Varuvan Vadivelan Institute of Technology

Dharmapuri – 636 703

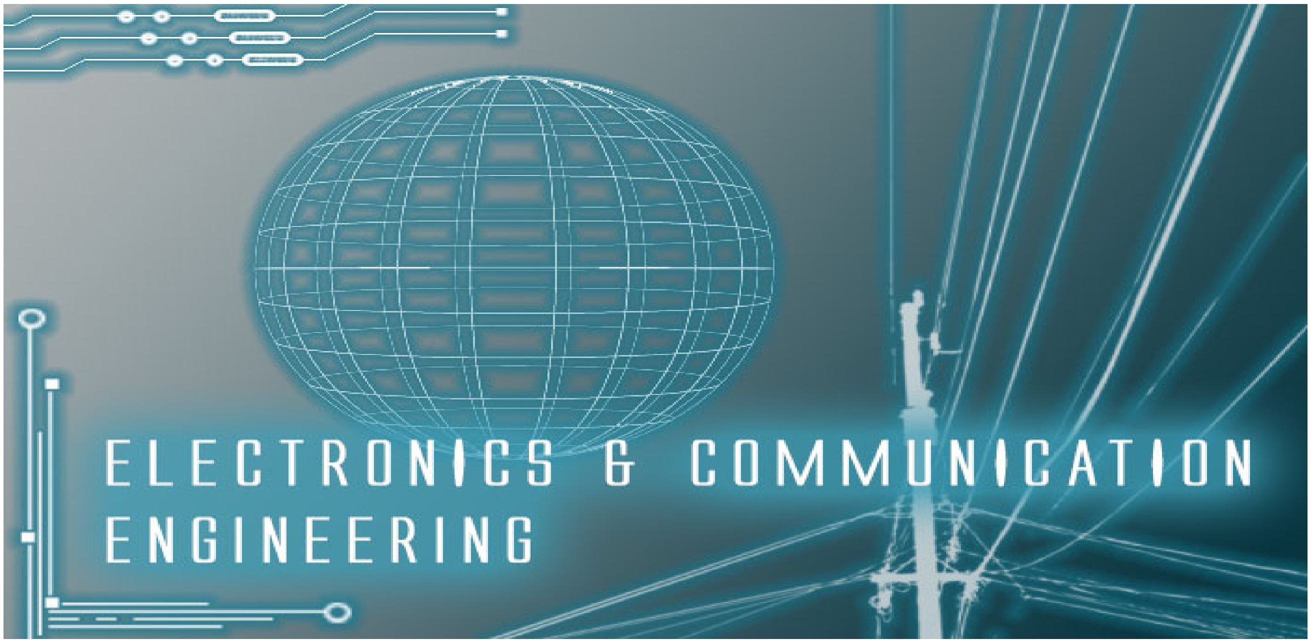
LAB MANUAL

Regulation : *2013*

Branch : *B.E. – ECE*

Year & Semester : III Year / V Semester

EC6513- MICROPROCESSOR AND MICROCONTROLLER LABORATORY



ANNA UNIVERSITY CHENNAI

Regulation 2013

EC6513- MICROPROCESSOR AND MICROCONTROLLER LABORATORY

SYLLABUS

LIST OF EXPERIMENTS

8086 Programs using kits and MASM

1. Basic arithmetic and Logical operations
2. Move a data block without overlap
3. Code conversion, decimal arithmetic and Matrix operations.
4. Floating point operations, string manipulations, sorting and searching
5. Password checking, Print RAM size and system date
6. Counters and Time Delay

Peripherals and Interfacing Experiments

7. Traffic light control
8. Stepper motor control
9. Digital clock
10. Key board and Display
11. Printer status
12. Serial interface and Parallel interface
13. A/D and D/A interface and Waveform Generation.

Experiments using kits and MASM

14. Basic arithmetic and Logical operations
15. Square and Cube program, Find 2's complement of a number
16. Unpacked BCD to ASCII

TOTAL: 45 PERIODS

INTRODUCTION TO MICROPROCESSORS & MICROCONTROLLERS

Microprocessor: is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC) at most a few integrated circuits. The microprocessor is a multipurpose, clock driven, register based, digital-integrated circuit which accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

Microcontroller: is a small computer on a single integrated circuit. In modern terminology, it is a system on a chip or SoC. A microcontroller contains one or more CPUs along with memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

INDEX

Ex. NO: 01

DATE:

16 BIT ADDITION USING ARITHMETIC OPERATION OF 8086 MICROPROCESSOR

AIM:

To write an assembly language program to perform addition of two 16 bit numbers using 8086.

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

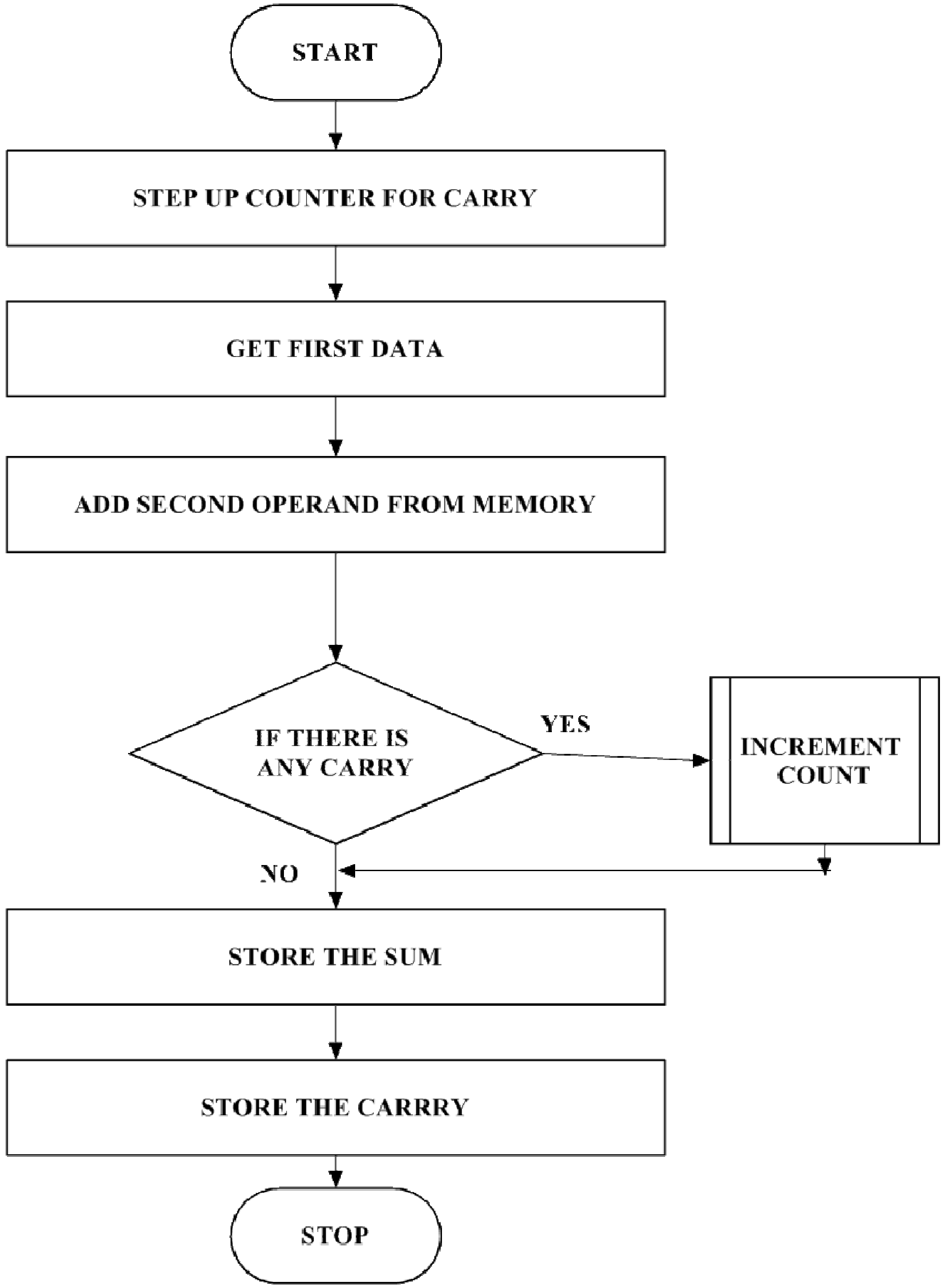
ALGORITHM:

16-bit addition

- Get the first number is specific address.
- Add the second number to the first number.
- Add the two values.
- Store the sum and carry.

