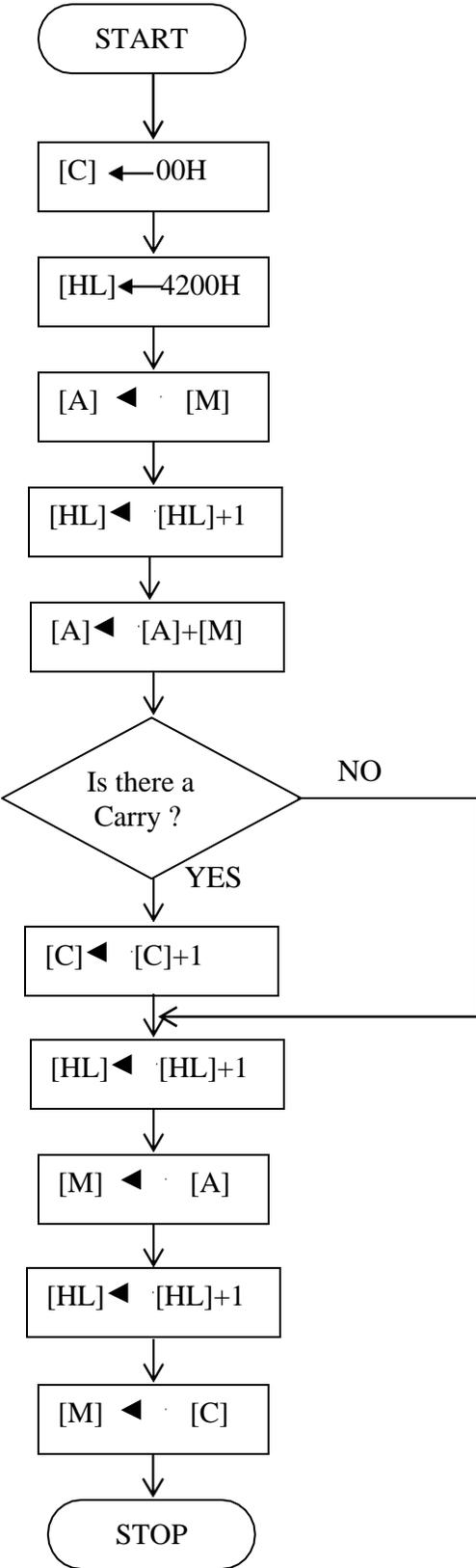
8-bit Microcontroller

7. Demonstration of basic instructions with 8051 Micro controller execution, including: (i) Conditional jumps & looping (ii) Calling subroutines.
8. Programming I/O Port and timer of 8051 (i) study on interface with A/D & D/A (ii) Study on Interface with DC & AC motors
9. Application hardware development using embedded processors.
10. Interfacing of LEDs and sensor with arduino / raspberry pi modules.

CONTENTS

| S. NO. | EXPERIMENT NAME | PAGE NO. |
|--|--|----------|
| 8085 Experiments | | |
| 1A | 8 bit data addition | 4 |
| 1B | 8 bit data subtraction | 8 |
| 1C | 8 bit data multiplication | 12 |
| 1D | 8 bit data division | 16 |
| 2A | Largest element in an array | 20 |
| 2B | Smallest element in an array | 24 |
| 3A | Sorting an array of data in Ascending order | 28 |
| 3B | Sorting an array of data in Descending order | 34 |
| 4A | Decimal to Hexadecimal conversion | 40 |
| 4B | Hexadecimal to decimal conversion | 44 |
| 4C | Hexa decimal TO ASCII conversion | 48 |
| 4D | ASCII to Hexa decimal conversion | 52 |
| 5 | Traffic light controller - Interfacing 8255 with 8085 | 56 |
| 6 | Interfacing 8253 Timer with 8085 | 62 |
| 7 | Interfacing 8279 Keyboard /Display with 8085 microprocessor | 68 |
| 8 | Interfacing 8251 USART with 8085 microprocessor | 72 |
| 9 | Interfacing Analog to Digital converter 8085 microprocessor | 78 |
| 10 | Interfacing Digital to Analog converter 8085 microprocessor | 82 |
| 8051 Experiments | | |
| 11A | 8 bit data addition | 88 |
| 11B | 8 bit data subtraction | 92 |
| 11C | 8 bit data multiplication | 96 |
| 11D | 8 bit data division | 100 |
| 12A | Largest element in an array | 104 |
| 12B | Smallest element in an array | 108 |
| 13 | Interfacing A/D and D/A converter with 8051 microcontroller | 112 |
| 14 | Interfacing stepper motor with 8051 microcontroller | 116 |
| 15 | Interfacing of LEDs and sensor with arduino /raspberry pi modules | 123 |
| 16 | Presentation of hardware development using embedded processor. | --- |
| ADVANCED ADDITIONAL EXPERIMENTS | | |
| 17 | Programs to verify Timer and Interrupts Operations In 8051 microcontroller | 133 |
| 18 | 2 X 2 Matrix Multiplication | 137 |

FLOW CHART:



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 | | 4202 | |
| 4201 | | 4203 | |

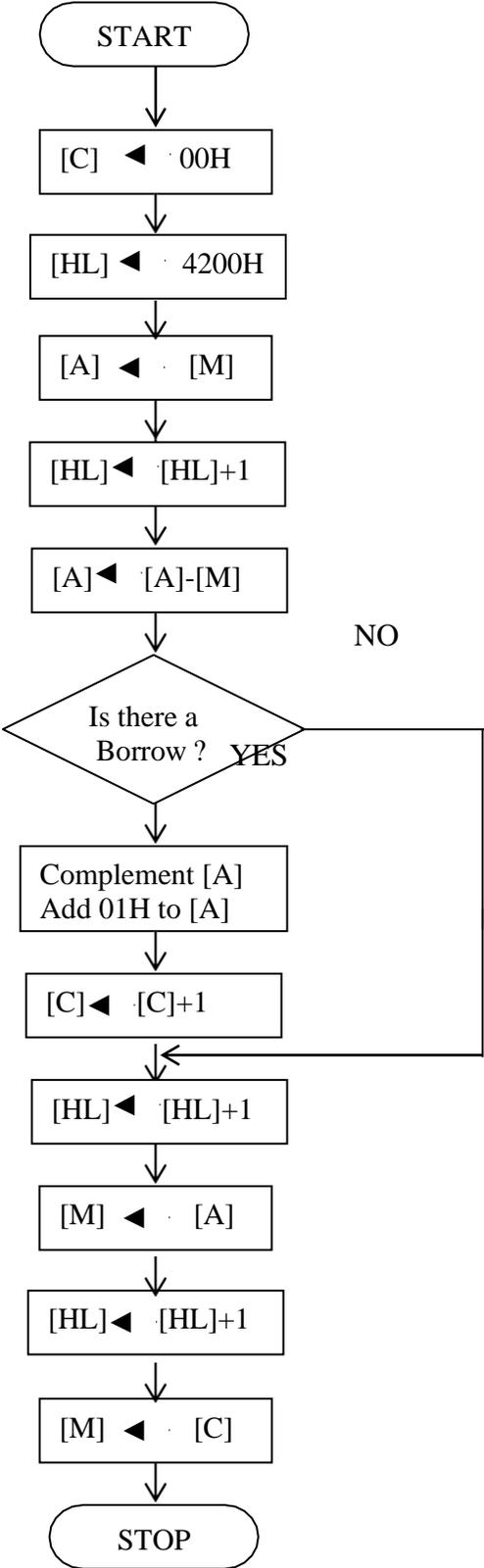
RESULT:

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:



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 | | 4202 | |
| 4201 | | 4203 | |

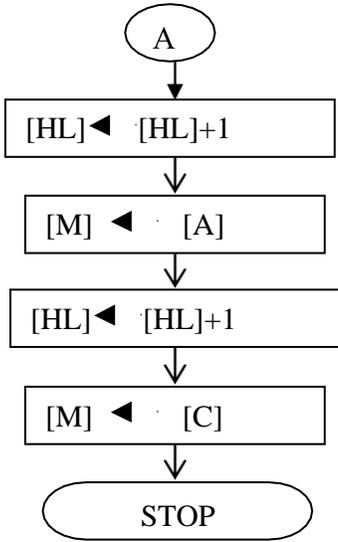
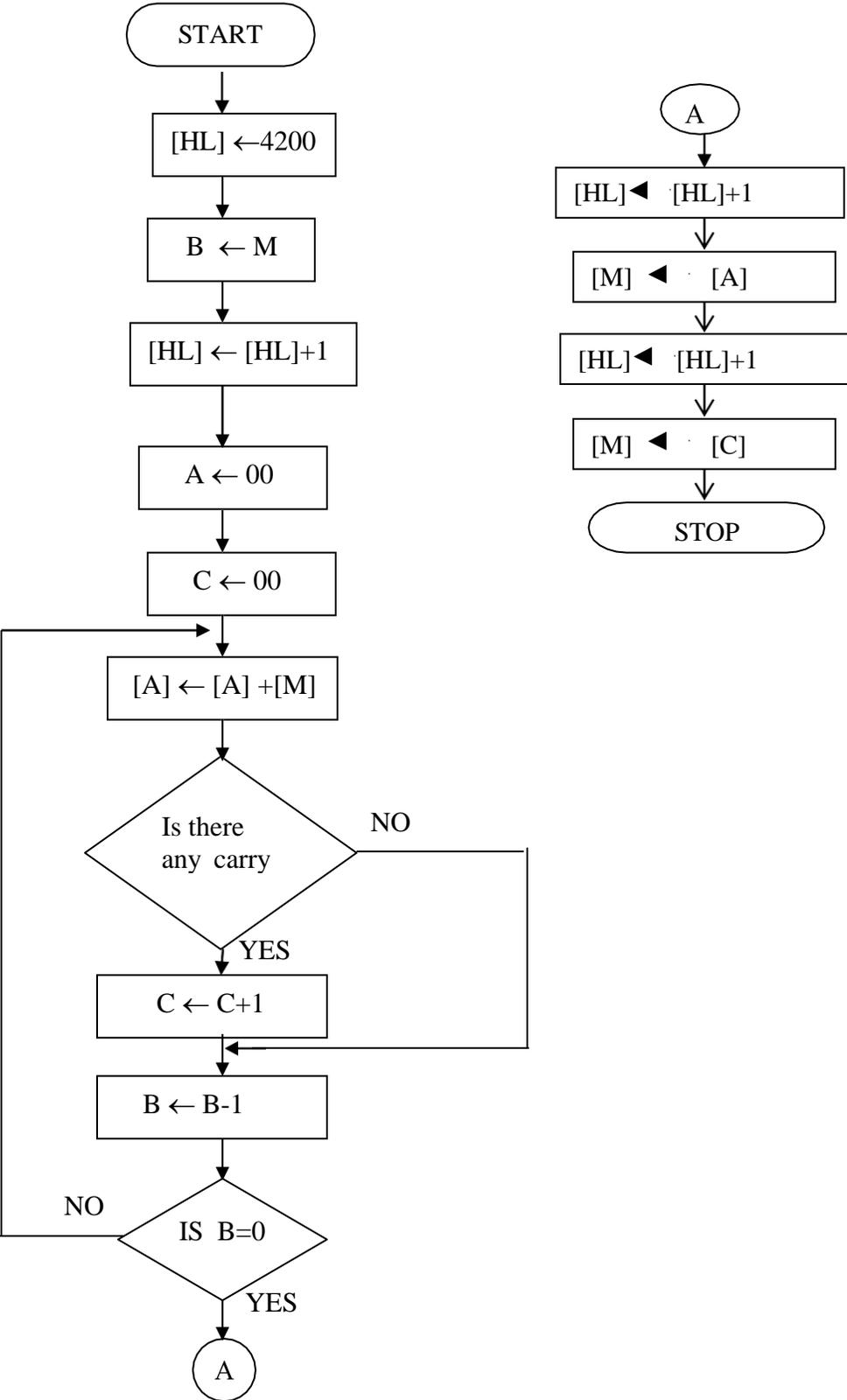
RESULT:

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:



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 | | 4202 | |
| 4201 | | 4203 | |

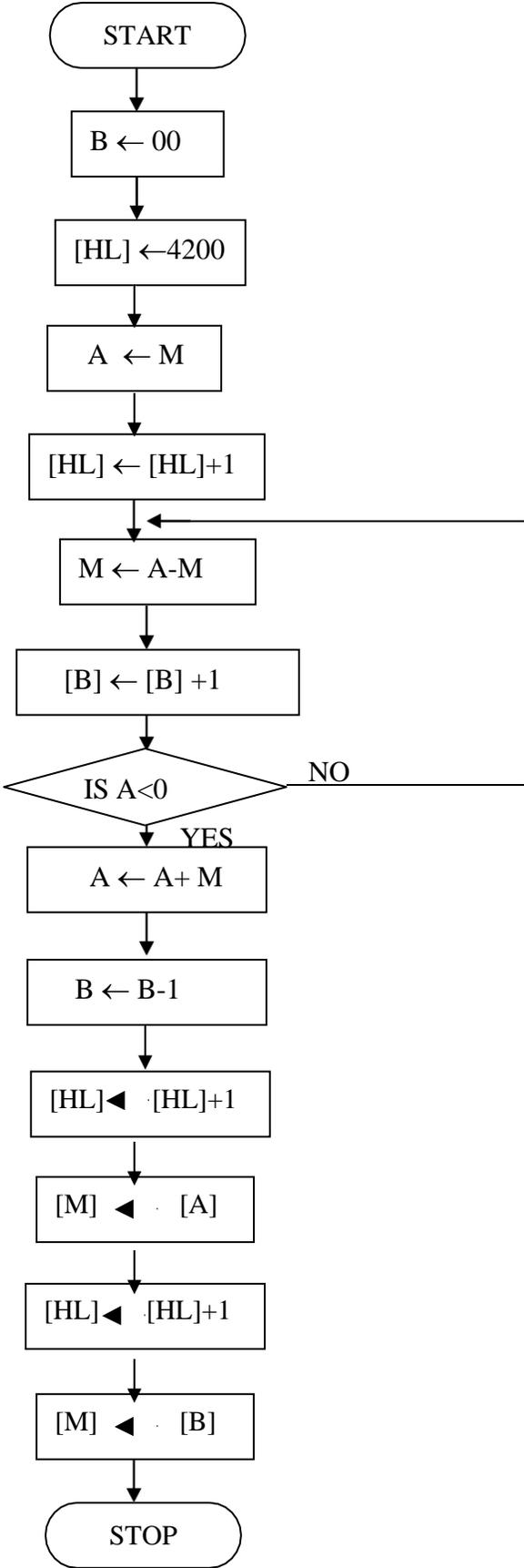
RESULT:

Thus the 8-bit multiplication was done in 8085 μ p 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:



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 | OPERAN D | COMMENTS |
|---------|--------|-------|-----------|-------------|---|
| 4100 | | | MVI | B,00 | Clear B reg for quotient |
| 4101 | | | | | |
| 4102 | | | LXI | H,4200 | Initialize HL reg. to 4500H |
| 4103 | | | | | |
| 4104 | | | | | |
| 4105 | | | MOV | A,M | Transfer dividend to acc. |
| 4106 | | | INX | H | Increment HL reg. to point next mem. Location. |
| 4107 | | LOOP | SUB | M | Subtract divisor from dividend |
| 4108 | | | INR | B | Increment B reg |
| 4109 | | | JNC | LOOP | Jump to LOOP if result does not yield borrow |
| 410A | | | | | |
| 410B | | | | | |
| 410C | | | ADD | M | Add divisor to acc. |
| 410D | | | DCR | B | Decrement B reg |
| 410E | | | INX | H | Increment HL reg. to point next mem. Location. |
| 410F | | | MOV | M,A | Transfer the remainder from acc. to memory. |
| 4110 | | | INX | H | Increment HL reg. to point next mem. Location. |
| 4111 | | | MOV | M,B | Transfer the quotient from B reg. to memory. |
| 4112 | | | HLT | | Stop the program |

OBSERVATION:

| INPUT | | OUTPUT | |
|---------|------|---------|------|
| ADDRESS | DATA | ADDRESS | DATA |
| 4200 | | 4202 | |
| 4201 | | 4203 | |

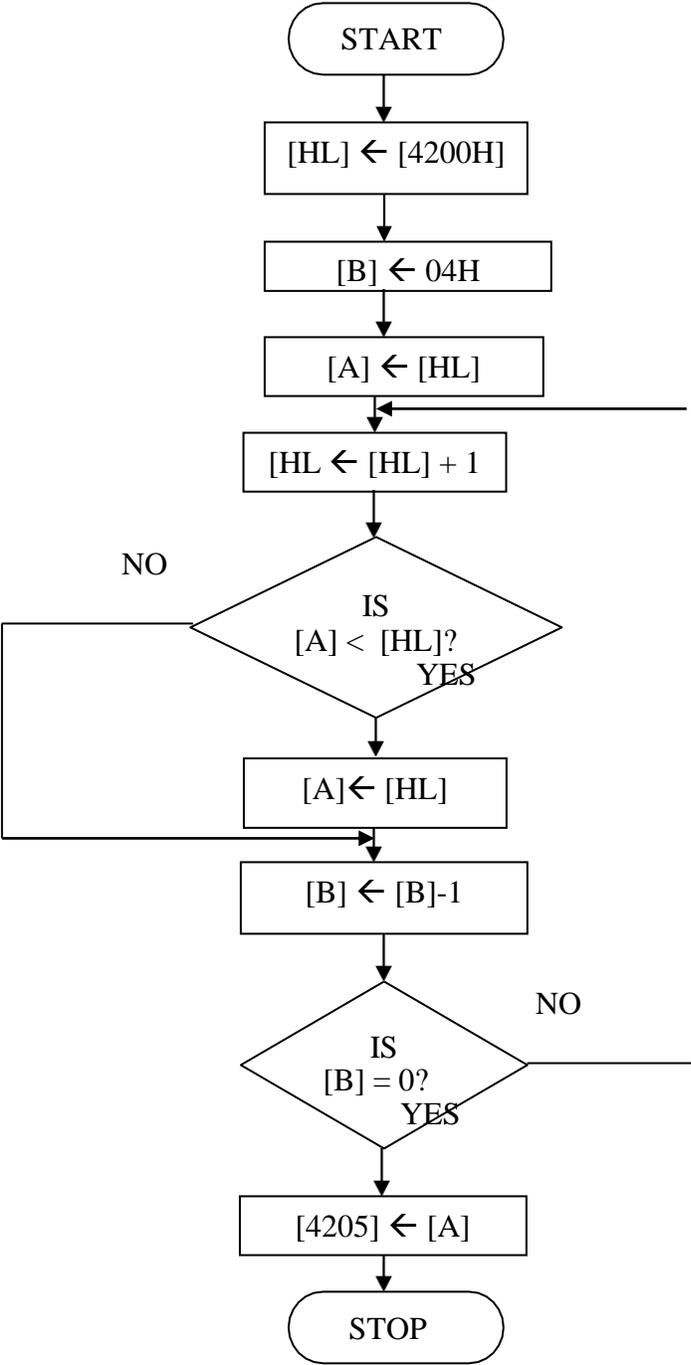
RESULT:

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:



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 | | 4205 | |
| 4201 | | | |
| 4202 | | | |
| 4203 | | | |
| 4204 | | | |