FLOW CHART:

ADDITION:



PROGRAM FOR ADDITION;

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
1000			*MOV CX,0000H*	Initialize counter CX
1003			*MOV AX,[1200]*	Get the first data in AX register.
1006			*MOV BX, [1202]*	Get the second data in BX register.
100A			*ADD AX,BX*	Add the contents of both the register AX & BX
100C			*JNC L1*	Check for carry
100E			*INC CX*	If carry exists, increment the CX
100F		LI	*MOV [1206],CX*	Store the carry
1013			*MOV [1204], AX*	Store the sum
1016			*INT 3*	Stop the program

OUTPUT FOR ADDITION:

	ADDRESS	DATA
INPUT	1200 1201 1202 1203	
OUTPUT	1204 1205 1206	

RESULT:

Thus the assembly language program to perform addition of two 16 bit numbers using 8086 Performed and the result is stored.

Ex. NO: 02

DATE:

16 BIT SUBTRACTION

USING ARITHMETIC OPERATION OF 8086 MICROPROCESSOR

AIM:

To write an assembly language program to perform subtraction of two 16 bit numbers using 8086.

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

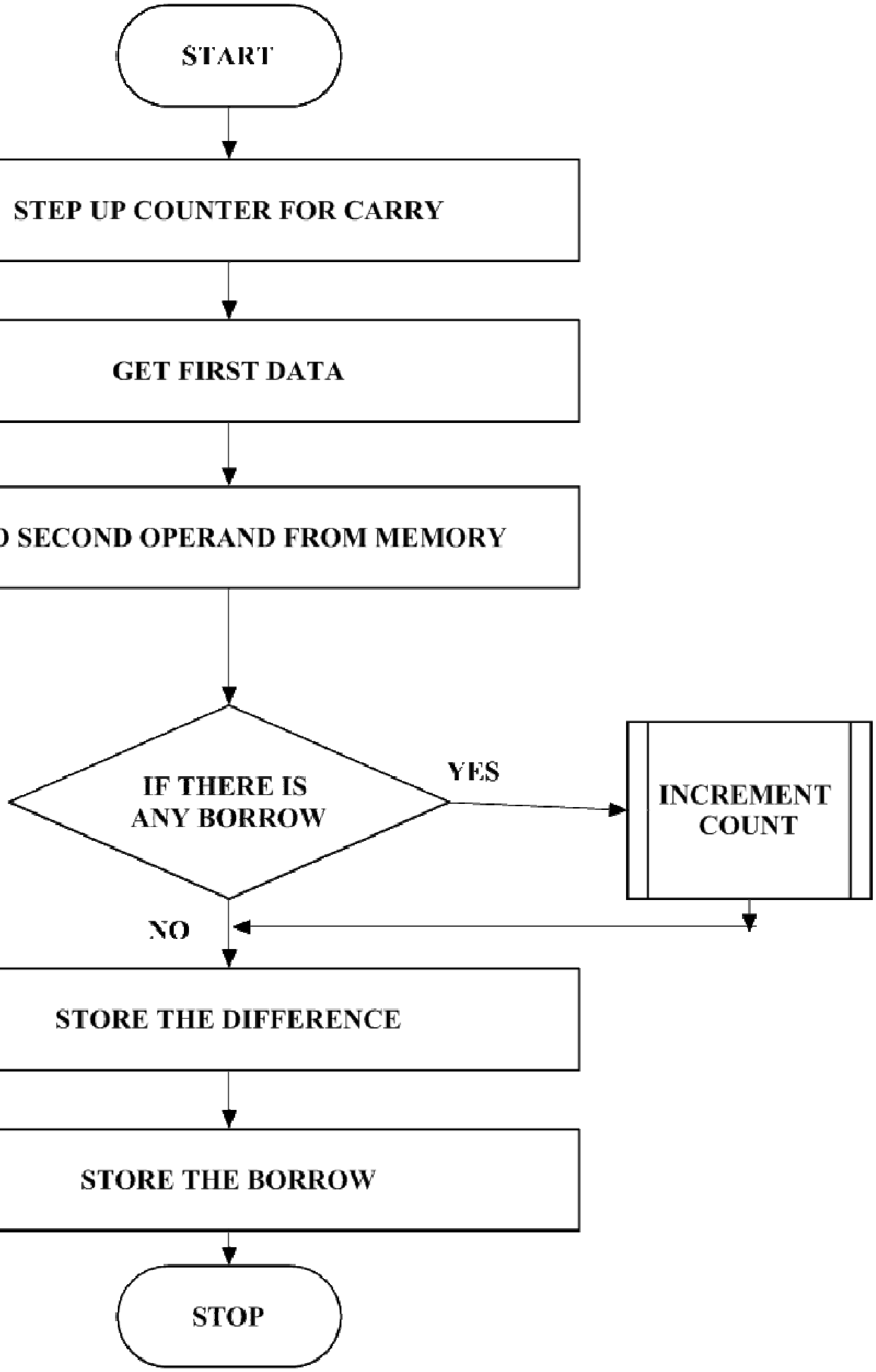
ALGORITHM:

16-bit SUBTRACTION:

- Initialize the MSBs of difference to 0
- Get the first number
- Subtract the second number from the first number.
- If there is any borrow, increment MSBs of difference by 1.
- Store LSBs of difference.
- Store MSBs of difference.

FLOECHART:

SUBTRACTION:



PROGRAM FOR SUBTRACTION:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
1000			`MOV CX,0000H`	Initialize counter CX
1003			`MOV AX,[1300]`	Get the first data in AX register
1006			`MOV BX, [1302]`	Get the second data in BX register.
100A			`SUB AX,BX`	Subtract the contents of both the register AX & BX
100C			`JNC A`	Check the Borrow.
100E			`INC CX`	If carry exists, increment the CX
100F			`MOV [1306],CX`	Store the Borrow.
1013			`MOV [1304], AX`	Store the difference.
1016			`INT 3`	Stop the program

OUTPUT FOR SUBTRACTION:

	ADDRESS	DATA
INPUT	1300 1301 1302 1303	
OUTPUT	1304 1305 1306	

RESULT:

Thus the assembly language program to perform subtraction of two 16 bit numbers using 8086 Performed and the result is stored.

Ex. NO: 03

DATE:

16 BIT MULTIPLICATION USING ARITHMETIC OPERATION OF 8086 MICROPROCESSOR

AIM:

To write an assembly language program to perform Multiplication of two 16 bit numbers using 8086.

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

ALGORITHM:

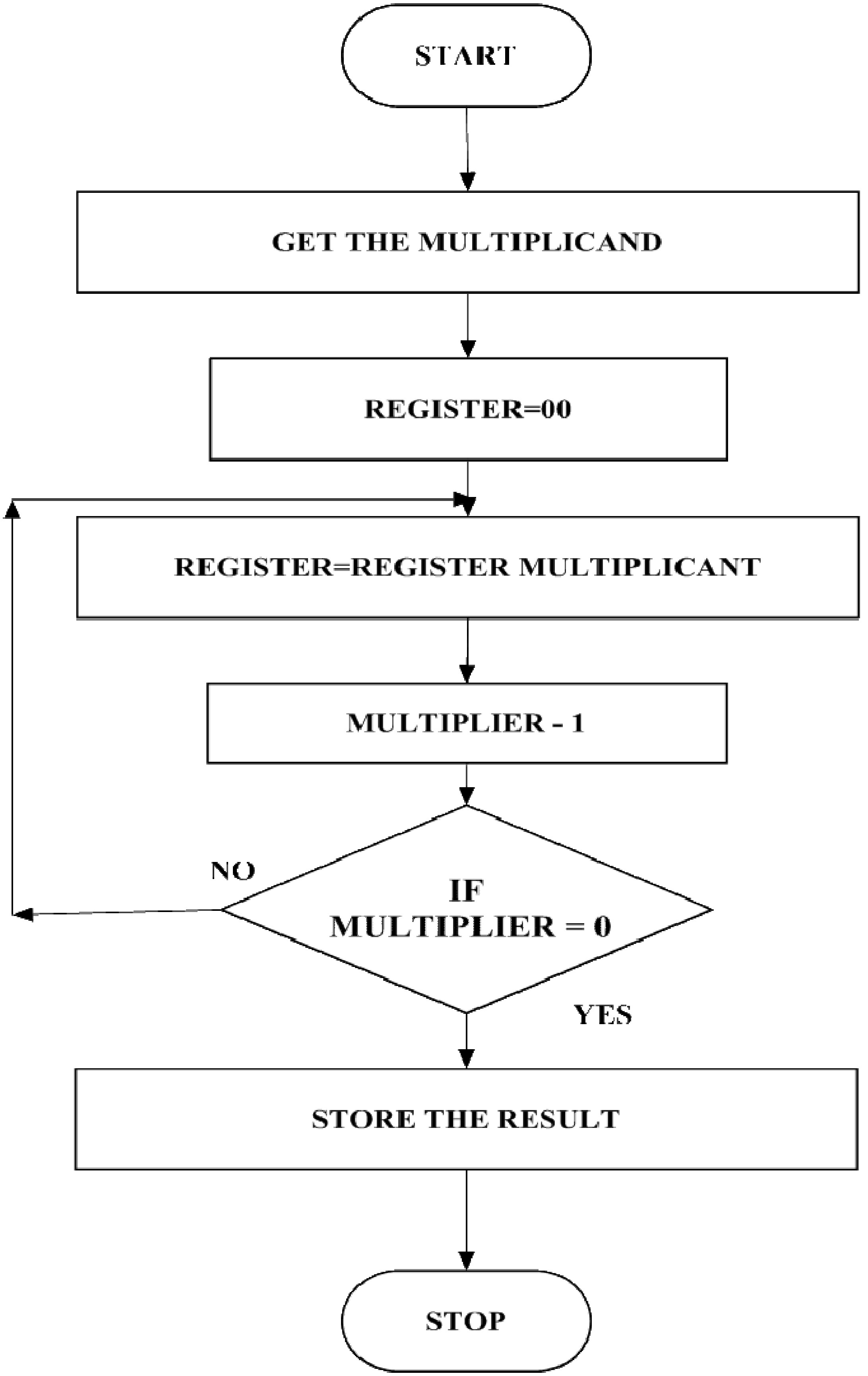
16-bit MULTIPLICATION

Multiplication of 16-bit numbers:

- Get the multiplier.
- Get the multiplicand
- Initialize the product to 0.
- Product = product + multiplicand
- Decrement the multiplier by 1.
- If multiplicand is not equal to 0, repeat from step (d) otherwise store the product.

FLOECHART:

MULTIPLICATION:



PROGRAM FOR MULTIPLICATION:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
1000			*MOV AX,1234H*	Get the first data in AX register.
1003			*MOV BX,0100H*	Get the second data in BX register.
1006			*MUL BX*	Multiply AX & BX data
1008			*INT 3*	Break point.

OUTPUT FORV MULTIPLICATION:

INPUT		
OUTPUT		

RESULT:

Thus the assembly language program to perform multiplication of two 16 bit numbers using 8086 Performed and the result is stored.

Ex. NO: 04

DATE:

16 BIT DIVISION USING ARITHMETIC OPERATION OF 8086 MICROPROCESSOR

AIM:

To write an assembly language program to perform division of two 16 bit numbers using 8086.

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIT	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

ALGORITHM:

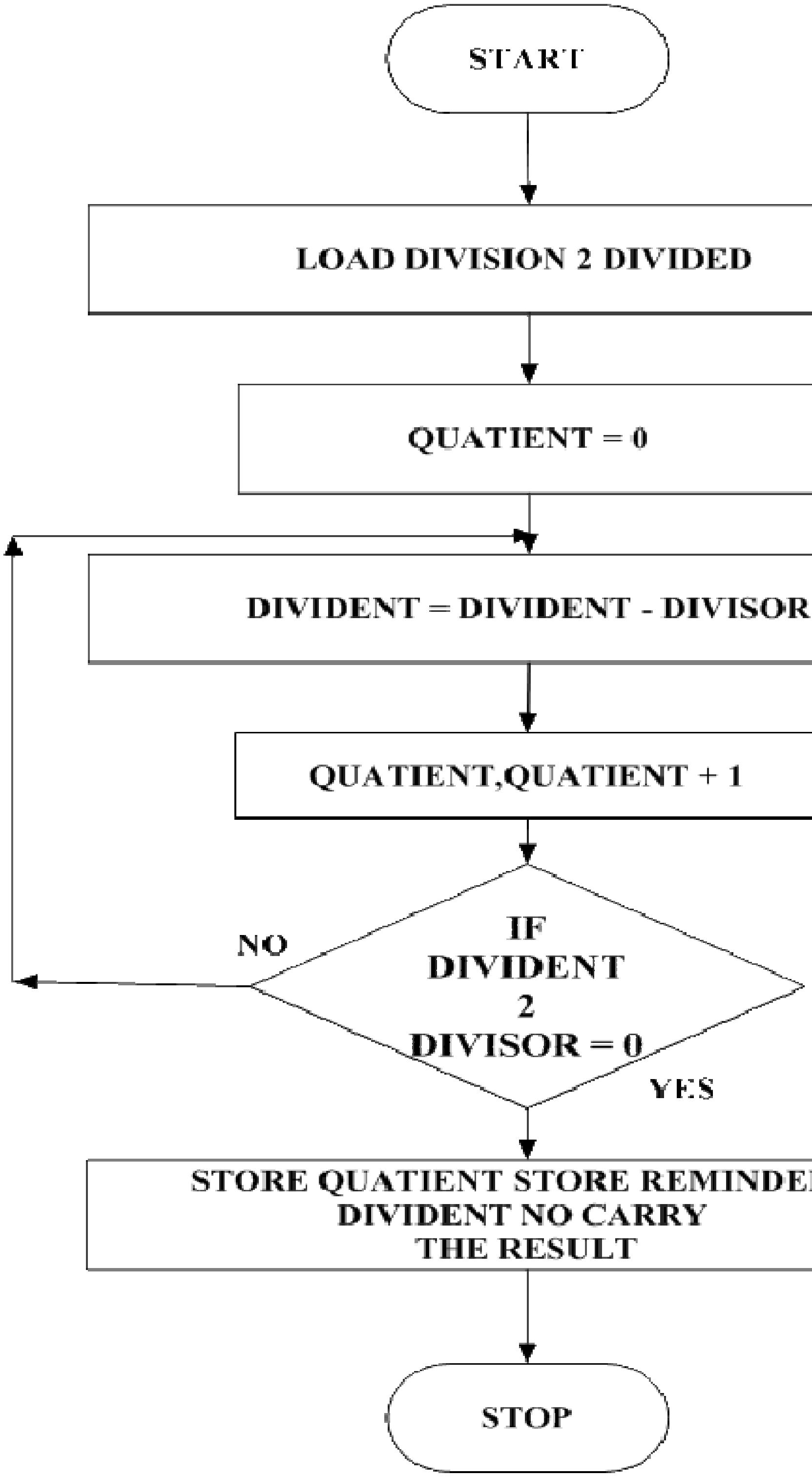
16-bit division

Division of 16-bit numbers:

- Get the dividend and divisor.
- Initialize the quotient to 0.
- Dividend = dividend–divisor
- If the divisor is greater, store the quotient
- Go to step 3
- If dividend is greater, quotient = quotient+ repeat from step 4.

FLOECHART:

DIVISION:



PROGRAM FOR DIVISION:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
1000			*MOV AX,[1200]*	Get the first data in AX register,
1003			*MOV DX,[1202]*	Get the second data in DX register.
1007			*MOV BX,[1204]*	Move the higher order data**.**
100D			*MOV [1206],AX*	Move ax register into address
100B			*DIV BX*	Divide the dividend by divisor
1010			*MOV AX,BX*	Copy the lower order data
1012			*MOV [1208],AX*	Store the higher order data.
1015			*INT 3*	Stop the program.

OUTPUT FOR DIVISION:

	ADDRESS	DATA
INPUT	1200 1201 1202 1203	
OUTPUT	1208 1209	

RESULT:

Thus the assembly language program to perform division of two 16 bit numbers using 8086 Performed and the result is stored.

EX. NO: 05

DATE :

LOGICAL OPERATIONS USING 8086 MICROCONTROLLER

AIM:

To move a data block without overlap

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

ALGORITHM:

- Initialize the memory location to the data pointer AL Register
- Increment B register.
- Increment accumulator by 1 and adjust it to decimal every time.
- Compare the given decimal number with accumulator value.
- Perform the given logical function value is in B register.
- Store the resultant in memory location.

PROGRAM FOR “*AND*” LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			***MOV AL,04***	Move data 04 to AL register
8003			***MOV BL,03***	Move data 03 to BL register
8007			***ANDI BL***	AND Operation
800D			***MOV #9000,BL***	Result store in 9000 address
800B			***HLT***	Stop the program

PROGRAM FOR “*OR*” LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			***MOV AL,05***	Move data 05 to AL register
8003			***MOV BL,04***	Move data 04 to BL register
8007			***ORI BL***	OR Operation
800D			***MOV #9000,BL***	Result store in 9000 address
800B			***HLT***	Stop the program

PROGRAM FOR "*EX- OR*" LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			***MOV AL,04***	Move data 04 to AL register
8003			***MOV BL,03***	Move data 03 to BL register
8007			***XOR BL***	EX-OR Operation
800D			***MOV #9000,BL***	Result store in 9000 address
800B			***HLT***	Stop the program

OUTPUT:

GATE	INPUT	OUTPUT
AND		
OR		
EX-OR		

RESULT:

Thus the assembly language program to perform logical operations AND, OR & EX-OR using 8086 Performed and the result is stored.

EX. NO: 06

DATE :

MOVE A DATA BLOCK WITHOUT OVERLAP

AIM:

To move a data block without overlap

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

ALGORITHM:

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

PROGRAM:

ADDRESS	OPCODES	PROGRAM	COMMENTS
1000		*MOV CL, 05*	Get the Data range
1002		*MOV SI, 1400*	Get the first data.
1005		*MOV DI, 1450*	Get the second data.
1008		*LD DSB*	Store the lower order product
1009		*MOV [DI], AL*	Store the result
100B		*INC DI*	Increment the pointer.
100C		*DEC 1008*	Dec Counter 0
1010		*INT 3*	Stop the program

OUTPUT:

INPUT		OUTPUT	
1400		**1450**	
1401		**1451**	
1402		**1452**	
1403		**1453**	
1404		**1454**	

RESULT:

Thus the output for the Move a data block without overlap was executed successfully.

EX. NO: 07

DATE :

CODE CONVERSION-DECIMAL TO HEXADECIMAL

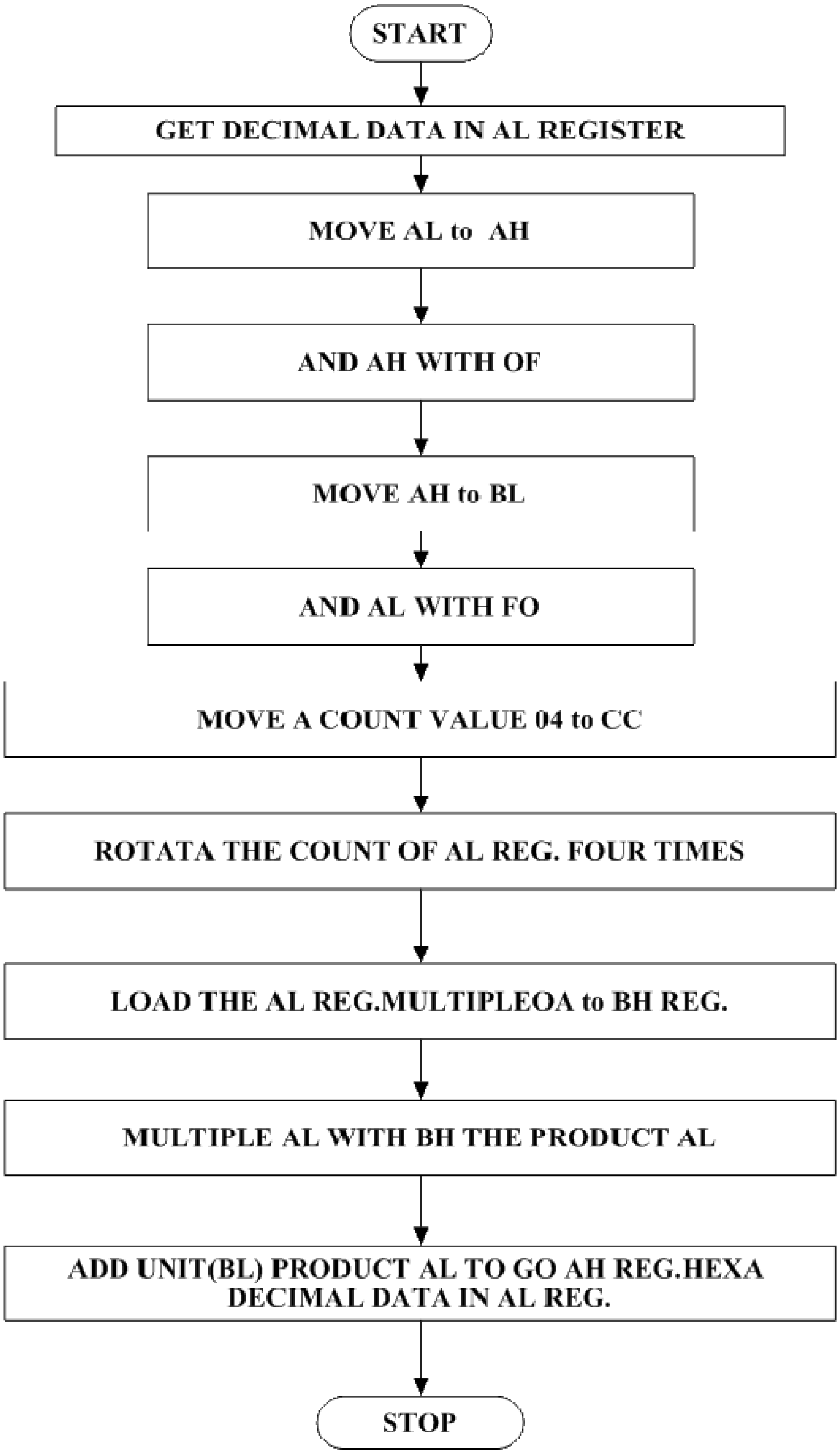
AIM:

To convert a given decimal number to hexadecimal.