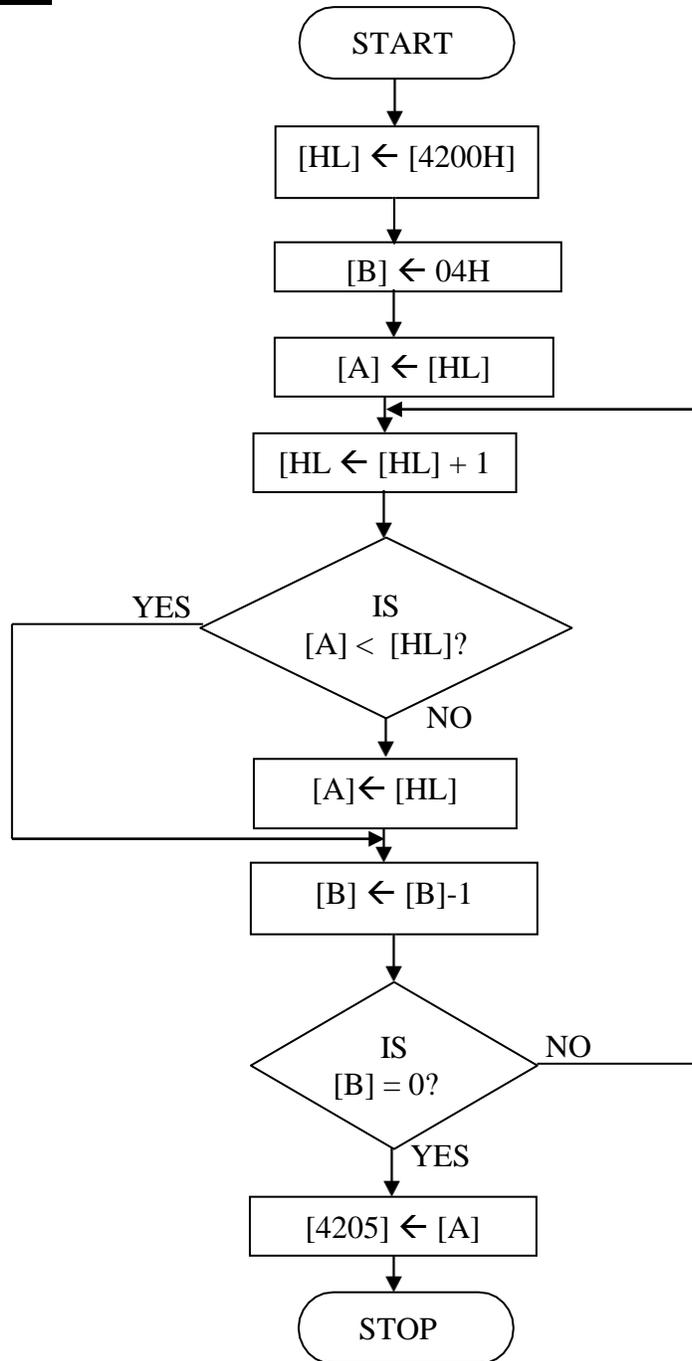
RESULT:

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:



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 | | 4205 | |
| 4201 | | | |
| 4202 | | | |
| 4203 | | | |
| 4204 | | | |

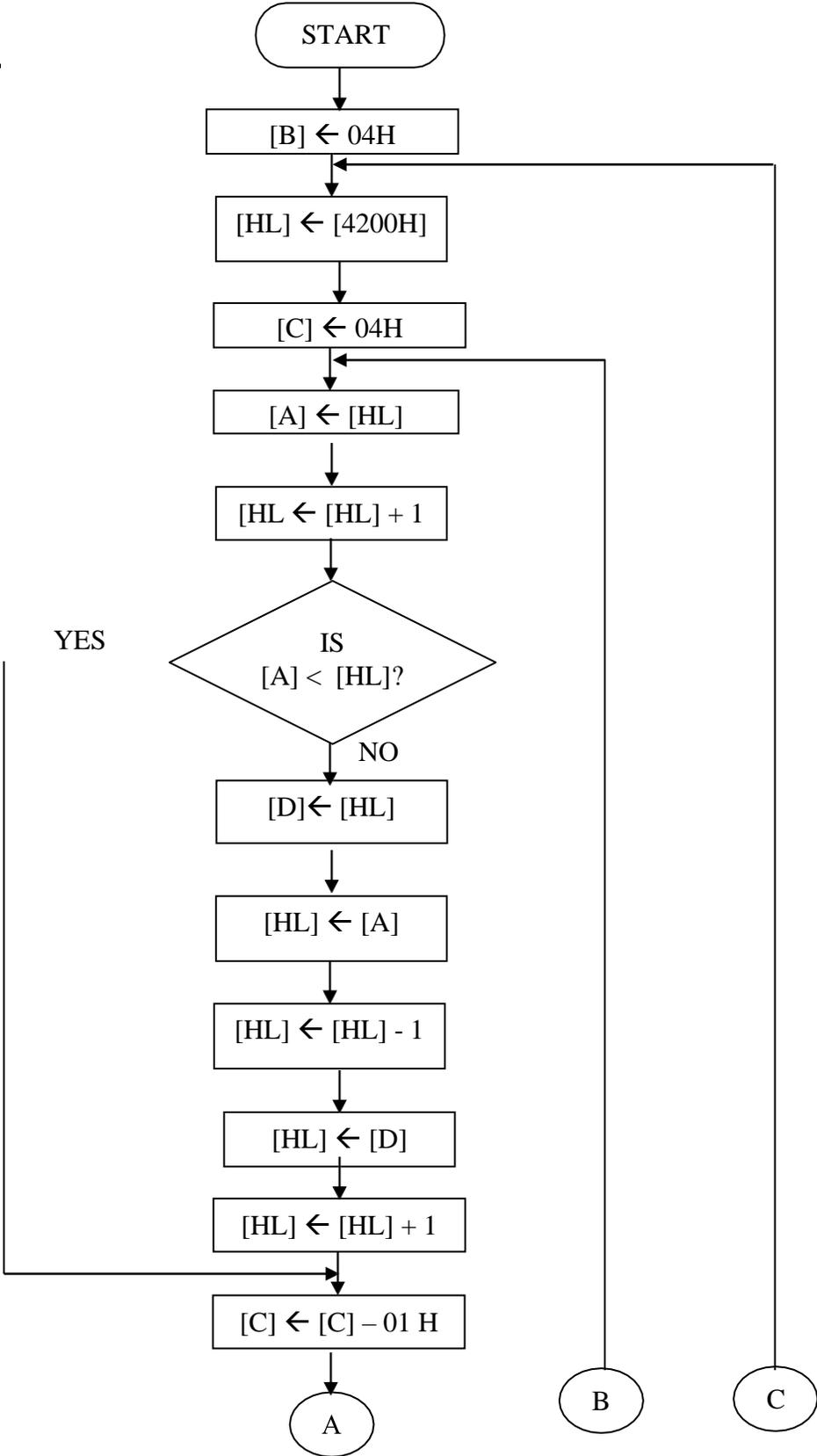
RESULT:

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:



3(A) ASCENDING ORDER

AIM:

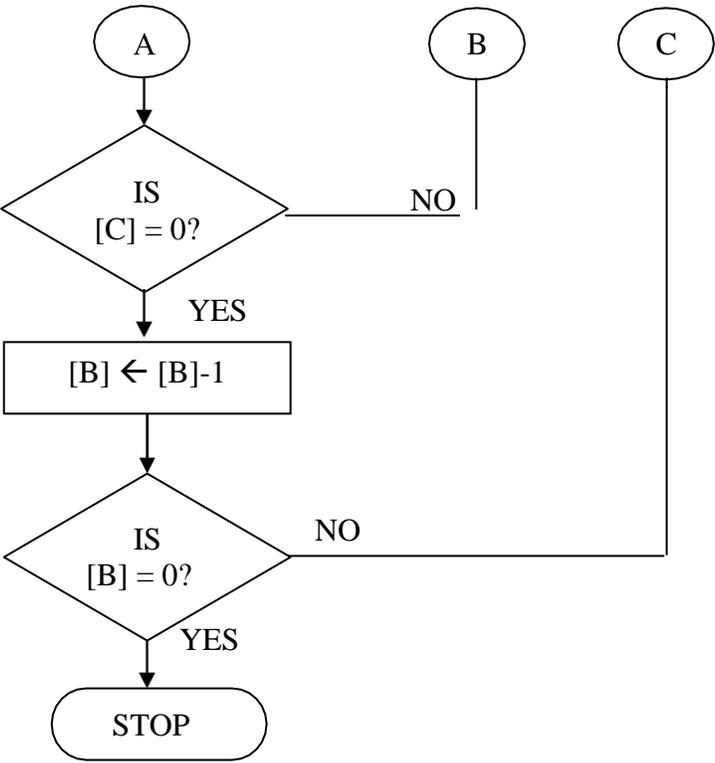
To sort the given numbers in the ascending order using 8085 microprocessor.

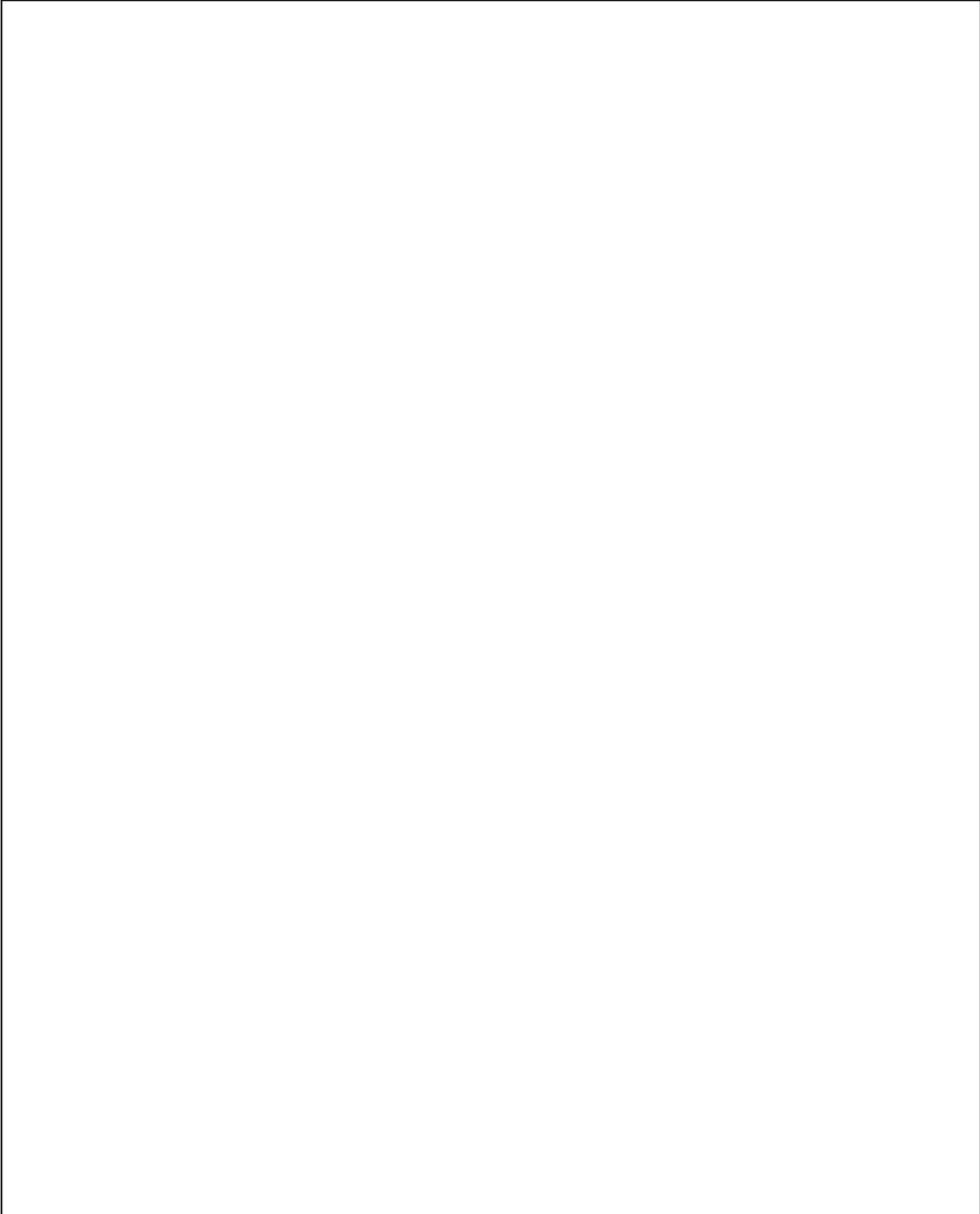
APPARATUS REQUIRED:

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





PROGRAM:

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

OBSERVATION:

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

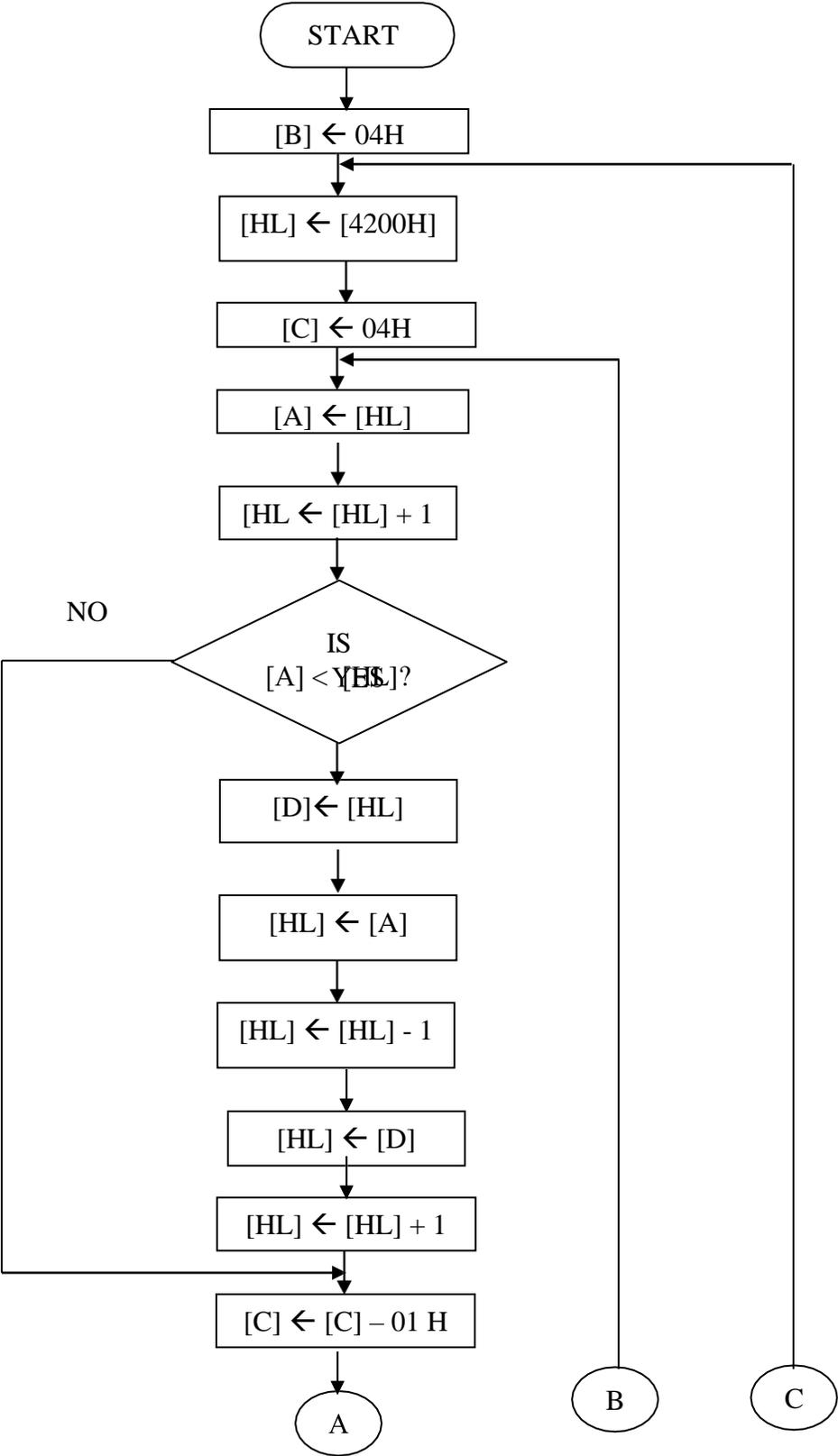
RESULT:

Thus the ascending order program is executed and 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:



3(B) DESCENDING ORDER

AIM:

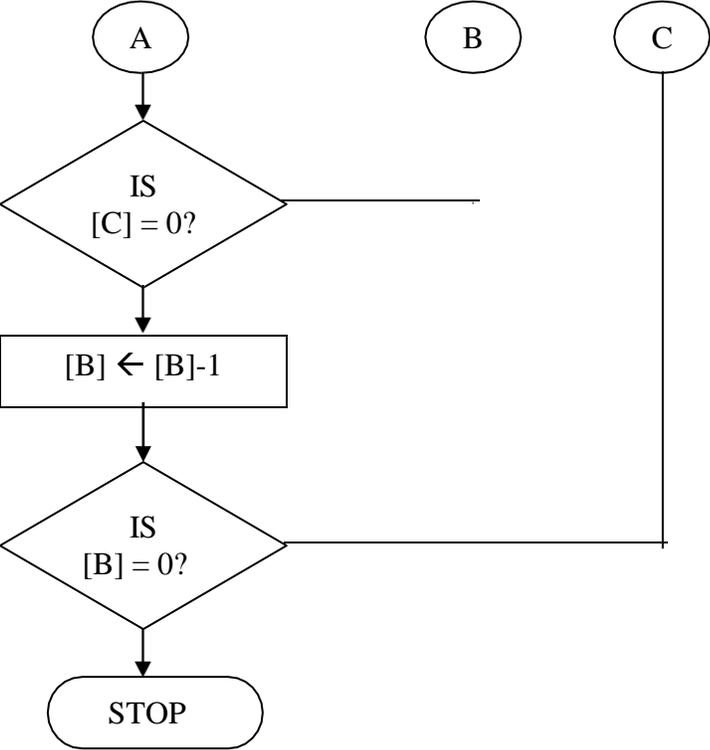
To sort the given numbers in the descending order using 8085 microprocessor.