ALGORITHM:

- ➢Initialize the memory location to the data pointer.
- ➢Increment B register.
- ➢Increment accumulator by 1 and adjust it to decimal every time.
- ➢Compare the given decimal number with accumulator value.
- ➢When both match, the equivalent hexadecimal value is in B register.
- ➢Store the resultant in memory location.

FLOWCHART:


START
GET DECIMAL DATA IN AL REGISTER
MOVE AL to AH
AND AH WITH OF
MOVE AH to BL
AND AL WITH FO
MOVE A COUNT VALUE 04 to CC
ROTATA THE COUNT OF AL REG. FOUR TIMES
LOAD THE AL REG.MULTIPLEOA to BH REG.
MULTIPLE AL WITH BH THE PRODUCT AL
ADD UNIT(BL) PRODUCT AL TO GO AH REG.HEXA DECIMAL DATA IN AL REG.
STOP

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENDS
1000			*MOV AL, [1100]*	Move data block AL
1003			*MOV AH, AL*	Move data lower to higher
1005			*MOV AH, 0F*	Move data OF into AH
1008			*MOV BL, AH*	Move data BL into AH
100A			*AND AL, F0*	AND the data AL to FO
100C			*MOV CL, 04*	Move data 04 to CL block
100E			*ROR AL, CL*	Rotate functions CL and AL
1010			*MOV BH, 0A*	Move data OA into BH
1012			*MUL BH*	Multiply BH
1014			*ADD AL, BL*	ADD the data AL And BL
1016			*MOV [2000], AL*	Move the store data
1019			*INT 3*	Stop the program

OUTPUT:[DECIMAL TO HEXADECIMAL]

DATA	ADRESS	DATA
INPUT		
OUTPUT		

RESULT:

Thus the code conversion of decimal to hexadecimal was executed successfully.

EX. NO: 08

DATE :

CODE CONVERSION –HEXADECIMAL TO DECIMAL

AIM:

To convert a given hexadecimal number to decimal

ALGORITHM:

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

PROGRAM;

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1000			*MOV AL, [1100]*	Move date to AL REG
1003			*MOV DX, 0000*	Move data AL TO DX
1006		HUND	*CMP AL, 64*	Move data to AX REG
1008			*JC TEN*	Jump carry
100A			*SUB AL, 64*	Subtract data
100C			*INC DL*	Increment DL
100E			*JMP HUND*	JUMP label data
1010		TEN	*CMP AL, 0A*	Compare register
1012			*JC UNIT*	Jump carry
1014			*SUB AL,0A*	Subtract data
1016			*INC DH*	Increment DH
1018			*JMP TEN*	JUMP carry
101A		UNIT	*MOV [2000],DL*	Move data to DL
101E			*MOV [2001],DH*	Move data to DH
1022			*MOV [2002],AL*	Move data to AL
1025			*MOV [2003],AH*	Move data to AH
1027			*HLT*	Stop the program

OUTPUT:

	INPUT	OUTPUT
MEMORY		
DATA		

RESULT:

Thus the code conversion of decimal to hexadecimal was executed successfully.

EX. NO: 09

DATE :

STRING MANIPULATION - SORTING & SEARCHING

ASCENDING & DESCENDING

AIM:

To write an Assembly Language Program (ALP) to sort a given array in Ascending and Descending order

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

PROBLEM STATEMENT:

An array of length 05 is given from the location. Sort it into descending and ascending order and store the result.

ALGORITHM:

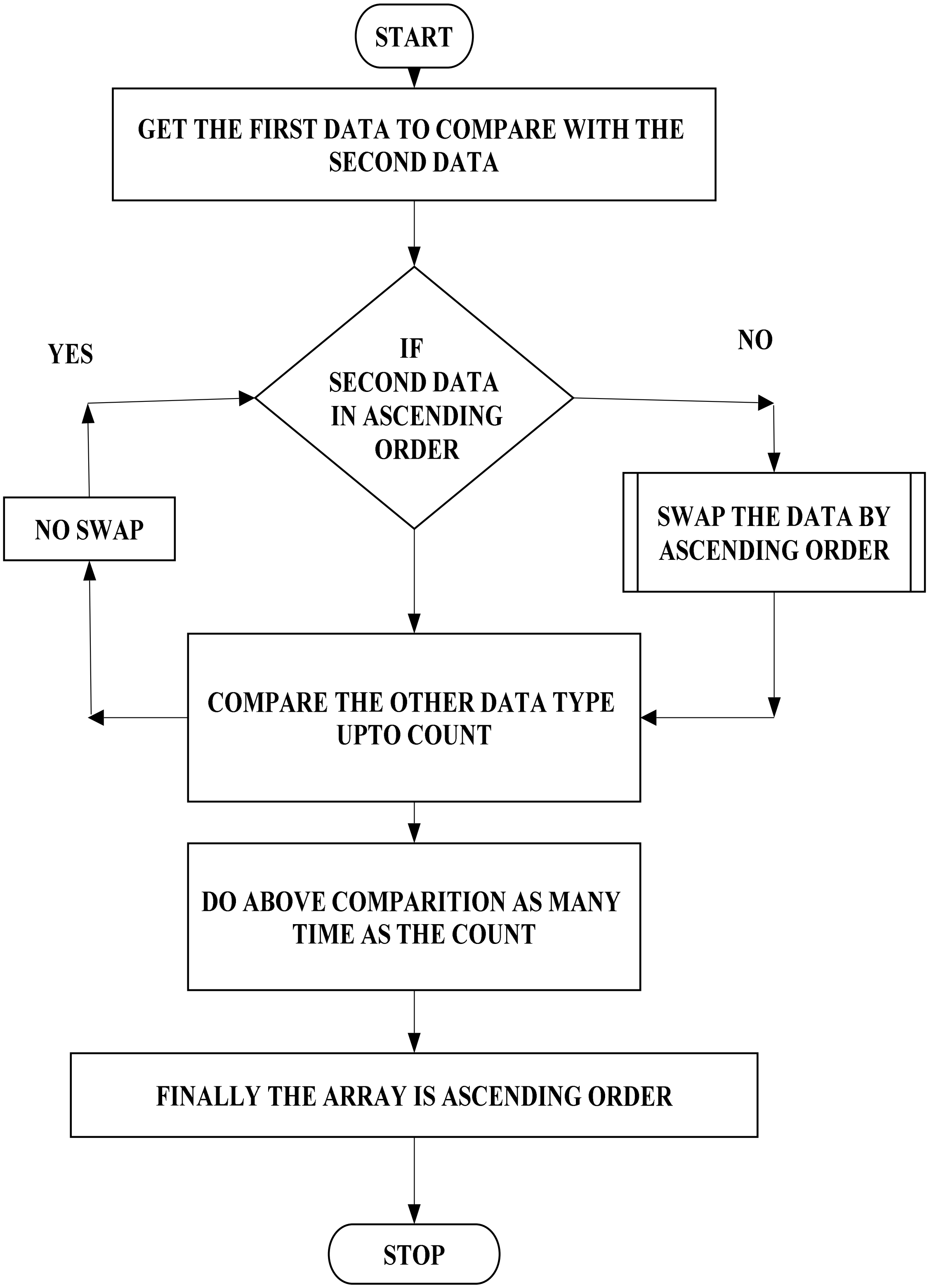
Sorting in ascending order:

- Load the array count in two registers C_1 and C_2.
- Get the first two numbers.
- Compare the numbers and exchange if necessary so that the two numbers are in ascending order.
- Decrement C_2.
- Get the third number from the array and repeat the process until C_2 is 0.
- Decrement C_1 and repeat the process until C_1 is 0.