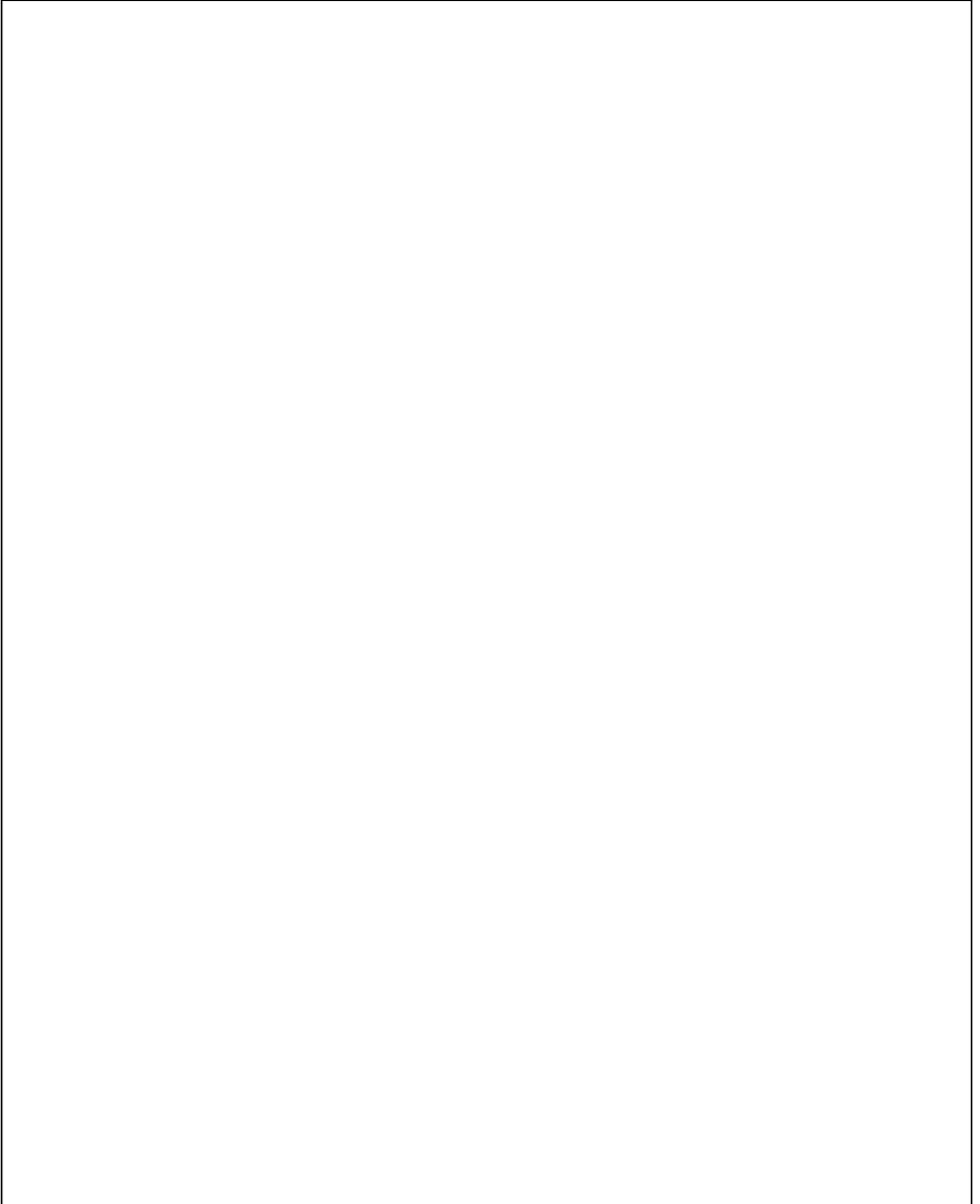
APPARATUS REQUIRED:

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





PROGRAM:

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

OBSERVATION:

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

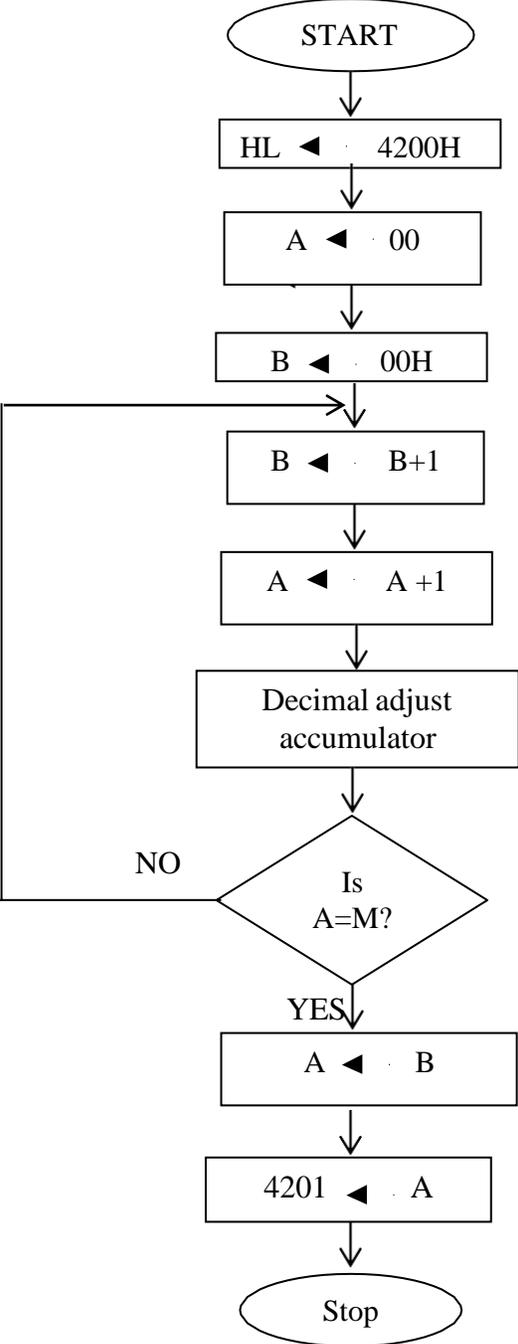
RESULT:

Thus the descending order program is executed and 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:



4(A) 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 | | 4201 | |

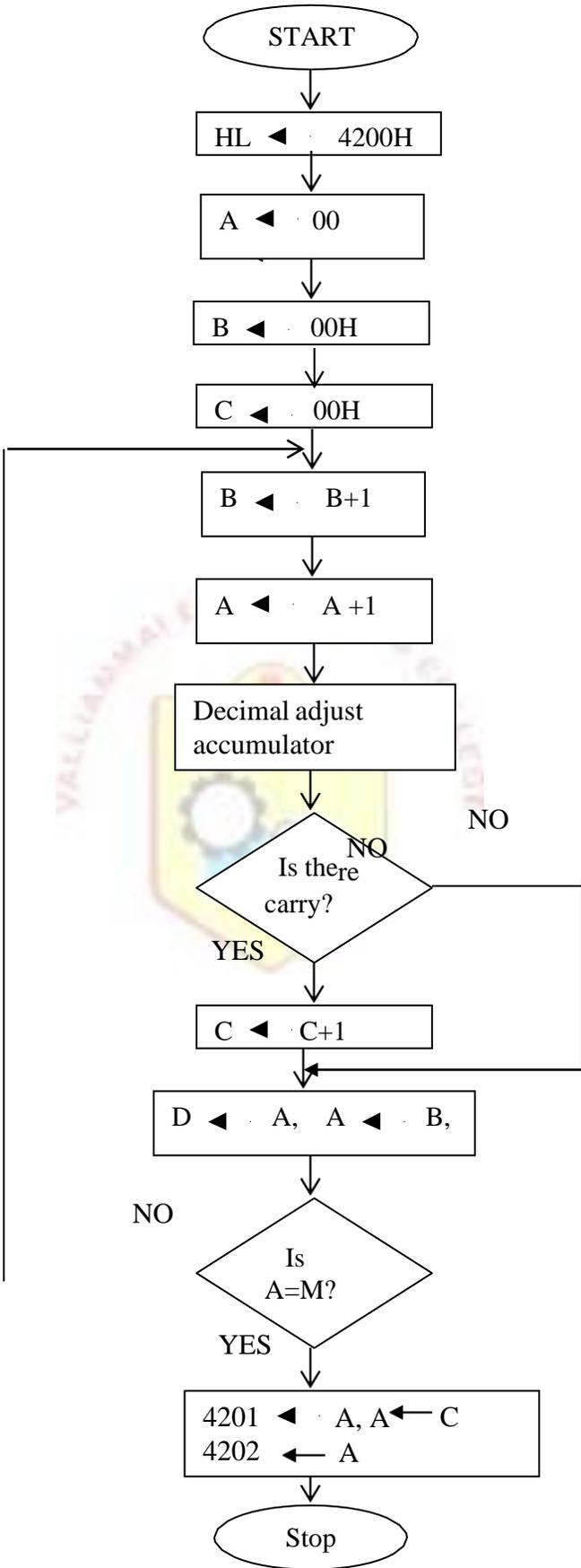
RESULT:

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:



4(B) 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 | | 4201 | |
| | | 4202 | |

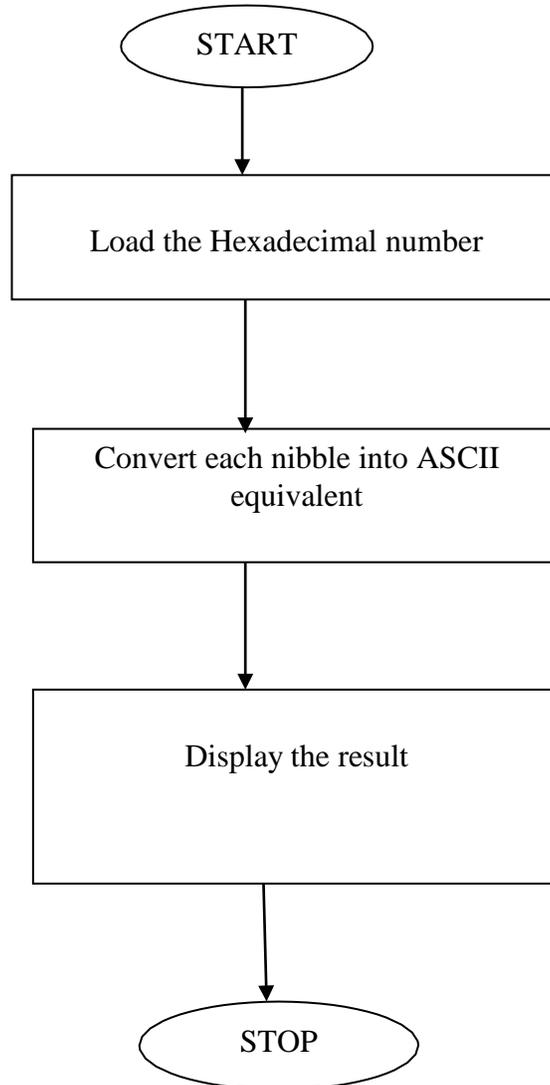
RESULT:

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:



4(C). CODE CONVERSION –HEXADECIMAL TO ASCII

Aim

To write an assembly language program to convert 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 | |
| 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 0A |
| 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 | | 4201 | |
| | | 4202 | |

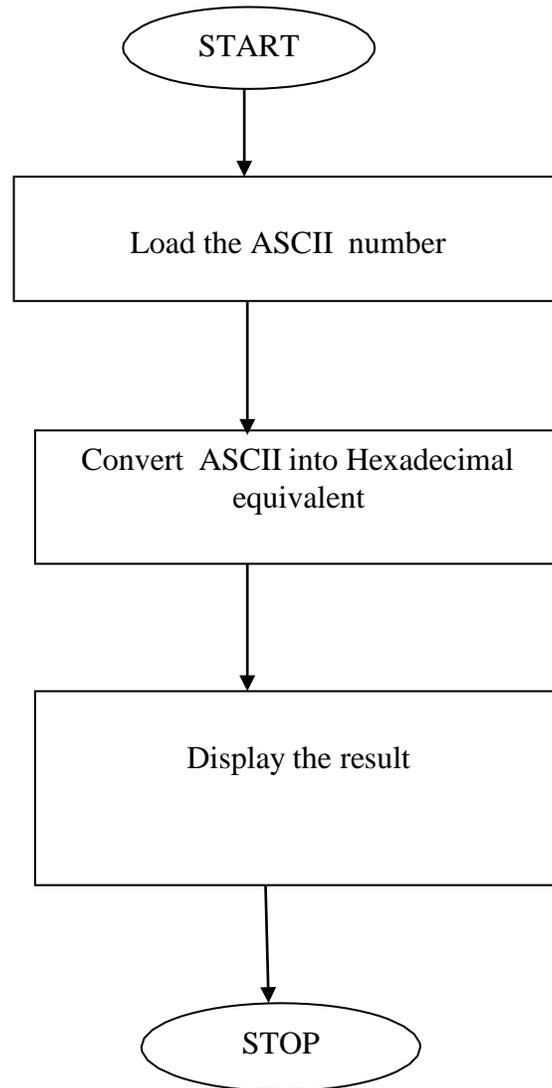
Result :

Thus assembly language program to convert 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:



4(D). CODE CONVERSION –ASCII TO HEXADECIMAL

AIM:

To write an assembly language program to convert 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 ASCII 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 0A |
| 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 | | 4201 | |

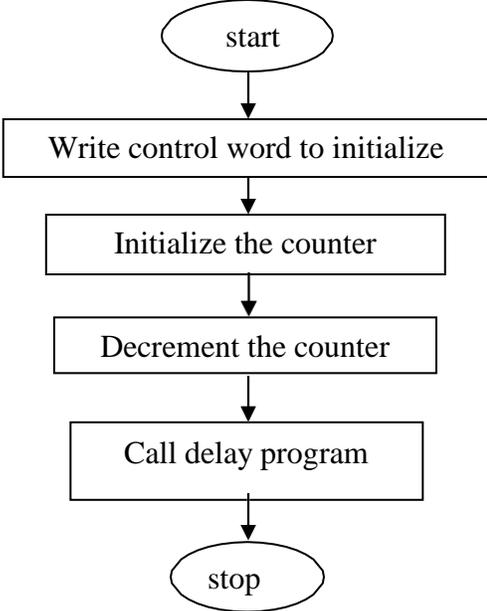
Result :

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

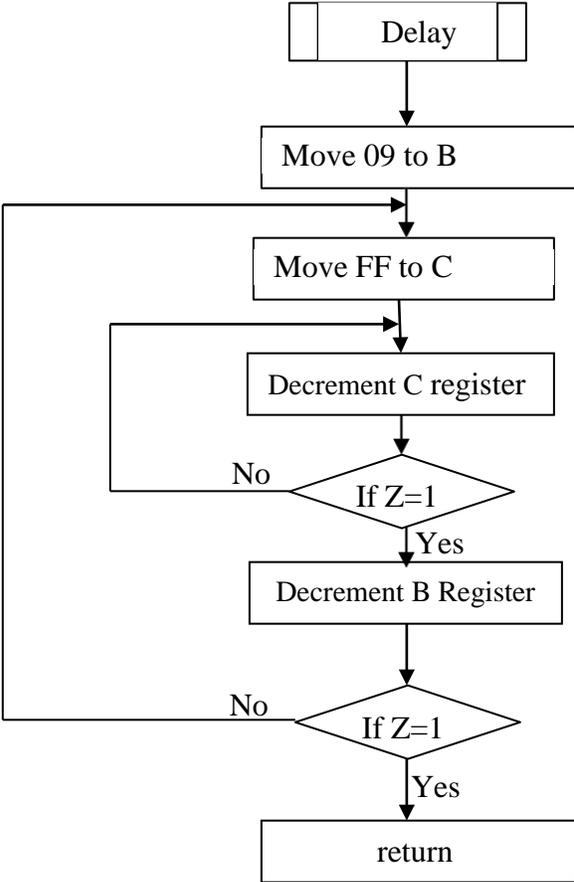
VIVA QUESTIONS:

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?

LOW CHART:



Delay Subroutine:



5 . 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 μ p kit, 8255 Interface board, DC regulated power supply, VXT parallel bus, Traffic light controller interface board.

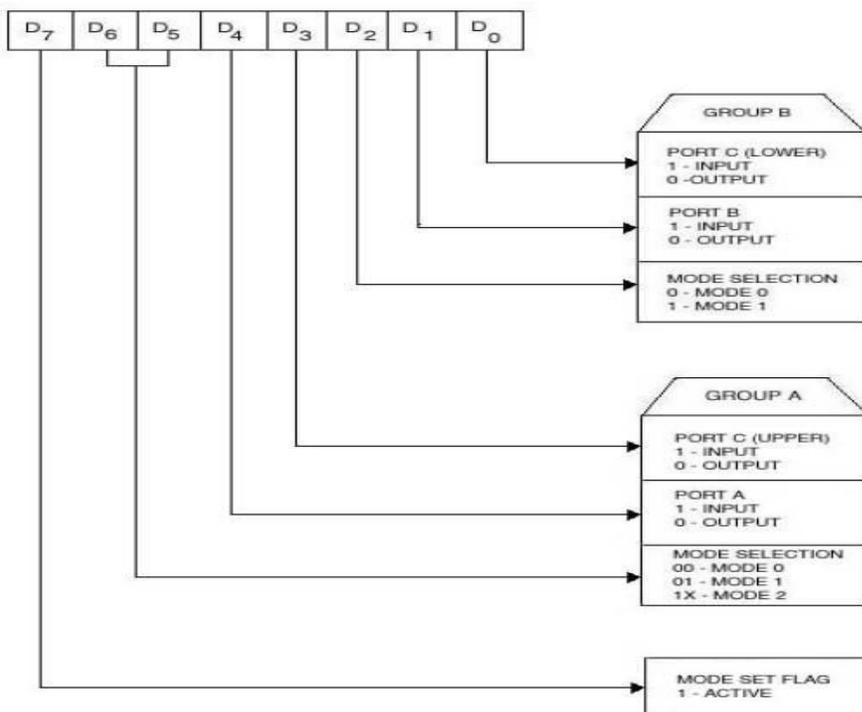
I/O MODES:

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:

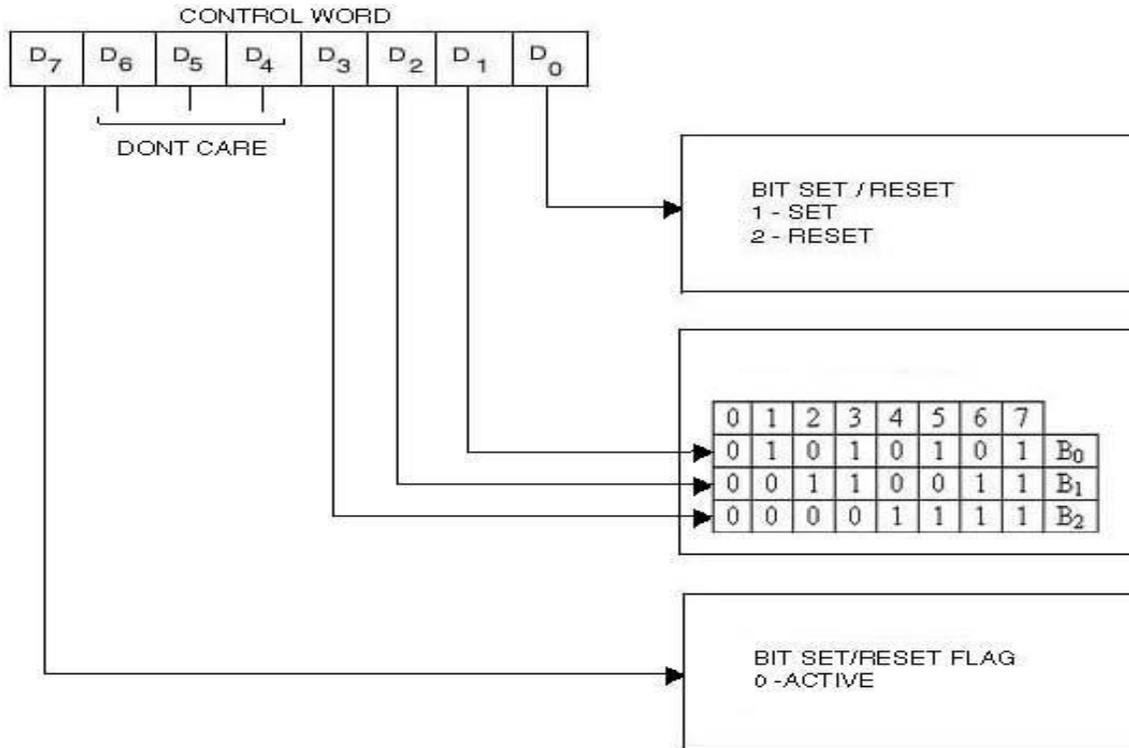
| ADDRESS | OPCODES | LABEL | MNEMONICS | OPERAND | COMMENTS |
|---------|---------|-------|-----------|------------|-----------------------------------|
| 4100 | | | LXI | H, Data | Load the data in HL register pair |
| 4103 | | | MVI | C,04 | Move 04 to c register |
| 4105 | | | MOV | A,M | Move M to A |
| 4106 | | | OUT | CNT | Out to control 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 0D to C register |
| 4122 | | LOOP3 | LXI | D,FF,FF | Load D register with FF |
| 4125 | | LOOP2 | DCX | D | Decrement D register |
| 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 mode 1