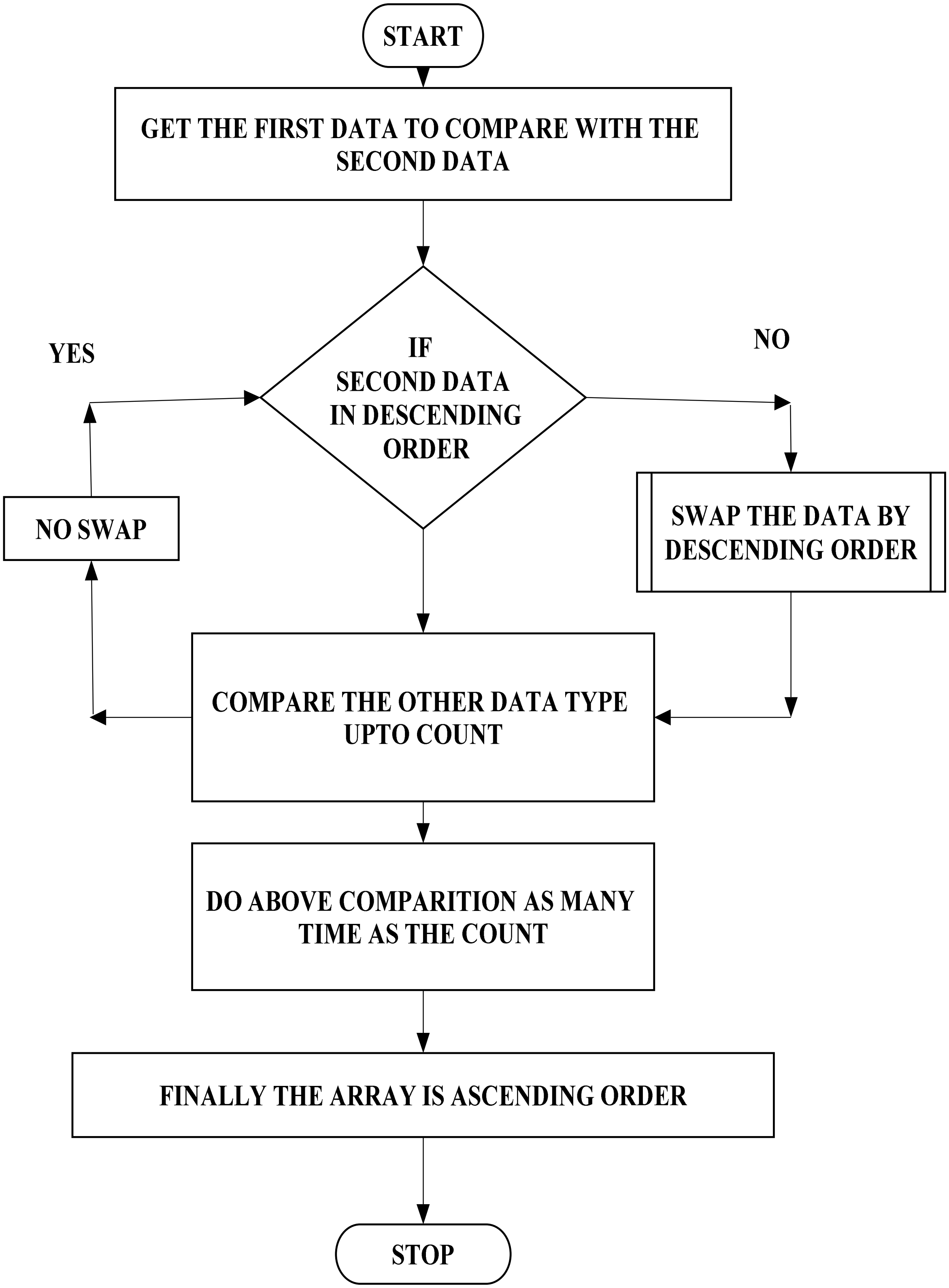
Sorting in descending order:

- Load the array count in two registers C_1 and C_2.
- Get the first two numbers.
- Compare the numbers and exchange if necessary so that the two numbers are in descending order.
- Decrement C_2.
- Get the third number from the array and repeat the process until C_2 is 0.
- Decrement C_1 and repeat the process until C_1 is 0.

FLOECHART:[ASCENDING]:


START
GET THE FIRST DATA TO COMPARE WITH THE SECOND DATA
IF SECOND DATA IN ASCENDING ORDER
YES
NO
NO SWAP
SWAP THE DATA BY ASCENDING ORDER
COMPARE THE OTHER DATA TYPE UPTO COUNT
DO ABOVE COMPARITION AS MANY TIME AS THE COUNT
FINALLY THE ARRAY IS ASCENDING ORDER
STOP

FLOWCHART :[DECENDING]:


START
GET THE FIRST DATA TO COMPARE WITH THE SECOND DATA
IF SECOND DATA IN DESCENDING ORDER
YES
NO
NO SWAP
SWAP THE DATA BY DESCENDING ORDER
COMPARE THE OTHER DATA TYPE UPTO COUNT
DO ABOVE COMPARITION AS MANY TIME AS THE COUNT
FINALLY THE ARRAY IS ASCENDING ORDER
STOP

PROGRAM FOR ASCENDING ORDER:

ADDRESS	LABEL	PROGRAM	COMMENTS
1000		*MOV SI,1200H*	Initialize memory location for array size
1002		*MOV CL,[SI]*	Number of comparisons in CL
1004		*L4 : MOVSI,1200H*	Initialize memory location for array size
1005	L4	*MOV DL,[SI]*	Get the count in DL
1007		*INC SI*	Go to next memory location
100D	L3	*MOV AL,[SI]*	Get the first data in AL
101B	L1	*L3 : INC SI*	Go to next memory location
101E	L2	*MOV BL,[SI]*	Get the second data in BL
1010		*CMP AL,BL*	Compare two data’s
1012		*JNB L1*	If AL < BL go to L1
1014		*DEC SI*	Else, Decrement the memory location
1016		*MOV [SI],AL*	Store the smallest data
1018		*MOV AL,BL*	Get the next data AL
1019		*JMP L2*	Jump to L2
101A		*L1 : DEC SI*	Decrement the memory location
101C		*MOV [SI],BL*	Store the greatest data in memory location
101E		*L2 : INC SI*	Go to next memory location
1020		*DEC DL*	Decrement the count
1022		*JNZ L3*	Jump to L3, if the count is not reached
1024		*MOV [SI],AL*	Store data in memory location
1026		*DEC CL*	Decrement the count
1028		*JNZ L4*	Jump to L4, if the count is not reached zero
1029		*HLT*	Stop the program

PROGRAM FOR DESCENDING ORDER:

ADDRESS	OPCODES	PROGRAM	COMMENTS
9000		*MOV SI,9000H*	Initialize memory location for array size
9002		*MOV CL,[SI]*	Number of comparisons in CL
9004		*L4 : MOV SI,9000H*	Initialize memory location for array size
9006		*MOV DL,[SI]*	Get the count in DL
9007		*INC SI*	Go to next memory location
9009		*MOV AL,[SI]*	Get the first data in AL
900B		*L3 : INC SI*	Go to next memory location
900D		*MOV BL,[SI]*	Move the data SI reg into BL reg
900F		*CMP AL,BC*	Compare BC and AL register
9010		*JB 101B*	Jump given address
9012		*DEC SI*	Decrement SI
9014		*MOV [SI],AL*	Move the data AL register into SI register
9016		*MOV AL,BL*	Move the data AL into BL
9018		*JMP 101E*	Jump given address

901A		*DEC SI*	Decrement SI
901C		*MOV [SI],AL*	Move the data AL into SI register
901E		*INC SI*	Increment SI
9020		*DEC SI*	Decrement SI
9022		*JNZ 1000*	Jump no zero
9024		*MOV [SI],AL*	Move AL into SI register
9026		*DEC CL*	Decrement CL
9028		*JNZ 1005*	Jump no zero 1005
902A		*INT 3*	Stop the program

OUTPUT FOR ASCENDING:

	DATA					
INPUT						
OUTPUT						

OUTPUT FOR DESCENDING ORDER:

	DATA					
INPUT						
OUTPUT						

RESULT:

Thus the given array of numbers are sorted in ascending & descending order.

EX. NO: 10

DATE :

LARGEST & SMALLEST

AIM:

To write an Assembly Language Program(ALP) to find the Largest and Smallest number in a given array.

APPARATUS REQUIRED:

S.NO	ITEM	SPECIFICATION	QUANTITY
1.	MICROPROCESSOR KIR	8086 KIT	1
2.	POWER SUPPLY	+ 5 V DC	1
3.	KEY BOARD	-	1

PROBLEM STATEMENT:

An array of length 5 is given from the location. Find the largest and smallest number and store the result.

ALGORITHM:

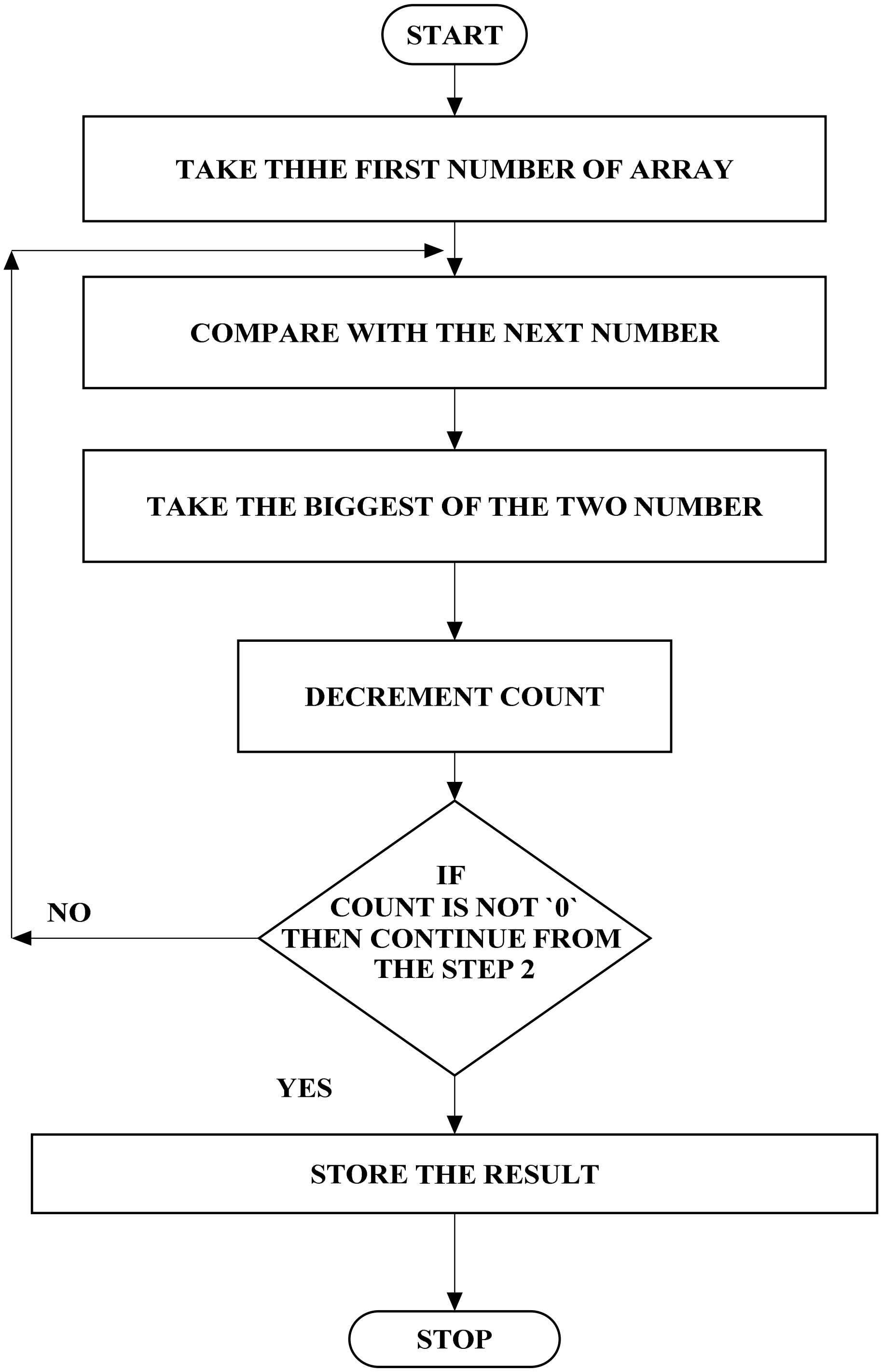
(i) Finding largest number:

- Load the array count in a register C_1.
- Get the first two numbers.
- Compare the numbers and exchange if the number is small.
- Get the third number from the array and repeat the process until C_1 is 0.