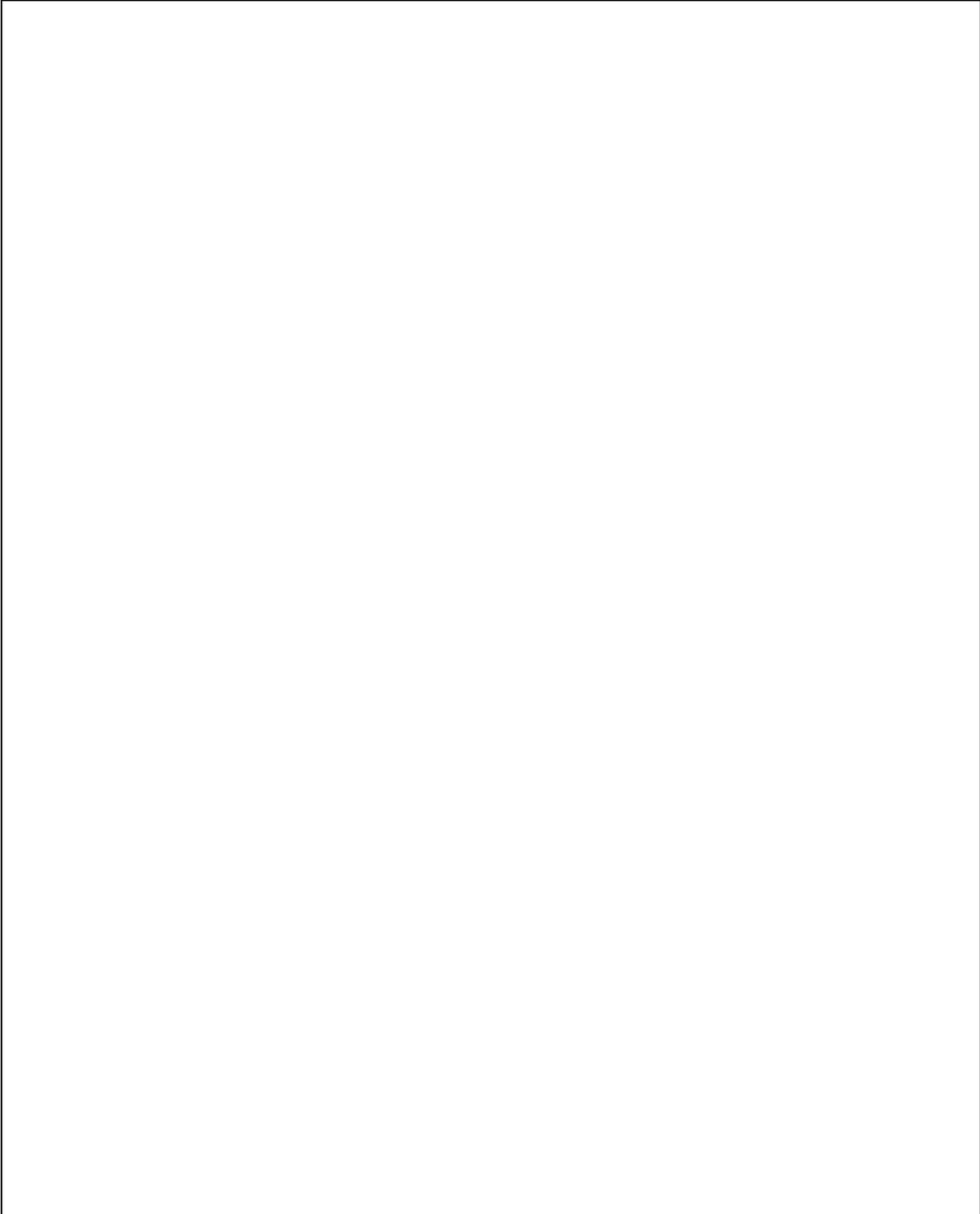
BSR MODE (Bit Set Reset mode)



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



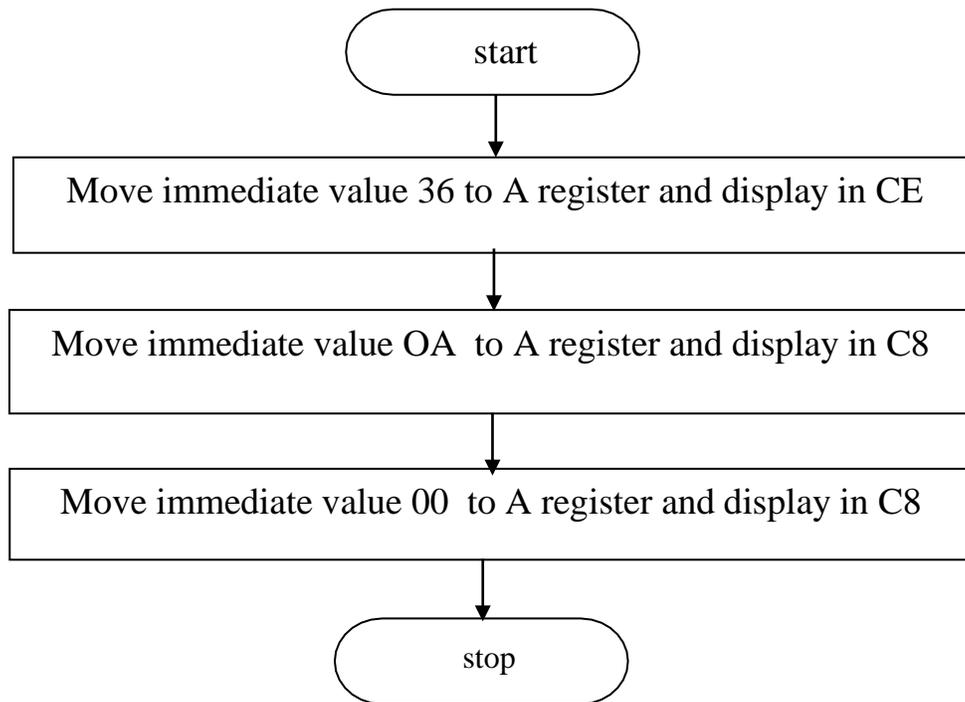
RESULT:

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

VIVA QUESTIONS:

1. When the 82C55 is reset, its I/O ports are all initialized 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 an 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?

FLOW CHART:



Mode 0 – Interrupt on terminal count:

Program:

| Address | Opcodes | Label | Mnemonic | Operands | Comments |
|---------|---------|--------|----------|----------|-------------------------|
| 4100 | | START: | MVI | A, 36 | Channel 0 in mode 0 |
| 4102 | | | OUT | CE | Send Mode Control word |
| 4104 | | | MVI | A, 0A | LSB of count |
| 4106 | | | OUT | C8 | Write count to register |
| 4108 | | | MVI | A, 00 | MSB of count |
| 410A | | | OUT | C8 | Write count to register |
| 410C | | | HLT | | |

Mode 1 – Programmable ONE-SHOT:

Program:

| Address | Opcodes | Label | Mnemonic | Operands | Comments |
|---------|---------|--------|----------|----------|-------------------------|
| 4100 | | START: | MVI | A, 32 | Channel 0 in mode 1 |
| 4102 | | | OUT | CE | Send Mode Control word |
| 4104 | | | MVI | A, 05 | LSB of count |
| 4106 | | | OUT | C8 | Write count to register |
| 4108 | | | MVI | A, 00 | MSB of count |
| 410A | | | OUT | C8 | Write count to register |
| 410C | | | OUT | D0 | Trigger Gate0 |
| 4110 | | | HLT | | |

6. INTERFACING 8253 TIMER WITH 8085

AIM:

To interface 8253 Interface board to 8085 μ p and verify the operation of 8253 in six different modes.

APPARATUS REQUIRED:

8085 μ p kit, 8253 Interface board, DC regulated power supply, VXT parallel bus, CRO.

Mode 0 – Interrupt on terminal count:

The output will be initially low after mode set operations. After loading the counter, the output will be remaining low while counting and on terminal count; the output will become high, until reloaded again.

Let us set the channel 0 in mode 0. Connect the CLK 0 to the debounce circuit by changing the jumper J3 and then execute the following program.

It is observed in CRO that the output of Channel 0 is initially LOW. After giving six clock pulses, the output goes HIGH.

Mode 1 – Programmable ONE-SHOT:

After loading the counter, the output will remain low following the rising edge of the gate input. The output will go high on the terminal count. It is retriggerable; hence the output will remain low for the full count, after any rising edge of the gate input.

Example:

The following program initializes channel 0 of 8253 in Mode 1 and also initiates triggering of Gate 0. OUT 0 goes low, as clock pulse after triggering the goes back to high level after 5 clock pulses. Execute the program, give clock pulses through the debounce logic and verify using CRO.

Mode 2 – Rate Generator:

It is a simple divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected but the subsequent period will reflect the new value

Example:

Using Mode 2, Let us divide the clock present at Channel 1 by 10. Connect the CLK1 to PCLK.

In CRO observe simultaneously the input clock to channel 1 and the output at Out1.

Mode 3 Square wave generator:

It is similar to Mode 2 except that the output will remain high until one half of count and go low for the other half for even number count. If the count is odd, the output will be high for (count + 1)/2 counts. This mode is used of generating Baud rate for 8251A (USART).

Example:

We utilize Mode 0 to generate a square wave of frequency 150 KHz at channel 0.

Mode 2 – Rate Generator:

Program:

| Address | Opcodes | Label | Mnemonic | Operands | Comments |
|---------|---------|--------|----------|----------|-------------------------|
| 4100 | 3E 74 | START: | MVI | A, 74 | Channel 1 in mode 2 |
| 4102 | D3 CE | | OUT | CE | Send Mode Control word |
| 4104 | 3E 0A | | MVI | A, 0A | LSB of count |
| 4106 | D3 CA | | OUT | CA | Write count to register |
| 4108 | 3E 00 | | MVI | A, 00 | MSB of count |
| 410A | D3 CA | | OUT | CA | Write count to register |
| 410C | 76 | | HLT | | |

Mode 3 Square wave generator:

Program:

| Address | Opcode | Label | Mnemonic | Operands | Comments |
|---------|--------|--------|----------|----------|-------------------------|
| 4100 | 3E 36 | START: | MVI | A, 36 | Channel 0 in mode 3 |
| 4102 | D3 CE | | OUT | CE | Send Mode Control word |
| 4104 | 3E 0A | | MVI | A, 0A | LSB of count |
| 4106 | D3 C8 | | OUT | C8 | Write count to register |
| 4108 | 3E 00 | | MVI | A, 00 | MSB of count |
| 410A | D3 C8 | | OUT | C8 | Write count to register |
| 410C | 76 | | HLT | | |

Mode 4: Software Triggered Strobe:

Program:

| Address | Opcode | Label | Mnemonic | Operands | Comments |
|---------|--------|--------|----------|----------|-------------------------|
| 4100 | | START: | MVI | A, 36 | Channel 0 in mode 0 |
| 4102 | | | OUT | CE | Send Mode Control word |
| 4104 | | | MVI | A, 0A | LSB of count |
| 4106 | | | OUT | C8 | Write count to register |
| 4108 | | | MVI | A, 00 | MSB of count |
| 410A | | | OUT | C8 | Write count to register |
| 410C | | | MVI | A, B8 | Channel 2 in Mode 4 |
| 410E | | | OUT | CE | Send Mode control Word |
| 4110 | | | MVI | A, 98 | LSB of Count |
| 4112 | | | OUT | CC | Write Count to register |
| 4114 | | | MVI | A, 3A | MSB of Count |
| 4116 | | | OUT | CC | Write Count to register |
| 4118 | | | HLT | | |

Set the jumper, so that the clock 0 of 8253 is given a square wave of frequency 1.5 MHz. This program divides this PCLK by 10 and thus the output at channel 0 is 150 KHz.

Vary the frequency by varying the count. Here the maximum count is FFFF H. So, the square wave will remain high for 7FFF H counts and remain low for 7FFF H counts. Thus with the input clock frequency of 1.5 MHz, which corresponds to a period of 0.067 microseconds, the resulting square wave has an ON time of 0.02184 microseconds and an OFF time of 0.02184 microseconds.

To increase the time period of square wave, set the jumpers such that CLK2 of 8253 is connected to OUT 0. Using the above-mentioned program, output a square wave of frequency 150 KHz at channel 0. Now this is the clock to channel 2.

Mode 4: Software Triggered Strobe:

The output is high after mode is set and also during counting. On terminal count, the output will go low for one clock period and becomes high again. This mode can be used for interrupt generation.

The following program initializes channel 2 of 8253 in mode 4.

Example:

Connect OUT 0 to CLK 2 (jumper J1). Execute the program and observe the output OUT 2. Counter 2 will generate a pulse after 1 second.

Mode 5 Hardware triggered strobe:

Counter starts counting after rising edge of trigger input and output goes low for one clock period when terminal count is reached. The counter is retriggerable.

Example:

The program that follows initializes channel 0 in mode 5 and also triggers Gate 0. Connect CLK 0 to debounce circuit.

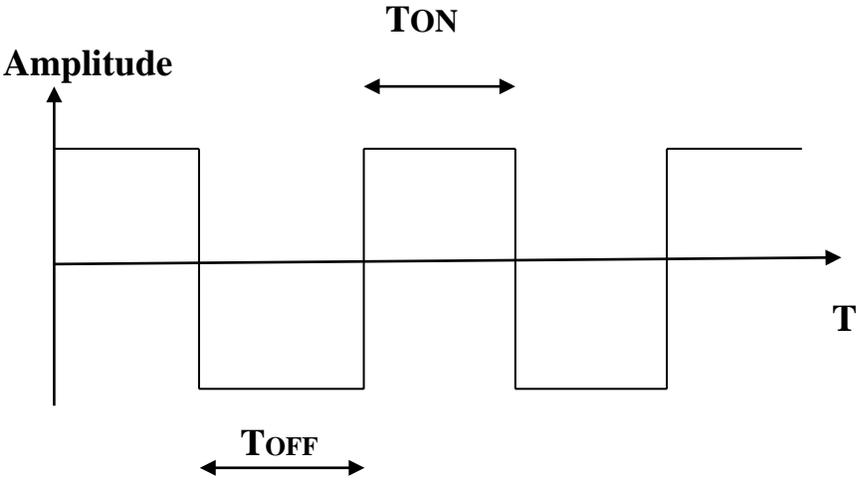
Execute the program. After giving Six clock pulses, you can see using CRO, the initially HIGH output goes LOW. The output (OUT 0 pin) goes high on the next clock pulse.

Mode 5 Hardware triggered strobe:

Program:

| Address | Opcode | Label | Mnemonic | Operands | Comments |
|---------|--------|--------|----------|----------|-------------------------|
| 4100 | | START: | MVI | A, 1A | Channel 0 in mode 5 |
| 4102 | | | OUT | CE | Send Mode Control word |
| 4104 | | | MVI | A, 05 | LSB of count |
| 4106 | | | OUT | C8 | Write count to register |
| 4108 | | | MVI | A, 00 | MSB of count |
| 410A | | | OUT | D0 | Trigger Gate 0 |
| 410C | | | HLT | | |

**OUTPUT:
 SQUARE WAVE:**



Observation:

TON:
 TOFF:
 $T = T_{ON} + T_{OFF}$
 FREQUENCY = $1/T$

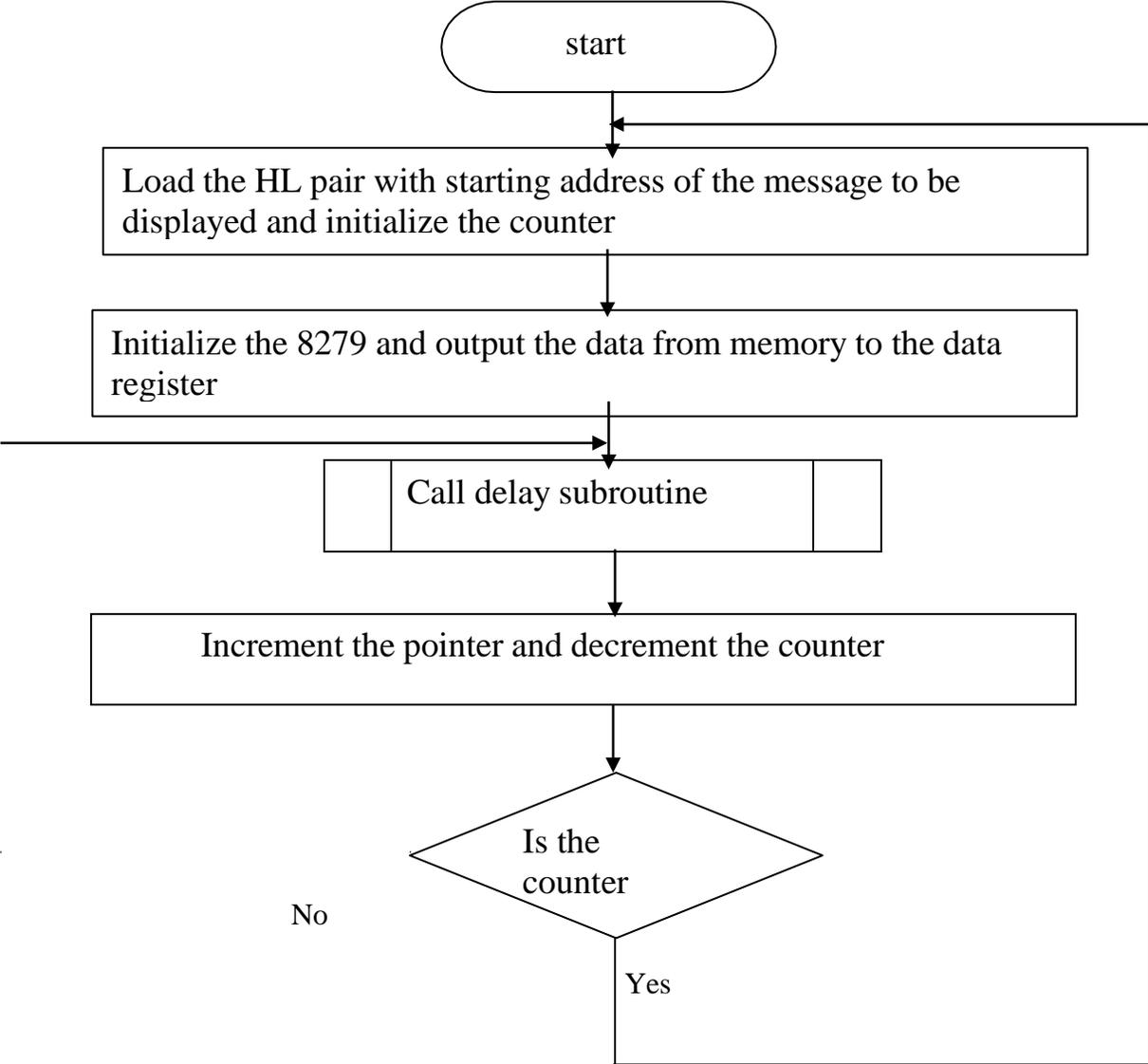
RESULT:

Thus the 8253 has been interfaced to 8085 μ p and six different modes of 8253 have been studied.

VIVA QUESTIONS:

1. If the crystal oscillator is operating at 15 MHz, what is the PCLK output of 8283 ?
2. By what factor does the 8284A clock generator divide the crystal oscillator's output frequency ?
3. What is number of counters that are present in the programmable timer device 8254 ?
4. What is the operation that can be performed on control word register?
5. What is the mode that is used to interrupt the processor by setting a suitable terminal count?
6. In mode 2, if N is loaded as the count value, then after (N-1) cycles, the output becomes low for how many cycles ?
7. In which mode the generation of square wave is possible ?
8. In control word register, if SC1=0 and SC0=1, then the counter selected is what?
9. In control word format, if RL1=1, RL0=1 then the operation performed is what?10. If BCD=0, then the operation is what in 8253?

FLOW CHART:



7 INTERFACING 8279 – KEYBOARD / DISPLAY CONTROLLER WITH 8085 MICROPROCESSOR

AIM:

To write an assembly language program to interface keyboard and display controller IC 8279 with 8085 microprocessor.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board.and 8279 kit.

ALGORITHM:

1. Start
2. Initialize the accumulator.
3. Load the appropriate command words in control register.
4. Load the appropriate words for the characters to be displayed, in the accumulator in analog form.
5. Send the converted data to 8279 display.
6. Stop.

PROGRAM:

| MEMO RY | OPCODE | LABEL | MNEMONICS | COMMENTS |
|------------|--------|-------|-------------|--|
| 4100 | | JMP1 | LXI H, 412C | Load pointer address to HL register |
| 4103 | | | MVI D,0F | Move look up table count value to D register |
| 4105 | | | MVI A,10 | Set display mode to right entry |
| 4107 | | | OUT C2 | Load control word into control register |
| 4109 | | | MVI A,CC | Clear Display RAM |
| 410B | | | OUT C2 | Load control word into control register |
| 410D | | | MVI A,90 | Read data to display memory |
| 410F | | | OUT C2 | Load control word into control register |
| 4111 | | LOOP | MOV A,M | Move HL memory content to accumulator |
| 4112 | | | OUT C0 | Move to LED segment register |
| 4114 | | | CALL DELAY | Move program control to 411F-delay loop |
| 4117 | | | INX H | Increment HL by one memory location |
| 4118 | | | DCR D | Decrement count value of look up table |
| 4119 | | | JNZ LOOP | Jump to 4111 till count value reaches zero |
| 411C | | | JMP JMP1 | Display the look up table values continually |
| 411F | | DELAY | MVI B, 0A | Load outer delay count value to B register |
| 4121 | | JMP3 | MVI C,FF | Load inner delay count value to C register |
| 4123 | | JMP2 | DCR C | Decrement C register content |
| 4124 | | | JNZ JMP2 | Jump to 4123 till count value reaches zero |
| 4127 | | | DCR B | Decrement B register content |
| 4128 | | | JNZ JMP3 | Jump to 4121 till count value reaches zero |
| 412B | | | RET | Return to main program |

LOOK UP TABLE:

| MEMORY | INPUT | MEMORY | INPUT |
|--------|-------|--------|-------|
| 412C | FF | 4134 | FF |
| 412D | FF | 4135 | 98 |
| 412E | FF | 4136 | 68 |
| 412F | FF | 4137 | 7C |
| 4130 | FF | 4138 | C8 |
| 4131 | FF | 4139 | FF |
| 4132 | FF | 413A | FF |
| 4133 | FF | 413B | FF |

OUTPUT: HELP US

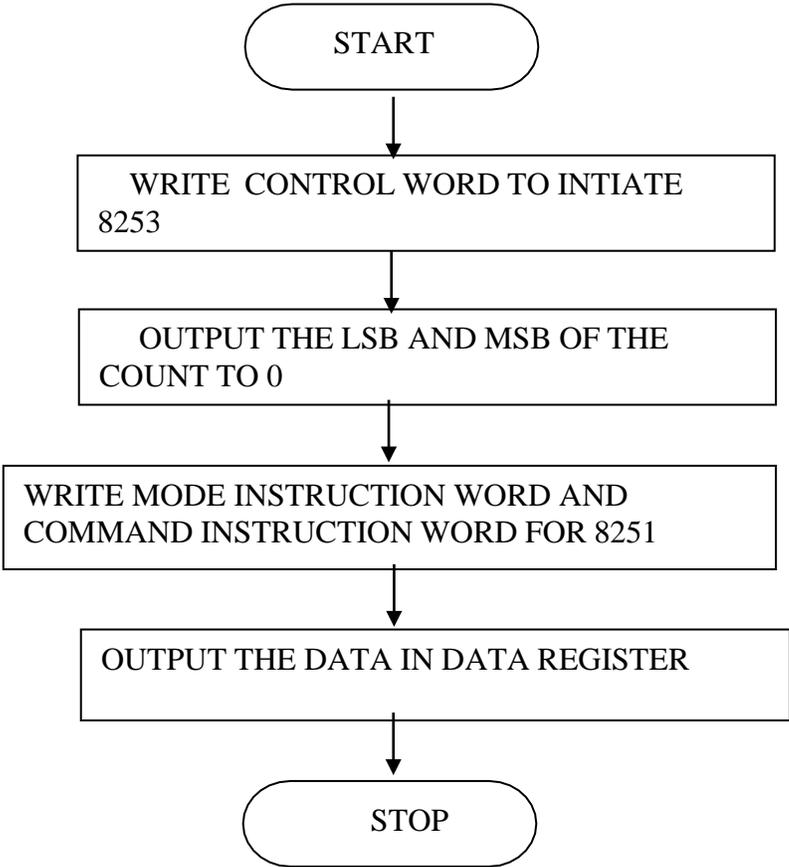
RESULT:

Thus an assembly language program for rolling display was written and is executed and verified by interfacing 8279 with 8085 microprocessor.

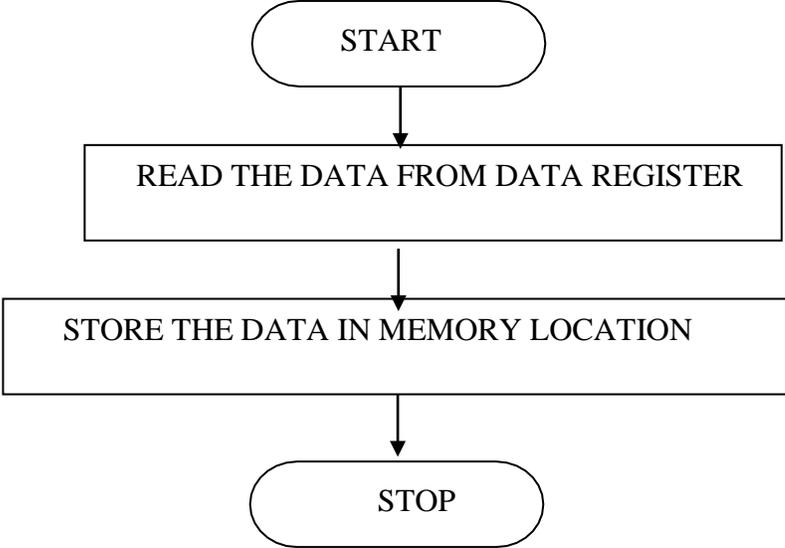
VIVA QUESTIONS:

1. What are the registers that store the keyboard and display modes and operations programmed by CPU ?
2. The sensor RAM acts as 8-byte first-in-first-out RAM in which mode?
3. What are the registers that holds the address of the word currently being written by the CPU from the display RAM ?
4. When a key is pressed, a debounce logic comes into operation in _____
5. What is the mode that is programmed using “end interrupt/error mode set command” ?
6. When a key is pressed, the debounce circuit waits for 2 keyboard scans and then checks whether the key is still depressed in _____
7. In which mode the data that is entered from the left side of the display
8. The FIFO status word is used to indicate the error in which mode?
9. What is the flag that increments automatically after each read or write operation to the display RAM is
11. If any change in sensor value is detected at the end of a sensor matrix scan, then the IRQ line goes _____

FLOW CHART-TRANSMITTER SIDE



RECIEVER SIDE



8 INTERFACING 8251 USART TO 8085 MICROPROCESSOR

AIM

To write an assembly language program to interface 8251 with 8085 Microprocessor and to transmit and receive data serially through 8251.

APPARATUS REQUIRED:

8085 microprocessor kit ,key board and8251 kit.

ALGORITHM

1. To Transmit Data serially, initialize a counter.
2. Load the 16-bit count in the counter.
3. Load the control word in 8251.
4. Send the data to 8251.
5. Halt.
6. To receive data serially, get the data from 8251.
7. Save the data in memory.
8. Halt

INTERFACING 8251 USART

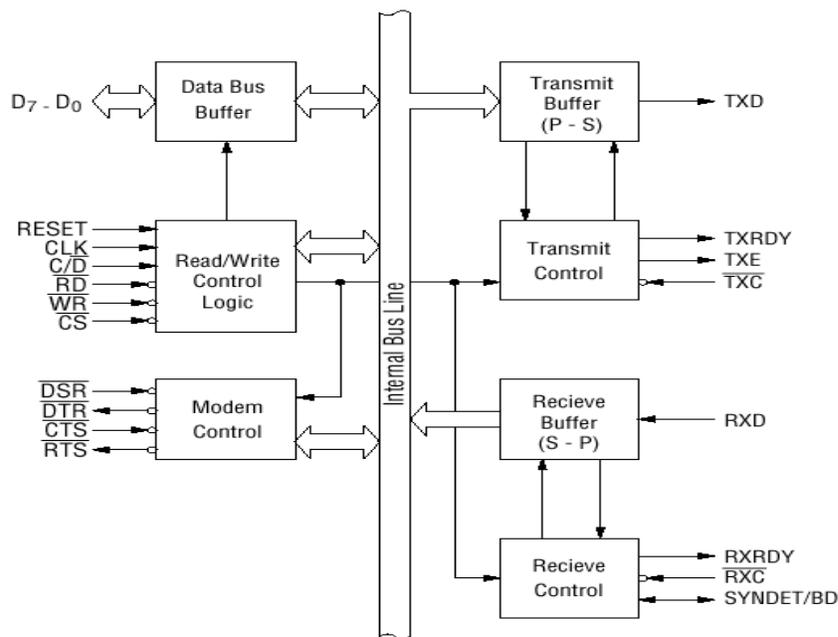


Fig.1 Block Diagram of 8251

PROGRAM
Transmitter Side

| MEMORY | OPCODE | MNEMONICS | COMMENTS |
|---------------|---------------|------------------|---|
| 4100 | | MVI A,36H | Configure 8253 to count binary,mode3, read / load LSB first then MSB, counter 0 |
| 4102 | | OUT CE | Load control word to control register of 8253 |
| 4104 | | MVI A, 0AH | Move lower 8 bit count value to accumulator |
| 4106 | | OUT C8H | Load count to channel 0 of 8253 |
| 4108 | | MVI A,00H | Move upper 8 bit count value to accumulator |
| 410A | | OUT C8H | Load count to counter 0 of 8253 |
| 410C | | MVI A, 4E | Configure mode instruction of 8251 |
| 410E | | OUT C2H | Load control word to control register of 8251 |
| 4110 | | MVI A,37 | Configure command instruction of 8251 |
| 4112 | | OUT C2 | Load control word to control register of 8251 |
| 4114 | | MVI A,41H | Move serial data to be transmitted |
| 4116 | | OUT C0H | Load the serial data to data register |
| 4118 | | HLT | Stop the process |

Receiver Side

| MEMORY | OPCODE | MNEMONICS | COMMENTS |
|---------------|---------------|------------------|--|
| 4200 | | IN C0 | Get the serial data from data register |
| 4202 | | STA 4150 | Store in 4150 memory location of 8085 |
| 4205 | | HLT | Stop the process |

Bit Configuration of Mode Instruction-Asynchronous

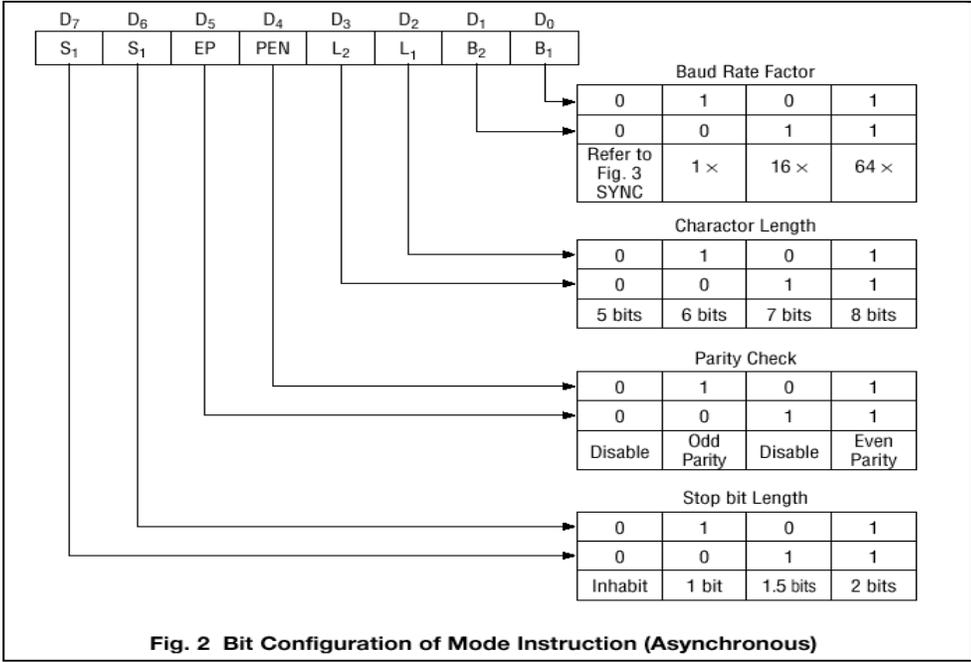


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

Bit Configuration of Mode Instruction -Synchronous

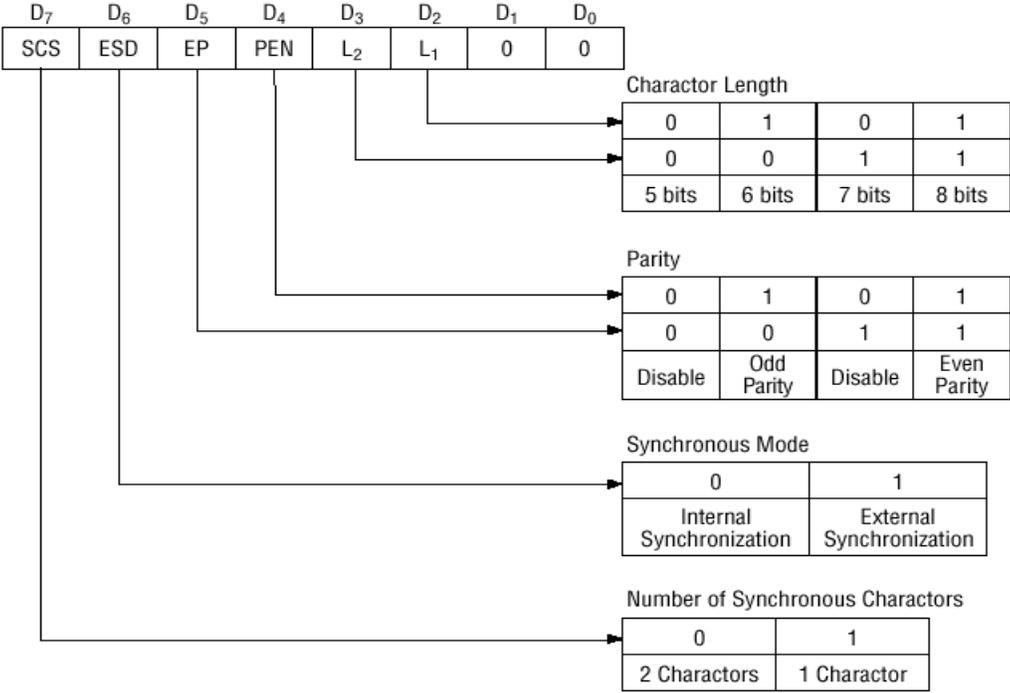
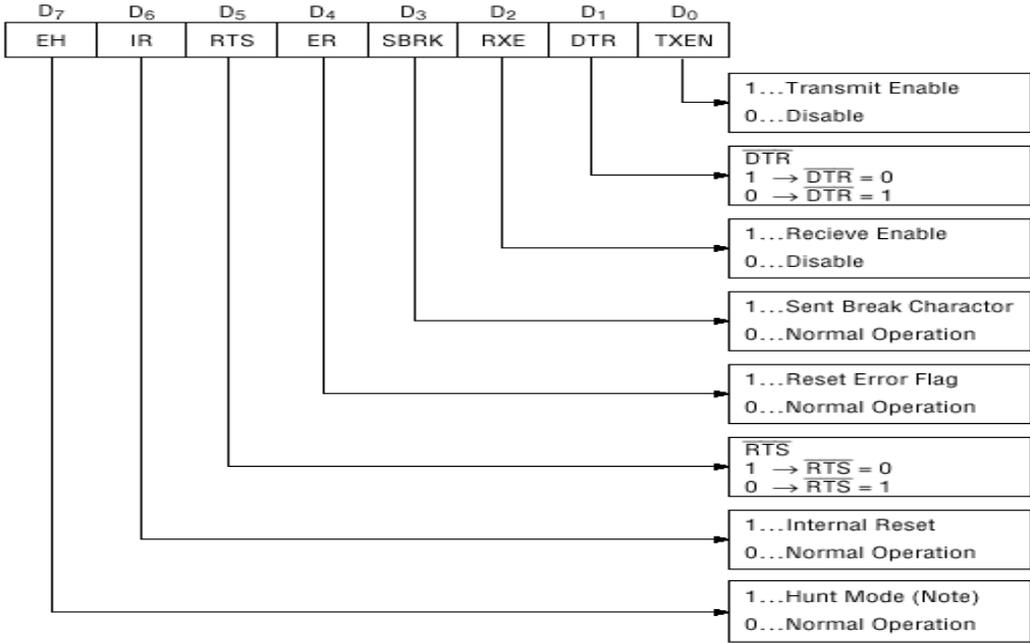


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

Bit Configuration of Mode Instruction-Command



Note: Search mode for synchronous characters in synchronous mode.

Fig. 4 Bit Configuration of Command

OUTPUT

| Memory address | Output value |
|----------------|--------------|
| 4150 | |
| | |

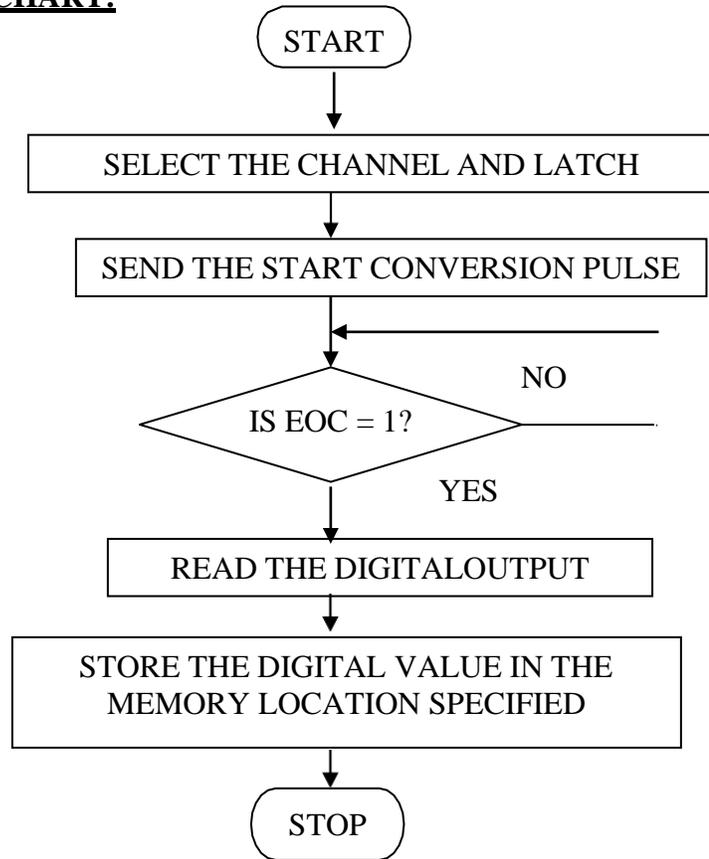
RESULT:

Thus an assembly language program was written to interface 8251 with 8085 microprocessor. It was executed and the output was verified.

VIVA QUESTIONS:

1. What is meant by 8251?
2. Which of the following is not a mode of data transmission? _____
a) simplex b) duplex c) semi duplex d) half duplex
3. If the data is transmitted only in one direction over a single communication channel, then it is of _____ mode.
4. If the data transmission takes place in either direction, but at a time data may be transmitted only in one direction then, it is of _____ mode.
5. In 8251A, what is the pin that controls the rate at which the character is to be transmitted.
6. TXD(Transmitted Data Output) pin carries serial stream of the transmitted data bits along with _____
7. The signal that may be used either to interrupt the CPU or polled by the CPU is _____
8. What are the disadvantage of RS-232C?
9. The USB supports the signaling rate of _____
10. What is the bit packet that commands the device either to receive data or transmit data in transmission of USB asynchronous communication?

FLOW CHART:



9 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:

| ADDRESS | LABEL | OPCODE | PROGRAM | 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:

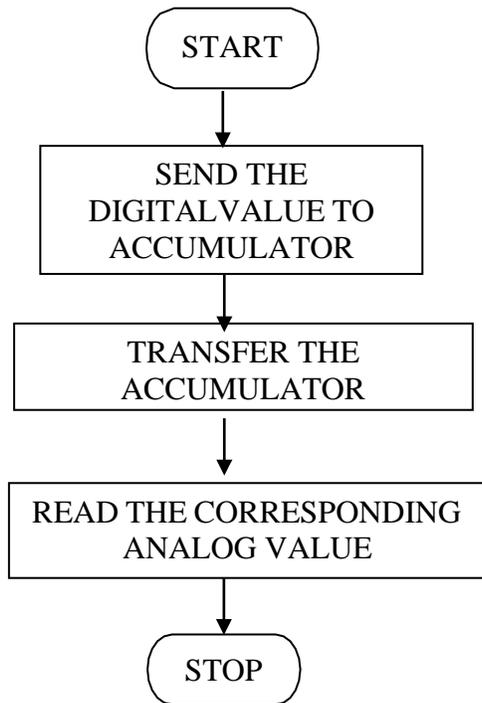
Thus the ADC was interfaced with 8085 and the given analog inputs were converted into its digital equivalent.

VIVA QUESTIONS:

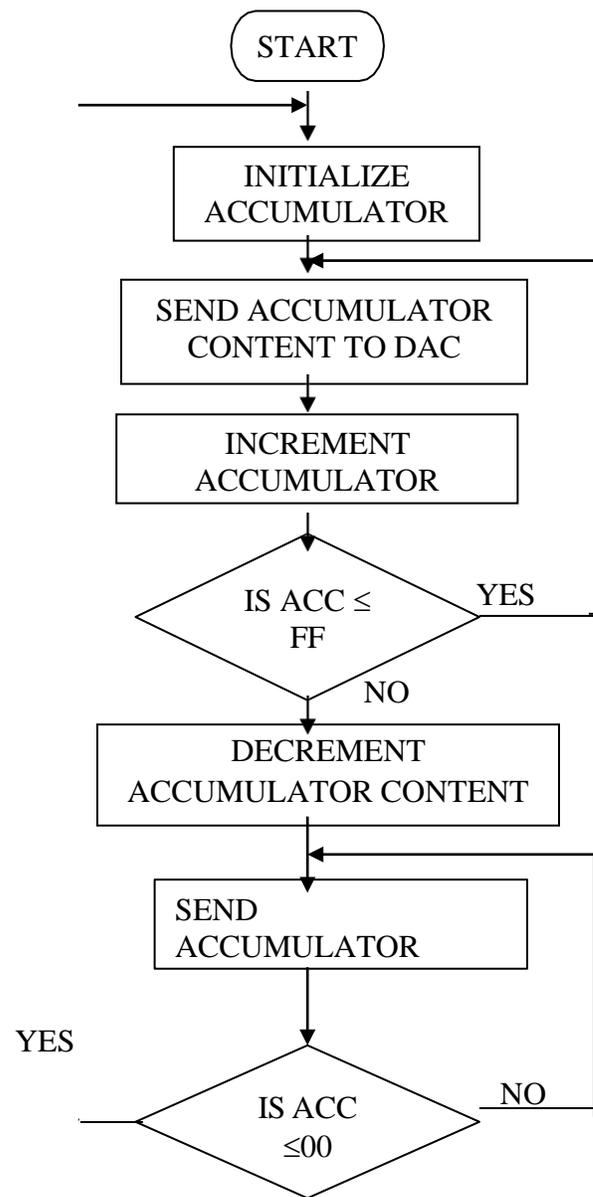
1. What is the name given to time taken by the ADC from the active edge of SOC(start of conversion) pulse till the active edge of EOC(end of conversion) signal ?
2. What are the popular technique that is used in the integration of ADC chips ?
3. The procedure of algorithm for interfacing ADC contain_____.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

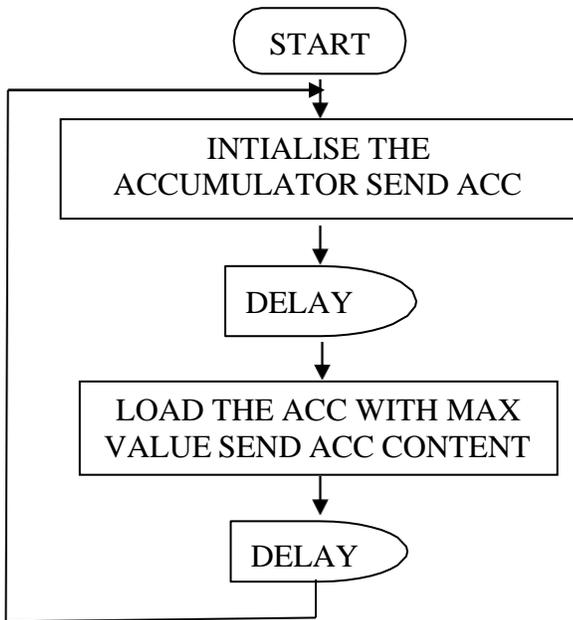
MEASUREMENT OF ANALOG VOLTAGE



TRIANGULAR WAVE FORM



SQUARE WAVE FORM



10 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. 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:

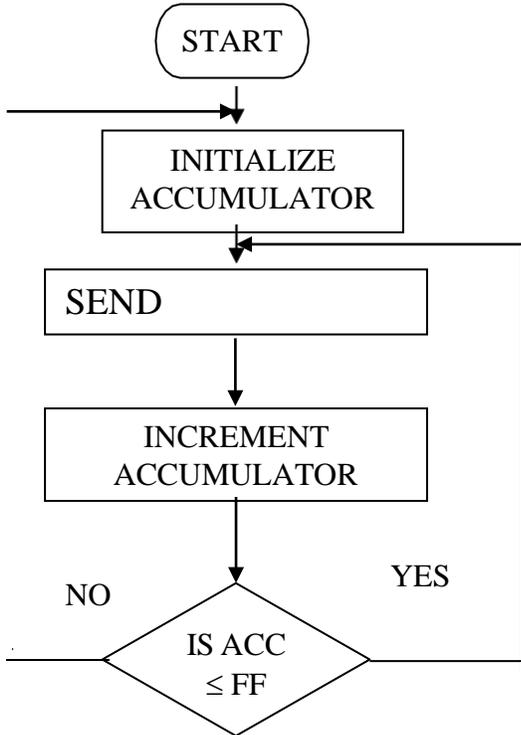
Square Waveform:

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



PROGRAM: 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 |

Triangular waveform:

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)

PROGRAM: Square Wave

| ADDRESS | 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 |

PROGRAM: Saw tooth Wave

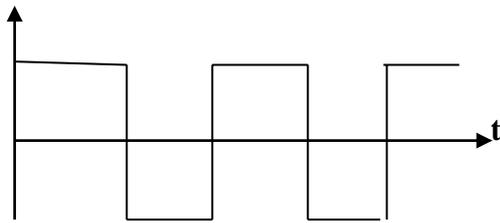
| ADDRESS | LABEL | PROGRAM | COMMENTS |
|---------|-------|-----------|--|
| 4100 | START | MVI A,00H | Load 00 in accumulator |
| 4102 | L1 | OUT C0 | Send through output port |
| 4103 | | INR A | Increment contents of accumulator |
| 4104 | | JNZ L1 | Send through output port until it reaches FF |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

PROGRAM: Triangular Wave

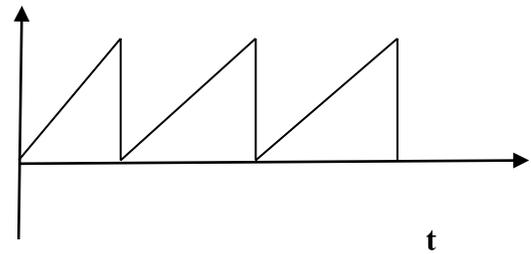
| ADDRESS | LABEL | PROGRAM | COMMENTS |
|---------|-------|-----------|--|
| 4100 | START | MVI L,00H | Load 00 in accumulator |
| 4102 | L1 | MOV A ,L | Move contents of L to A |
| 4103 | | OUT C8 | Send through output port |
| 4104 | | INR C | Increment contents of accumulator |
| 4105 | | JNZ L1 | 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:

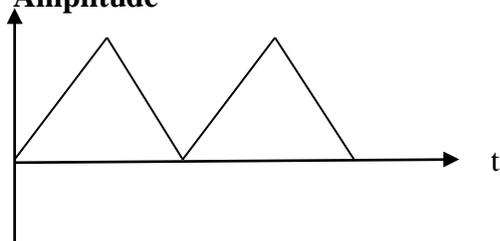
Square Waveform
Amplitude



Saw-tooth waveform
Amplitude



Triangular waveform
Amplitude



RESULT OF WAVEFORM GENERATION:

| WAVEFORMS | AMPLITUDE | TIMEPERIOD |
|--------------------------------|-----------|------------|
| <i>Square Waveform</i> | | |
| Saw-tooth waveform | | |
| Triangular waveform | | |

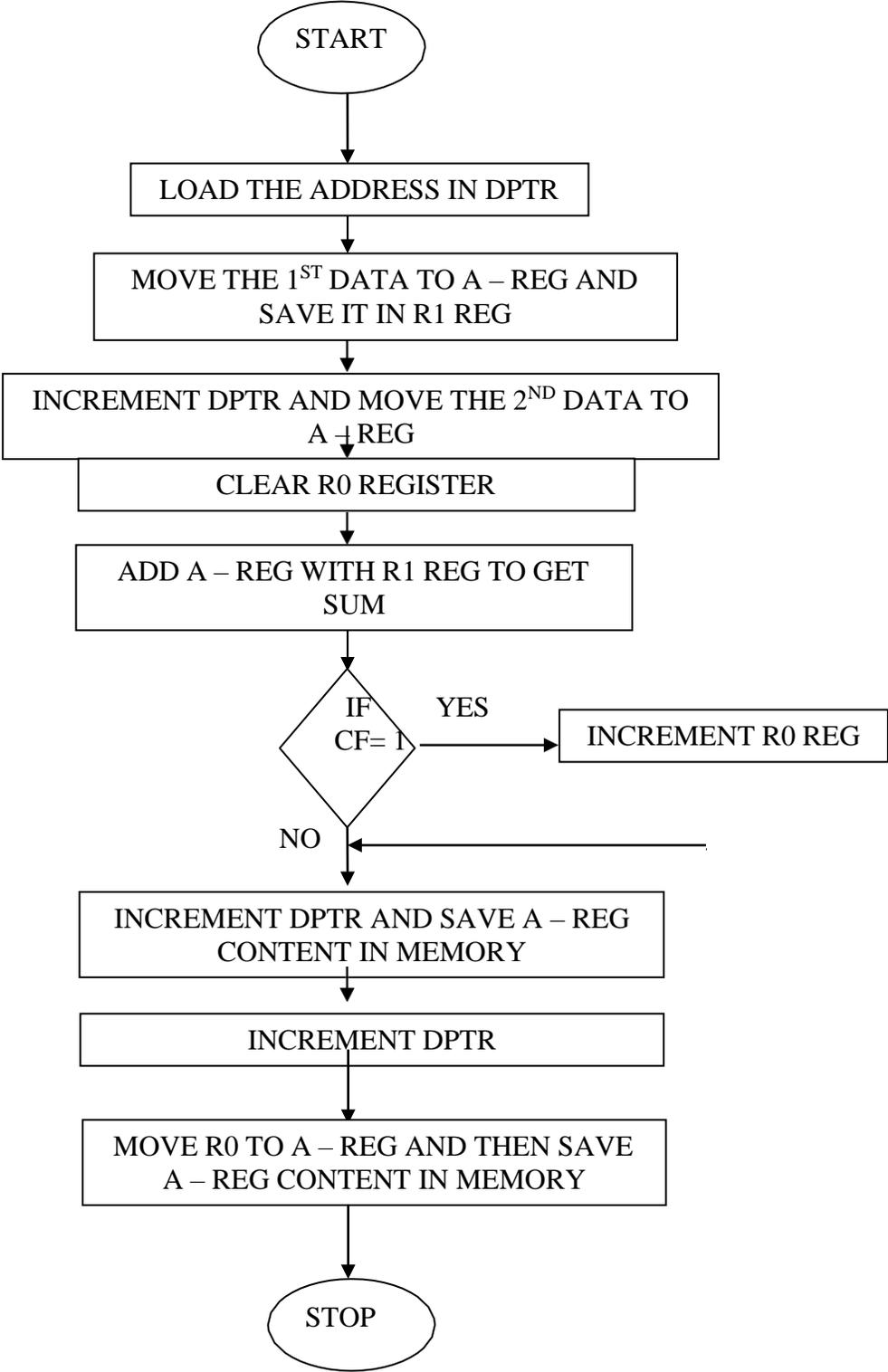
RESULT:

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?

FLOW CHART:



11(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 R₀ 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 |
| 4500 | |

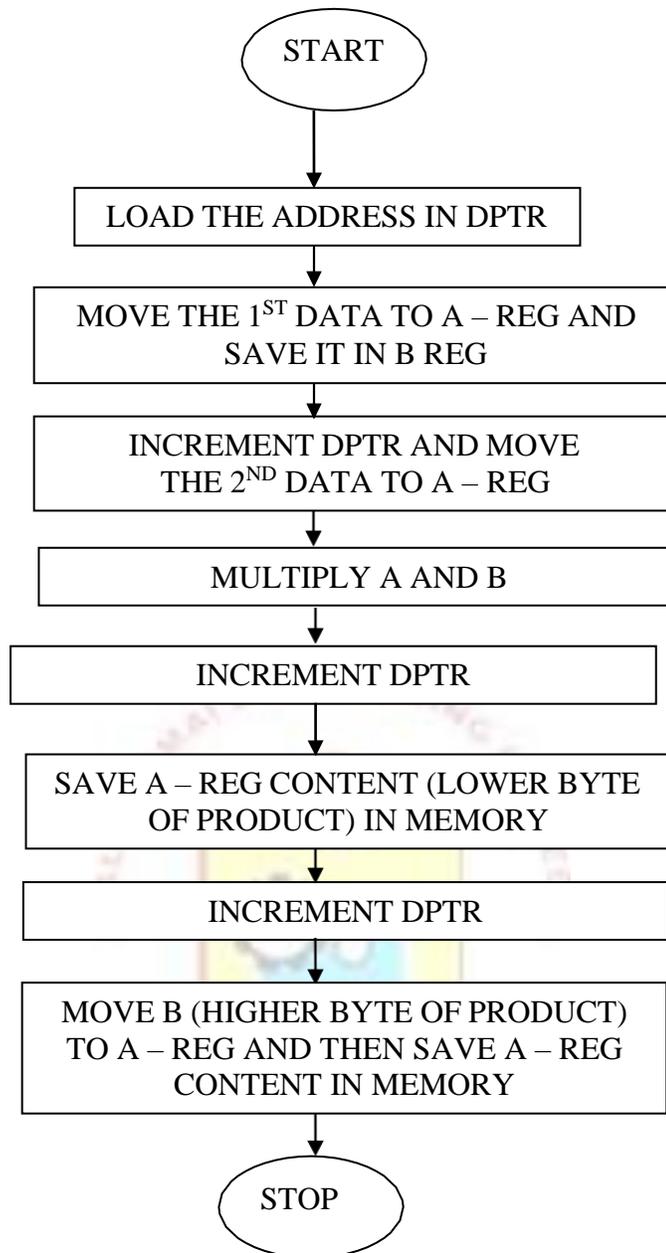
RESULT:

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:



11(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 |
| 4500 | 07 |

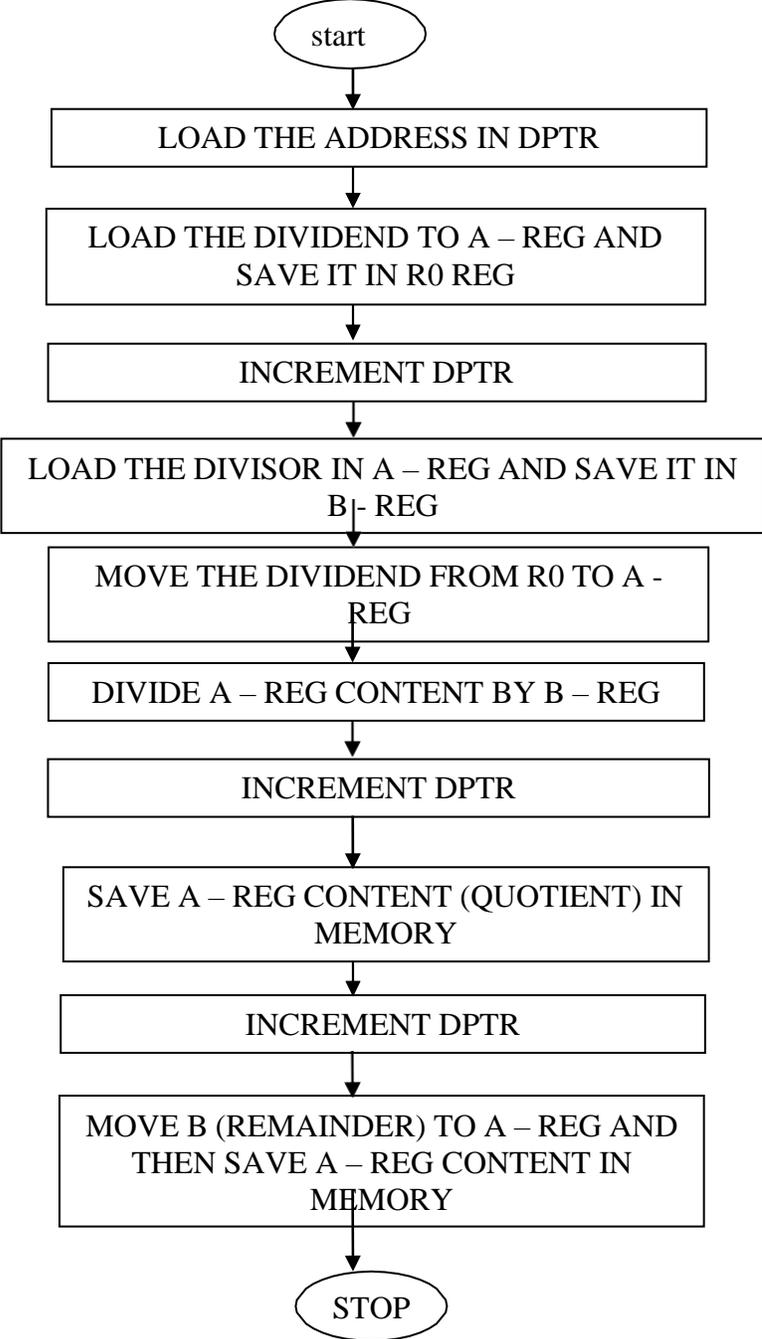
RESULT:

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 in 8051.
7. What is the SCON register in 8051?

FLOWCHART:



11(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 |

OBSERVATION:

| OUTPUT | |
|-----------------|------|
| MEMORY LOCATION | DATA |
| 4500 | |
| 4501 | |

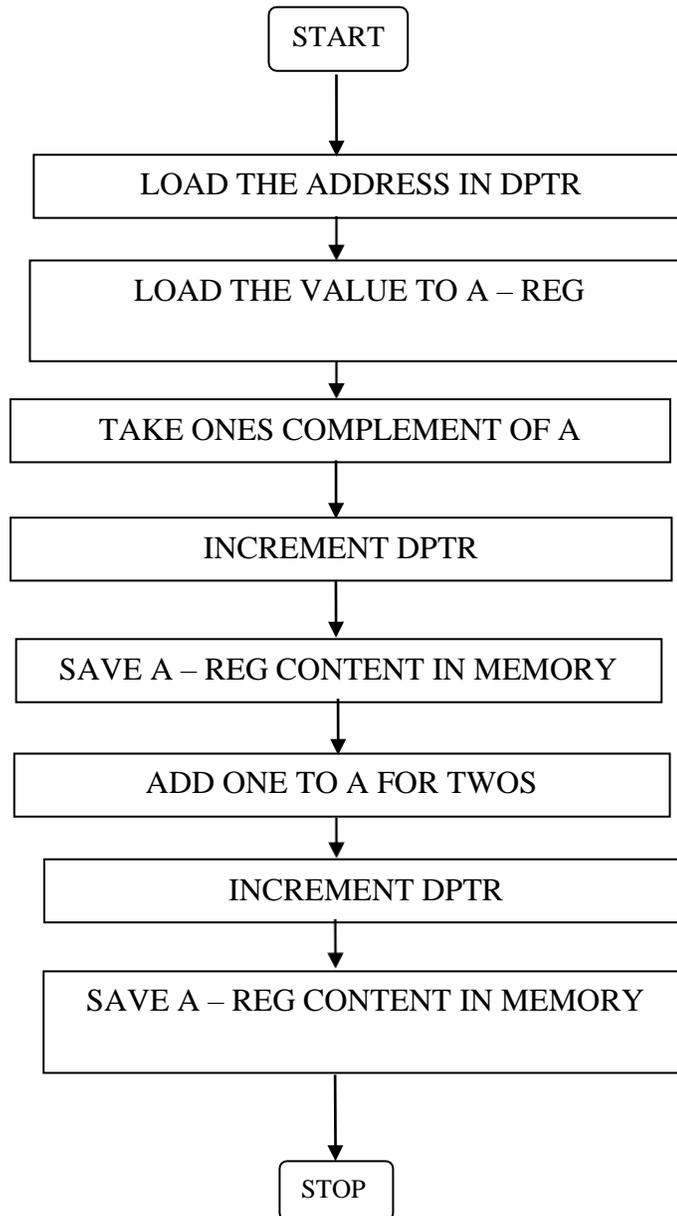
RESULT:

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:



1(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 |

OBSERVATION:

| OUTPUT | |
|------------------|------|
| MEMORY LOCATION | DATA |
| 4500 (remainder) | |
| 4501 (quotient) | |

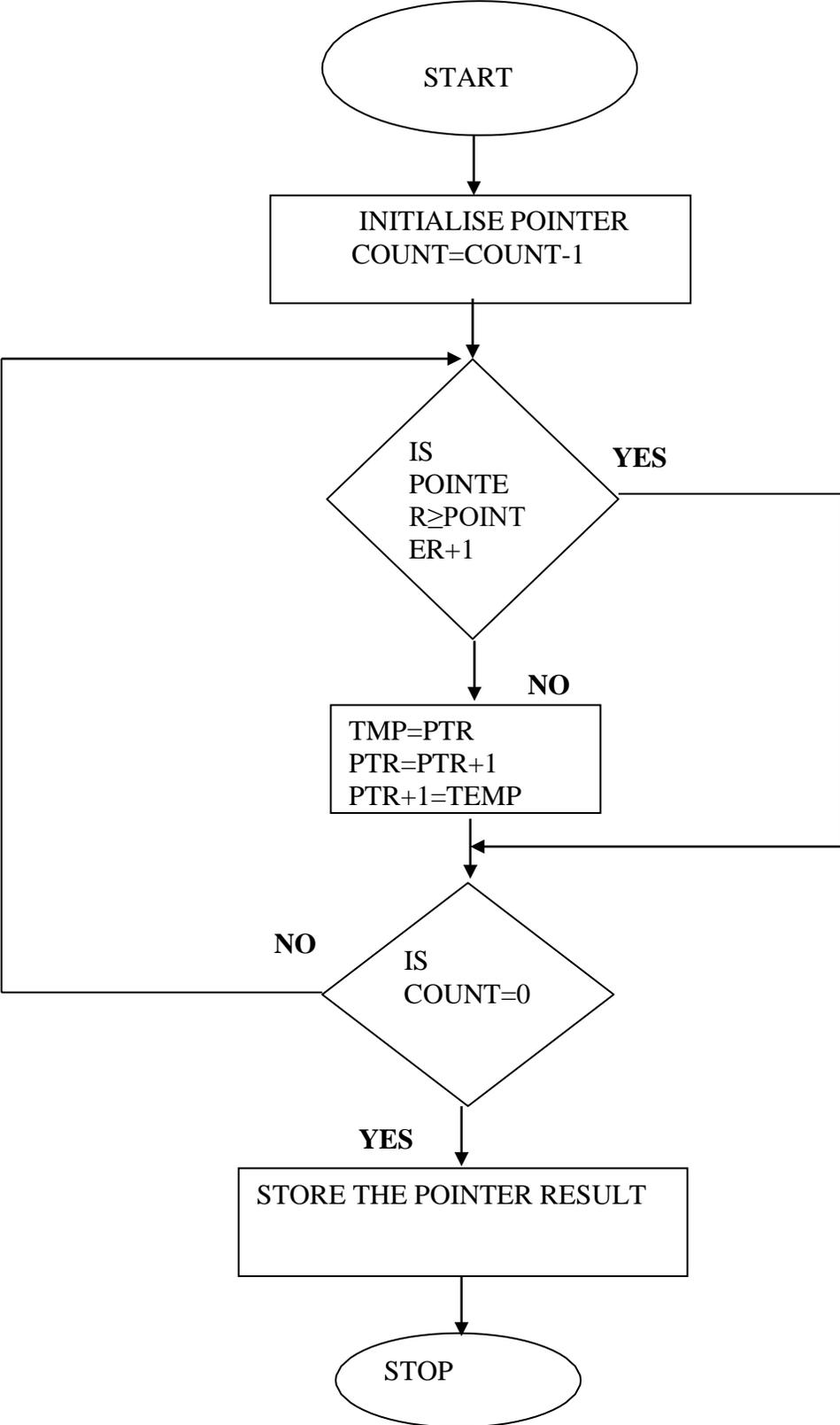
RESULT:

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:



12(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 to 16 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 to 16 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 | | 4207 | |
| 4201 | | | |
| 4202 | | | |
| 4203 | | | |
| 4204 | | | |
| 4205 | | | |
| 4206 | | | |

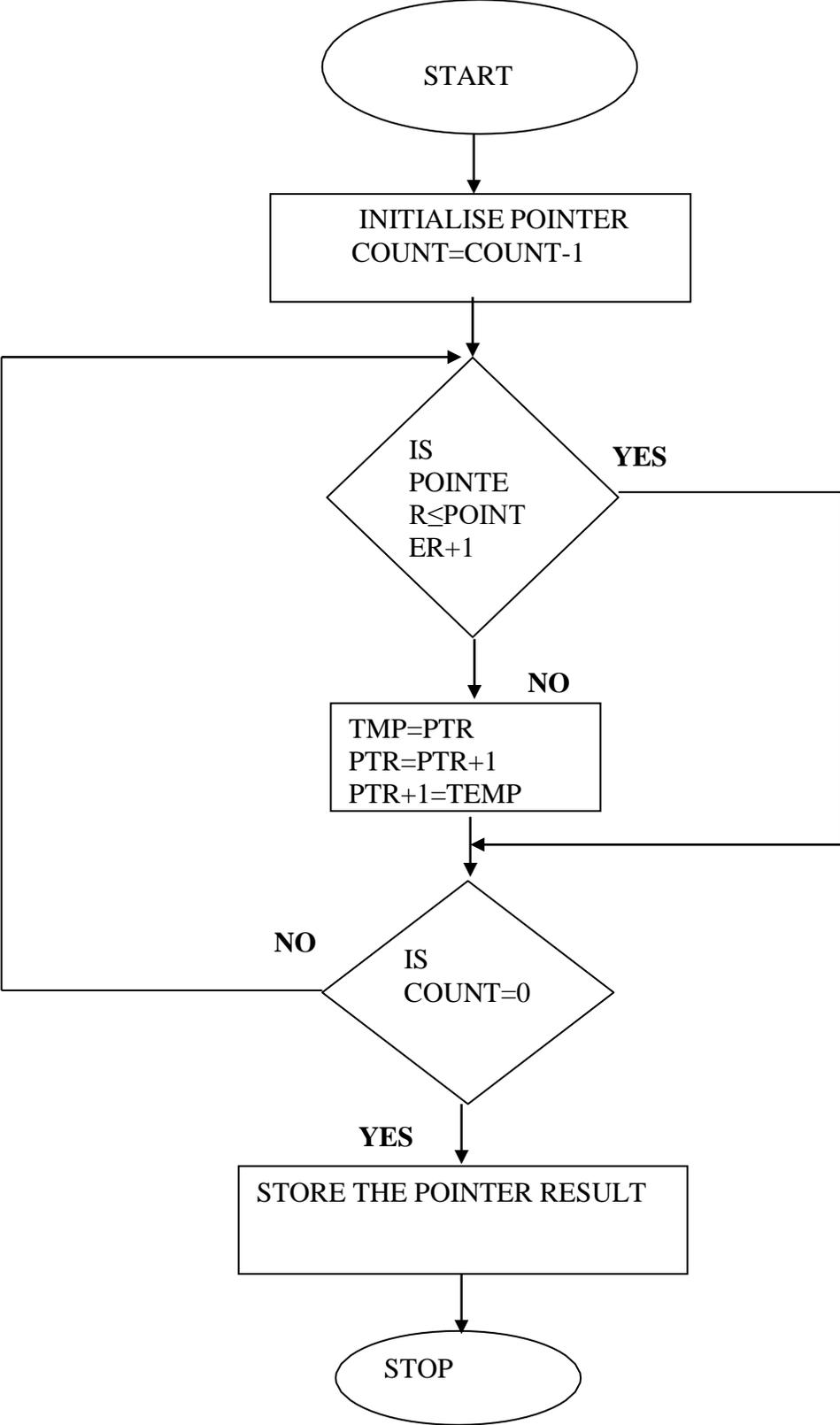
RESULT:

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:



12(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,#FF | Load zero to memory 40H |
| 4106 | | | MOV R5,#07 | Move Array size to R5 |
| 4108 | | LOOP2 | MOVX A,@DPTR | Accumulator is moved to 16 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 to 16 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 | | 4207 | |
| 4201 | | | |
| 4202 | | | |
| 4203 | | | |
| 4204 | | | |
| 4205 | | | |
| 4206 | | | |

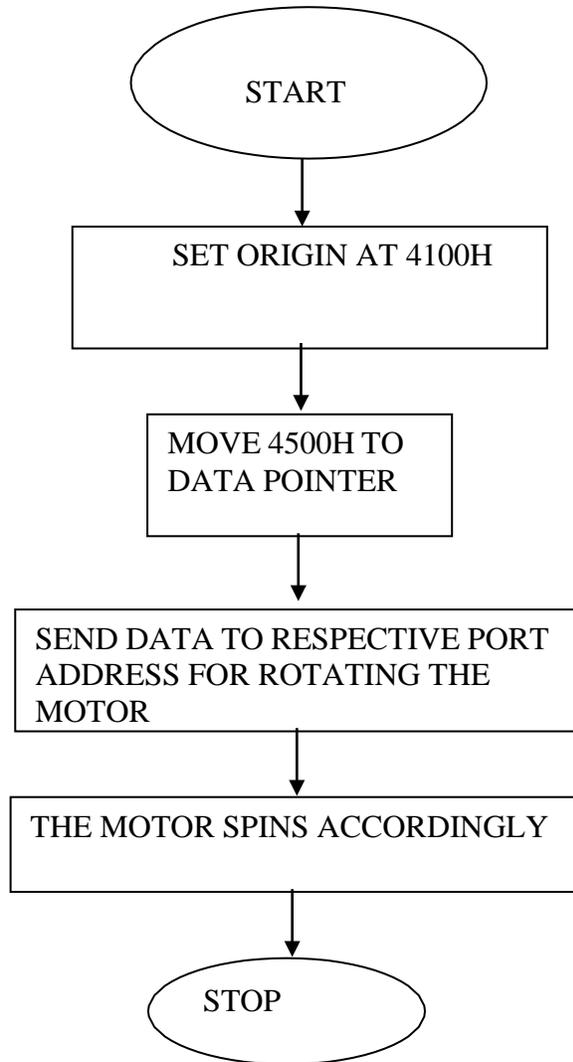
RESULT:

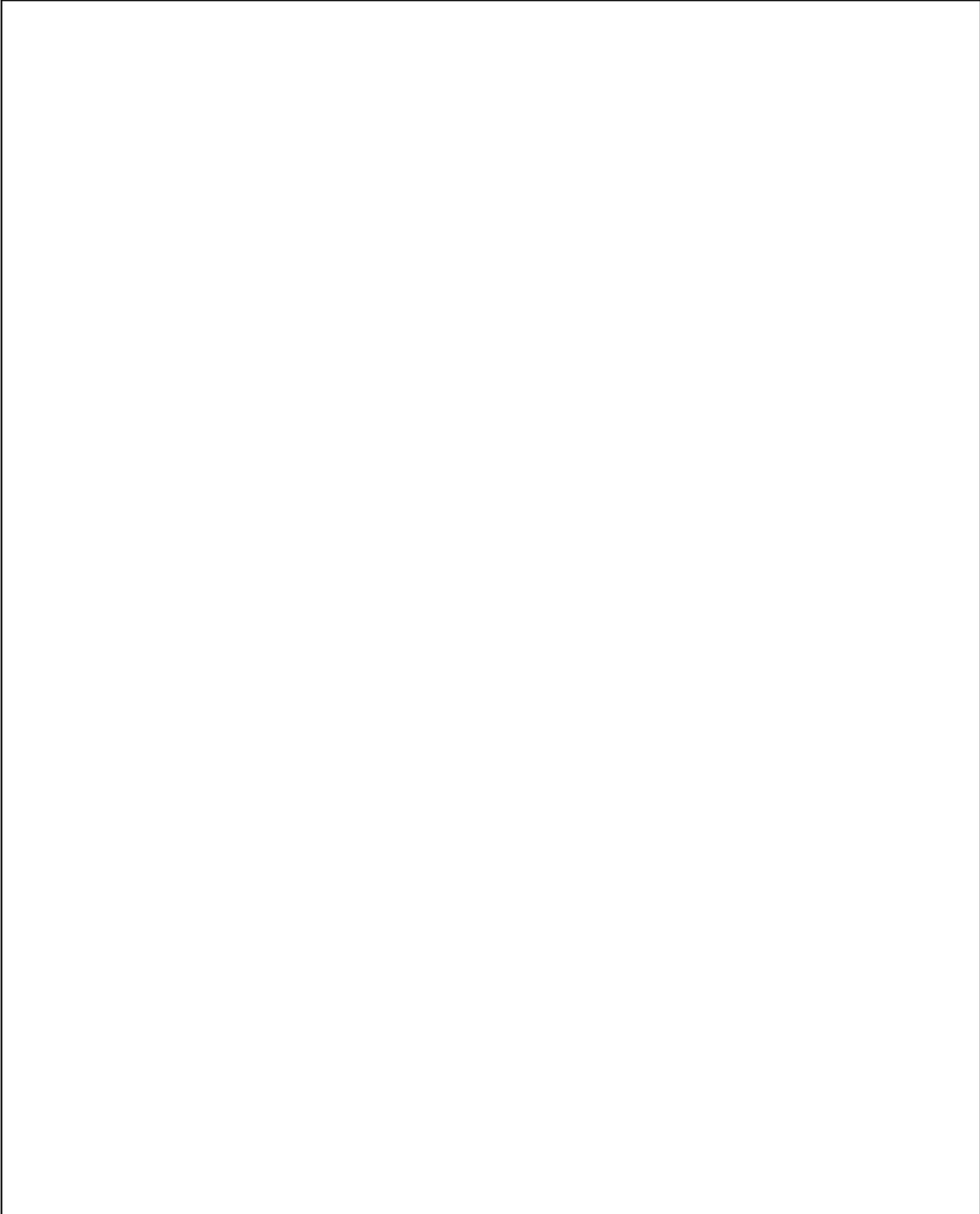
Thus an assembly language program written for finding the smallest element in an array was executed using 8051 microcontroller and the output was verified.

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?

FLOW CHART:





13 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:

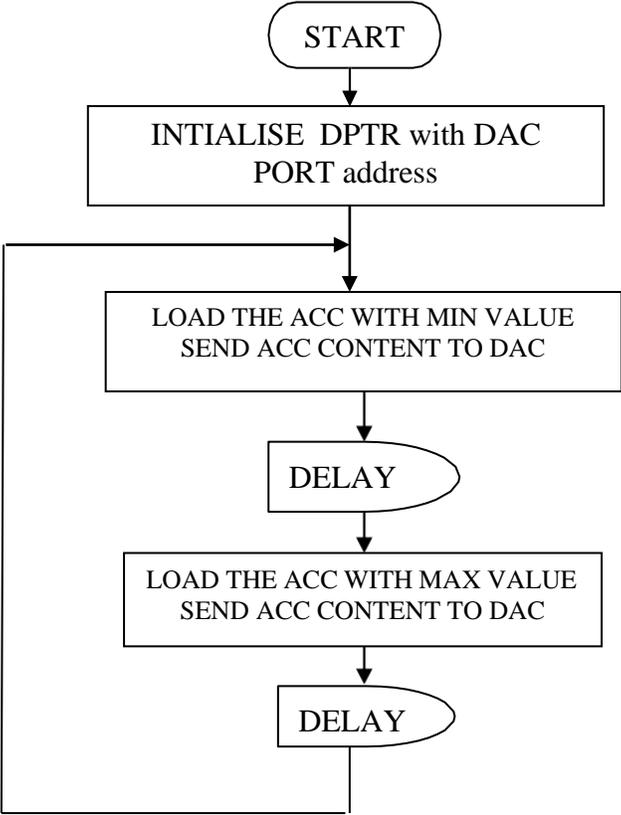
Thus a stepper motor was interfaced with 8051 and run in forward and reverse directions at various speeds.

VIVA QUESTIONS:

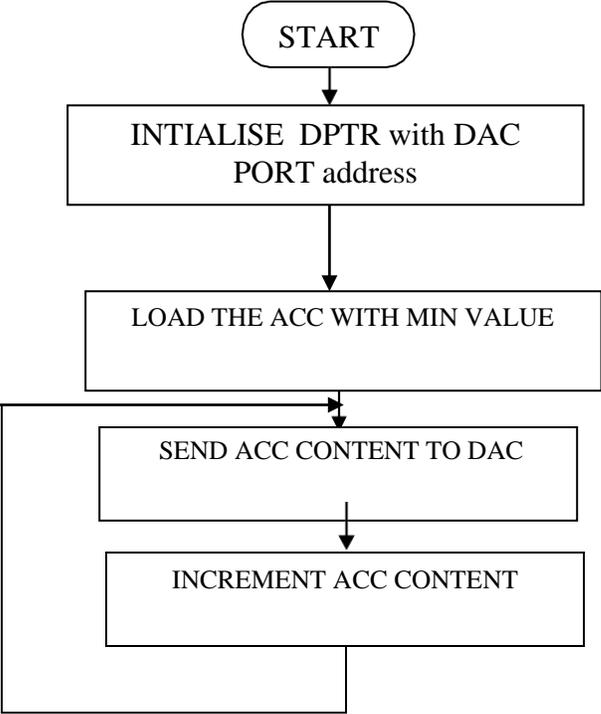
1. What are the application of stepper motor?
2. What is meant by step angle?
3. What are the methods to control the speed of stepper motor?
4. What is the formula for steps per revolution?
5. What is the use of DB instruction?
6. What is the use of PUSH and POP operation ?
7. How a stepper motor differs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

FLOWCHART:

SQUARE WAVE FORM:



SAWTOOTH WAVE FORM:



14 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

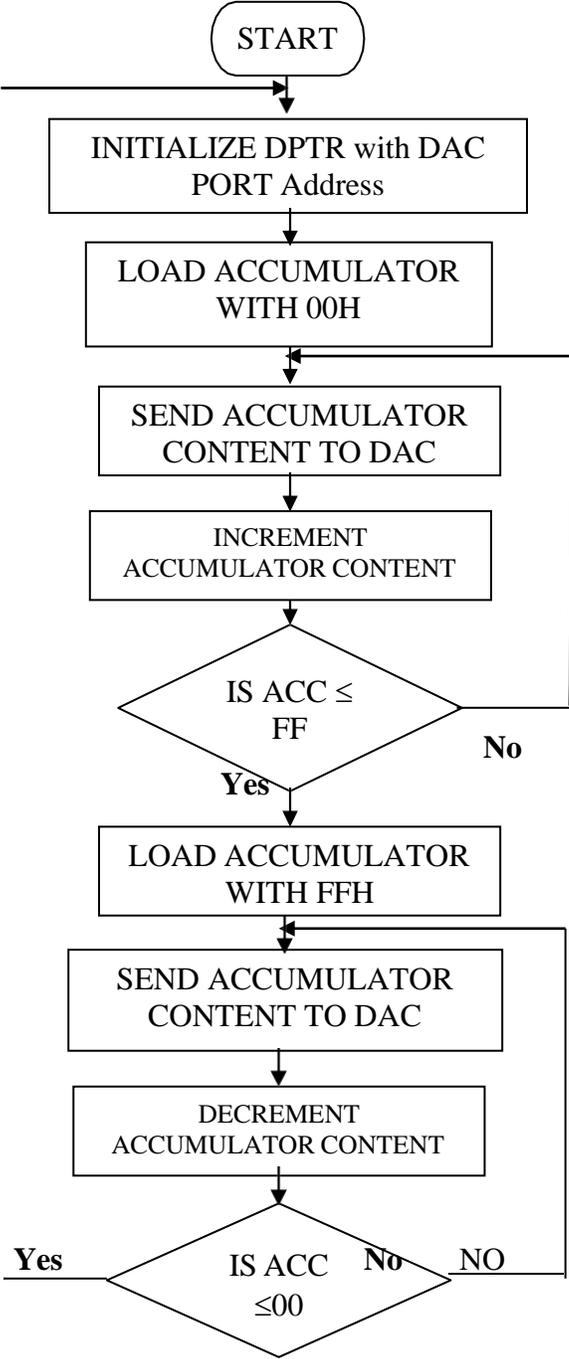
SAWTOOTH WAVE GENERATION:

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 | | |
| 4118 | | SJMP START | | Return and jump to 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?

15 INTERFACING OF LEDS AND SENSOR WITH ARDUINO /RASPBERRY PI MODULES

AIM:

To perform Interfacing of LEDs and sensor with arduino /raspberry pi modules

ARDUINO PROGRAMMING

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are:

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the microcontroller into a more accessible package.



Fig: Arduino Board

Here are the components that make up an Arduino board and what each of their functions are.

1. Reset Button – This will restart any code that is loaded to the Arduino board
2. AREF – Stands for “Analog Reference” and is used to set an external reference voltage
3. Ground Pin – There are a few ground pins on the Arduino and they all work the same
4. Digital Input/output – Pins 0-13 can be used for digital input or output
5. PWM – The pins marked with the (~) symbol can simulate analog output
6. USB Connection – Used for powering up your Arduino and uploading sketches
7. TX/RX – Transmit and receive data indication LEDs
8. ATmega Microcontroller – This is the brains and is where the programs are stored
9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source
10. Voltage Regulator – This controls the amount of voltage going into the Arduino board
11. DC Power Barrel Jack – This is used for powering your Arduino with a power supply
12. 3.3V Pin – This pin supplies 3.3 volts of power to your projects
13. 5V Pin – This pin supplies 5 volts of power to your projects
14. Ground Pins – There are a few ground pins on the Arduino and they all work the same
15. Analog Pins – These pins can read the signal from an analog sensor and convert it to digital

SENSOR PROGRAMMING AND INTERFACE

```
void setup ()  
{  
  // put your setup code here, to run once:  
}  
void loop ()  
{  
  // put your main code here, to run repeatedly:  
}
```

setup:

It is called only when the Arduino is powered on or reset. It is used to initialize variables and pin modes

loop:

The loop functions run continuously till the device is powered off. The main logic of the code goes here. Similar to while (1) for micro-controller programming.

PinMode

- A pin on arduino can be set as input or output by using pinMode function.

```
pinMode (13, OUTPUT); // sets pin 13 as output
```

```
pinMode (13, INPUT); // sets pin 13 as input
```

Reading/writing digital values

```
digitalWrite(13, LOW);
```

```
// Makes the output voltage on pin 13 , 0V
```

```
digitalWrite(13, HIGH);
```

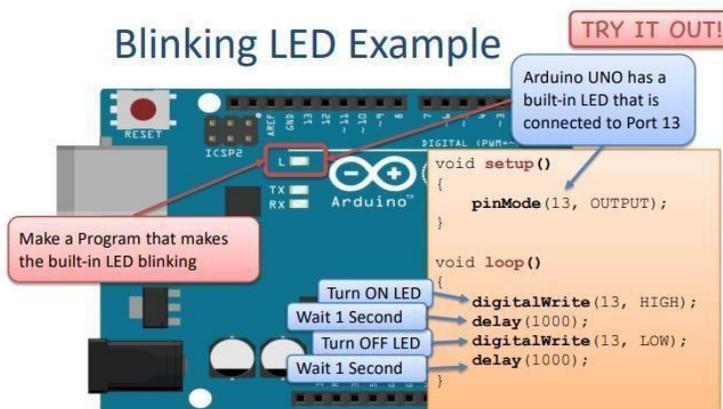
```
// Makes the output voltage on pin 13 , 5V
```

```
buttonState = digitalRead(2);
```

```
// reads the value of pin 2 in buttonState
```

PROGRAMMING

Blinking LED Example TRY IT OUT!



```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Turn ON LED → digitalWrite(13, HIGH);
Wait 1 Second → delay(1000);
Turn OFF LED → digitalWrite(13, LOW);
Wait 1 Second → delay(1000);

BLINKING OF LED IN ARDUINO BOARD

```
void setup()
{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

SERIAL PROGRAMMING USING ARDUINO:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  int i;
  for(i=0; i<10; i++)Serial.println(i);
}
```

OUTPUT

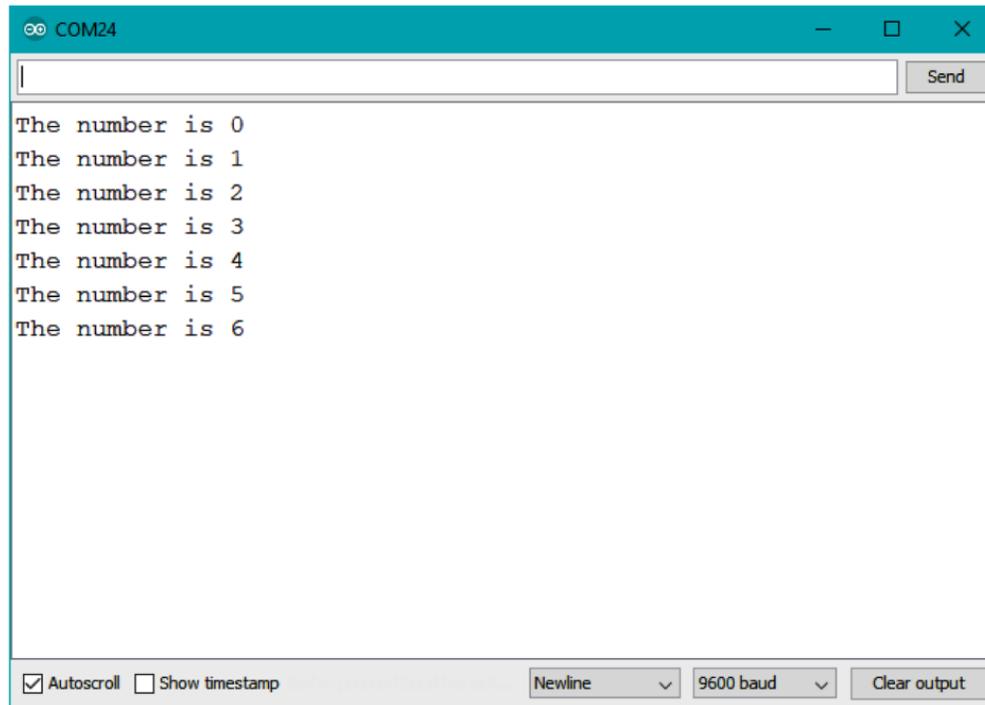


Fig: Serial Monitor output for serial programming

PROGRAMMING USING ULTRASONIC SENSOR INTERFACE

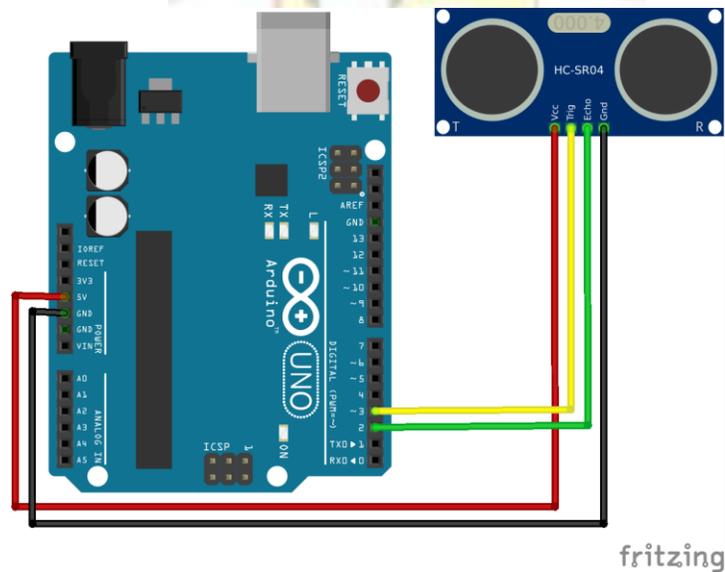


Fig: Ultrasonic Sensor Interface with Arduino

```
int trigPin= 9;
```

```
int echoPin= 10;void
setup ()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
void loop ()
{
  //Serial.println("loop"); long
  duration, distance;
  digitalWrite(trigPin,HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigPin, LOW);

  duration=pulseIn(echoPin, HIGH);
  distance =(duration/2)/29.1;
  Serial.print(distance);
  Serial.println("CM");

  delay(10);
}
```

PROGRAMMING USING IR SENSOR INTERFACE

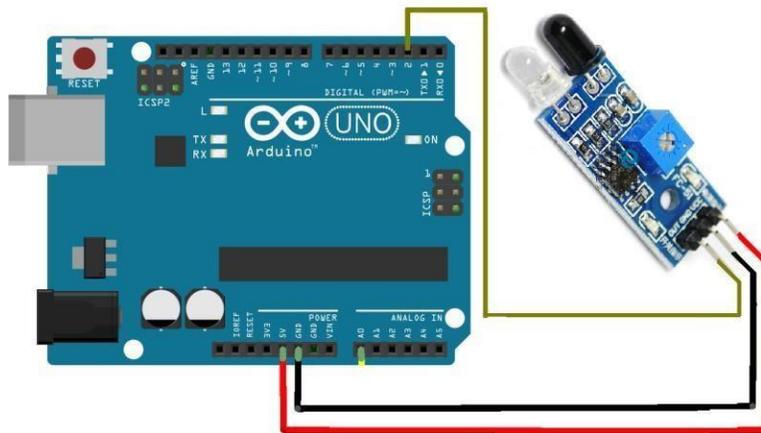


Fig:IR Sensor Interface with Arduino

```
#define s 7
#define led 13
void setup()
{
  pinMode(s,INPUT);
  pinMode(led, OUTPUT);

  // put your setup code here, to run once:
}
void on()
{
  digitalWrite(led,HIGH);
}
void off()
{
  digitalWrite(led,LOW);
}
```

```
void loop()
{
  if (digitalRead(s)==0)
  {
    on();
  }
  else
  {
    off();
  }
}
```

Result

Thus the Interfacing of LEDs and sensor with arduino /raspberry pi modules is done and the output is verified.

VIVA QUESTIONS:

1. What is Raspberry Pi?
2. How does the Raspberry Pi work?
3. Tell about any interesting project with Raspberry Pi.
4. How is Raspberry Pi used in IoT?
5. What are the different components of a Raspberry Pi board?
6. How many types of Arduino do we have?
7. What are the three important parts of Arduino *?
8. Why we should use Arduino?
9. What are the features of Arduino?

ADDITIONAL EXPERIMENTS

17 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 another and 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 square wave to be generated be F_s KHz.

And the time period of the square wave be T_s Sec.

Oscillator Frequency = 11.0592MHz.

One machine cycle = 12 clock periods

Time taken to complete one machine cycle = $12 * (1/11.0592\text{MHz}) = 1.085\text{microsec}$.

$Y(\text{dec}) = (T_s/2)/(1.085\text{microsec})$

$\text{Count}(\text{dec}) = 65536(\text{dec}) - Y(\text{dec})$

$= \text{Count}(\text{hexa})$

```
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
```

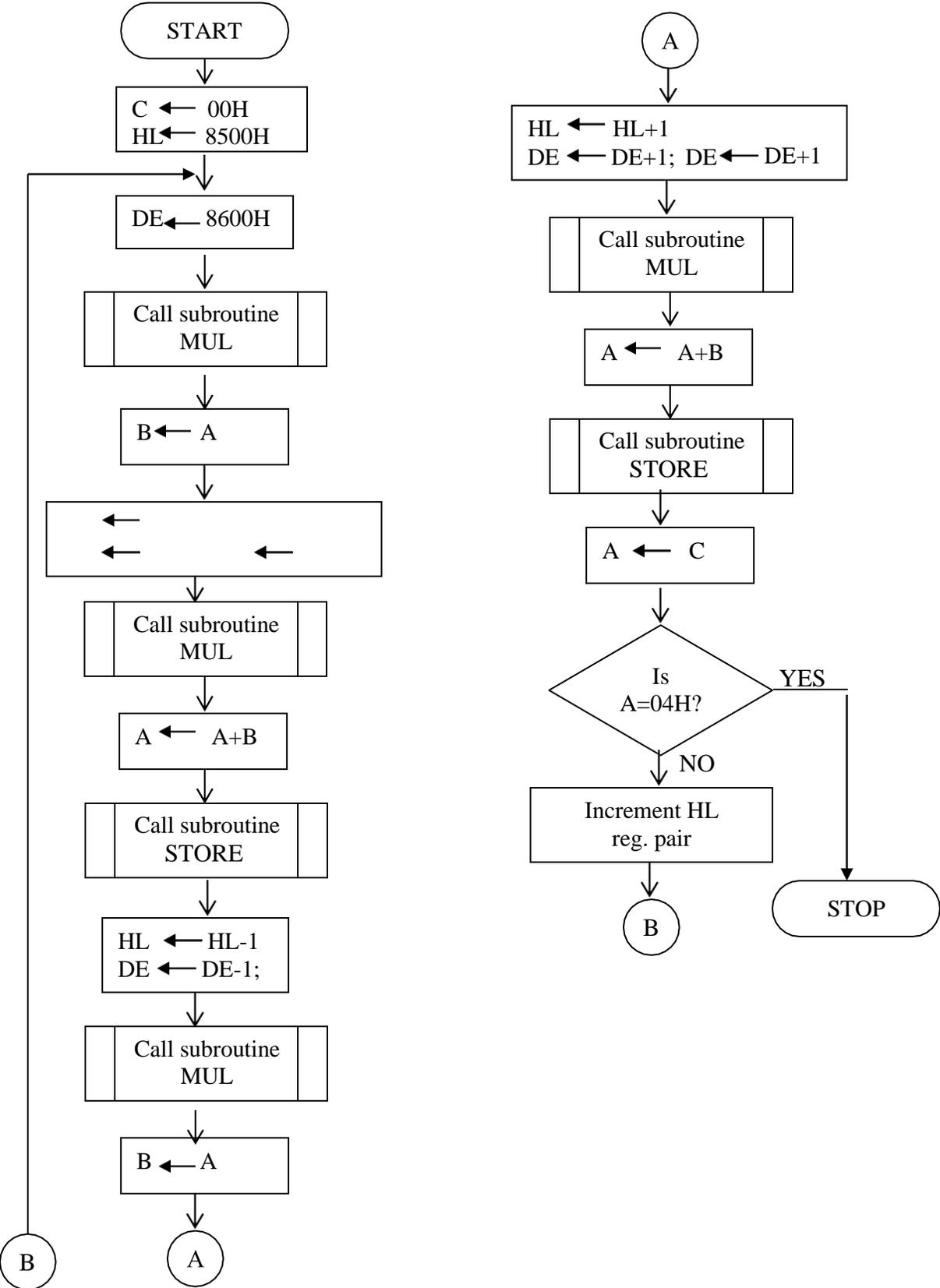
Result:

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:



18 2 X 2 MATRIX MULTIPLICATION

AIM:

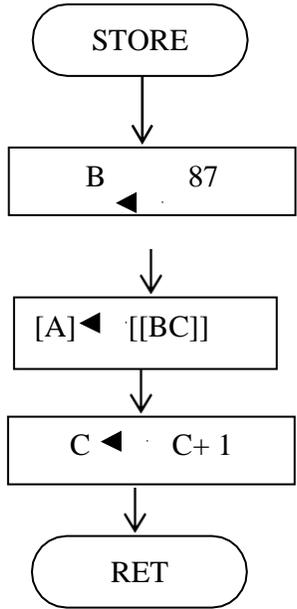
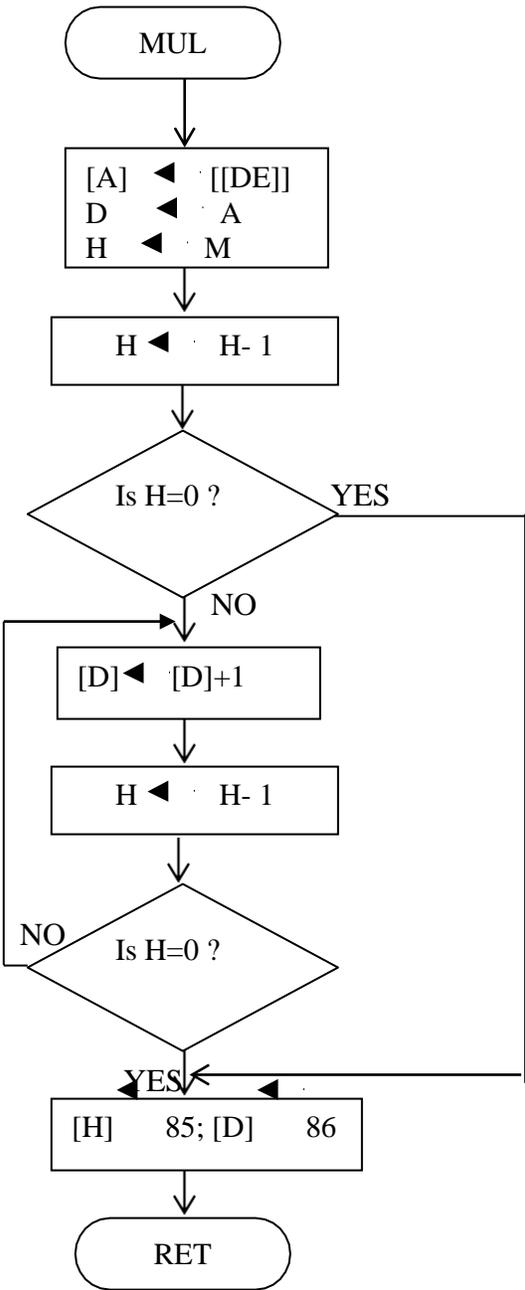
To perform the 2 x 2 matrix multiplication using 8085 microprocessor.

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



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 | | 4600 | | 4700 | |
| 4501 | | 4601 | | 4701 | |
| 4502 | | 4602 | | 4702 | |
| 4503 | | 4603 | | 4703 | |

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?