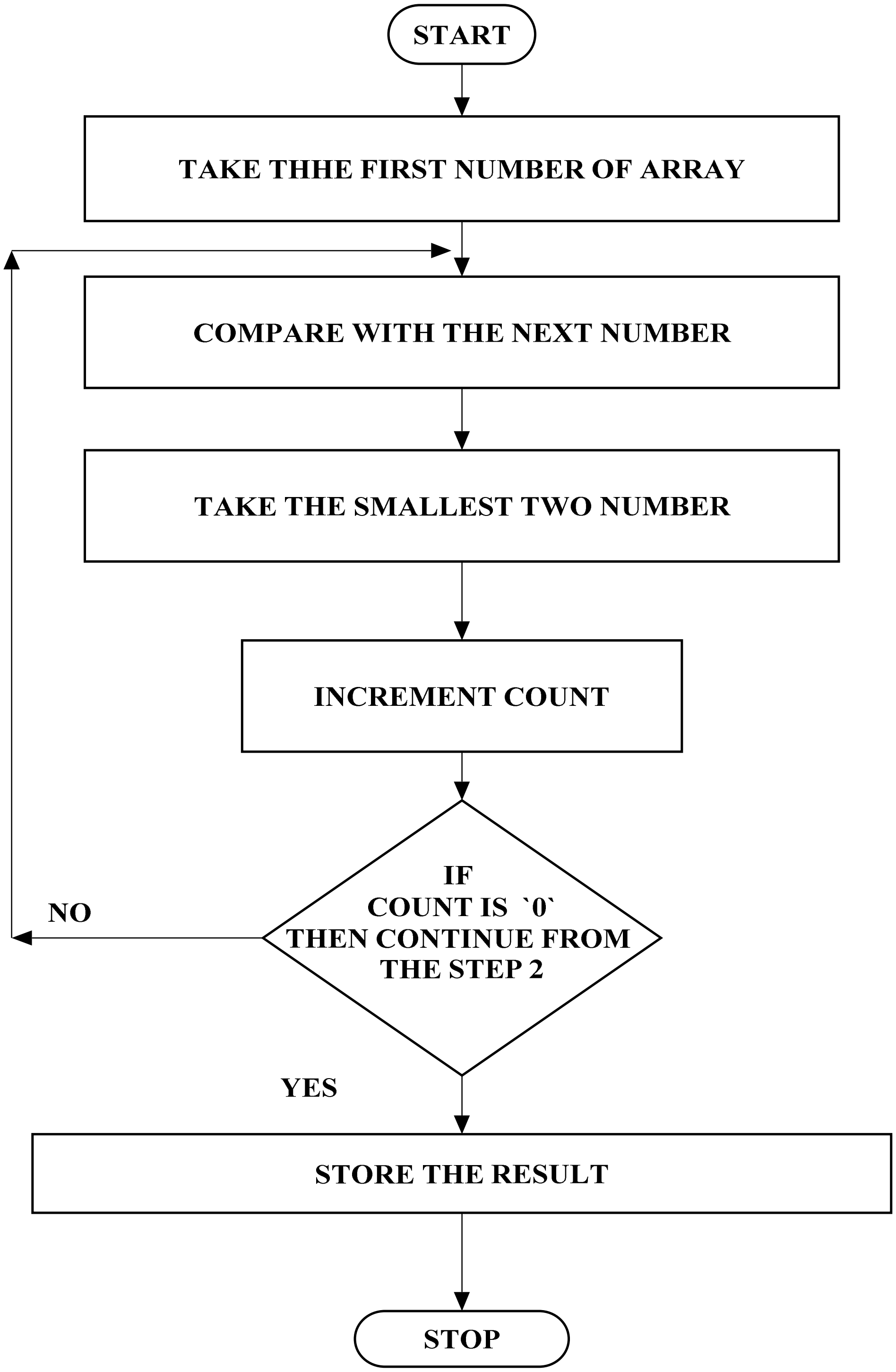
(ii) Finding smallest number:

- Load the array count in a register C1.
- Get the first two numbers.
- Compare the numbers and exchange if the number is large.
- Get the third number from the array and repeat the process until C1 is 0.

FLOECHART:[LARGEST]


START
TAKE THHE FIRST NUMBER OF ARRAY
COMPARE WITH THE NEXT NUMBER
TAKE THE BIGGEST OF THE TWO NUMBER
DECREMENT COUNT
IF
COUNT IS NOT `0`
THEN CONTINUE FROM
THE STEP 2
NO
YES
STORE THE RESULT
STOP

FLOECHART:[SMALLEST]


START
TAKE THHE FIRST NUMBER OF ARRAY
COMPARE WITH THE NEXT NUMBER
TAKE THE SMALLEST TWO NUMBER
INCREMENT COUNT
IF
COUNT IS `0`
THEN CONTINUE FROM
THE STEP 2
NO
YES
STORE THE RESULT
STOP

PROGRAM FOR FINDING LARGEST NUMBER:

ADDRESS	OPCODES	PROGRAM	COMMENDS
1000		***MOV SI,9000H***	Initialize array size
1002		***MOV CL,[SI]***	Initialize the count
1004		***INC SI***	Go to next memory location
1006		***MOV AL,[SI]***	Move the first data in AL
1007		***DEC CL***	Reduce the count
1009		***INC SI***	Move the SI pointer to next data
100A	L2	***CMP AL,[SI]***	Compare two data's
100E		***JNB L1***	If AL > [SI] then go to L1 (no swap)
1011	L1	***MOV AL,[SI]***	Else move the large number to AL
1012		***L1 : DEC CL***	Decrement the count
1014		***JNZ L2***	If count is not zero go to L2
1016		***MOV DI,9500H***	Initialize DI with 1300H
1018		***MOV [DI],AL***	Else store the biggest number in 1300 location
1010		***HLT***	Stop the program

PROGRAM FOR FINDING SMALLEST NUMBER:

ADDRESS	OPCODES	PROGRAM	COMMENDS
1000		***MOV SI,9000H***	Initialize array size
1002		***MOV CL,[SI]***	Initialize the count
1004		***INC SI***	Go to next memory location
1006		***MOV AL,[SI]***	Move the first data in AL
1007		***DEC CL***	Reduce the count
1009		***L2 : INC SI***	Move the SI pointer to next data
100A	L2	***CMP AL,[SI]***	Compare two data's
100E		***JB L1***	If AL < [SI] then go to L1 (no swap)
1011	L1	***MOV AL,[SI]***	Else move the large number to AL
1012		***L1 : DEC CL***	Decrement the count
1014		***JNZ L2***	If count is not zero go to L2
1016		***MOV DI,9500H***	Initialize DI with 1300H
1018		***MOV [DI],AL***	Else store the biggest number in 1300 location
1010		***HLT***	Stop the program

OUTPUT FOR LARGESTNUMBER:

	DATA					
INPUT						
OUTPUT						

OUTPUT FOR SMALLEST NUMBER:

	DATA					
INPUT						
OUTPUT						

RESULT:

Thus the largest and smallest number is found in a given array.

EX. NO: 11

DATE :

PASSWORD CHECKING

AIM:

To write an Assembly Language Program (ALP) for performing the Password checking by using MASM

APPARATUS REQUIRED:

SL .No	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8086 kit	1
2.	Power Supply	+5 V dc	1

PROGRAM:

```
; PASSWORD IS MASM1234

DATA SEGMENT

PASSWORD DB 'MASM1234'

LEN EQU ($-PASSWORD)

MSG1 DB 10, 13,'ENTER YOUR PASSWORD: $'

MSG2 DB 10, 13,' WELCOME TO ELECTRONICS WORLD!!$'

MSG3 DB 10, 13,'INCORRECT PASSWORD!$'

NEW DB 10, 13,'$'

INST DB 10 DUP (0)

DATA ENDS

CODE SEGMENT
```

```
 ASSUME CS: CODE, DS: DATA
 START:
 MOV AX, DATA
 MOV DS, AX

 LEA DX, MSG1
 MOV AH, 09H
 INT 21H
 MOV SI, 00
 UP1:
 MOV AH, 08H
 INT 21H
 CMP AL, 0DH
 JE DOWN
 MOV [INST+SI], AL
 MOV DL,'*'
 MOV AH, 02H
 INT 21H
 INC SI
 JMP UP1
 DOWN:
MOV BX, 00
MOV CX, LEN
CHECK:
MOV AL,[INST+BX]
MOV DL,[PASSWORD+BX]
CMP AL, DL
JNE FAIL
```

```
INC BX

LOOP CHECK

LEA DX, MSG2

MOV AH, 09H

INT 21H

JMP FINISH

FAIL:

LEA DX, MSG3

MOV AH, 009H

INT 21H

FINISH:

INT 3

CODE ENDS

END START

END
```

RESULT:

Thus the output for the Password checking, Print RAM size and system date was executed successfully

EXP.NO: 12

DATE :

TRAFFIC LIGHT CONTROLLER

AIM:

To write an assembly language program in 8086 to Traffic light control

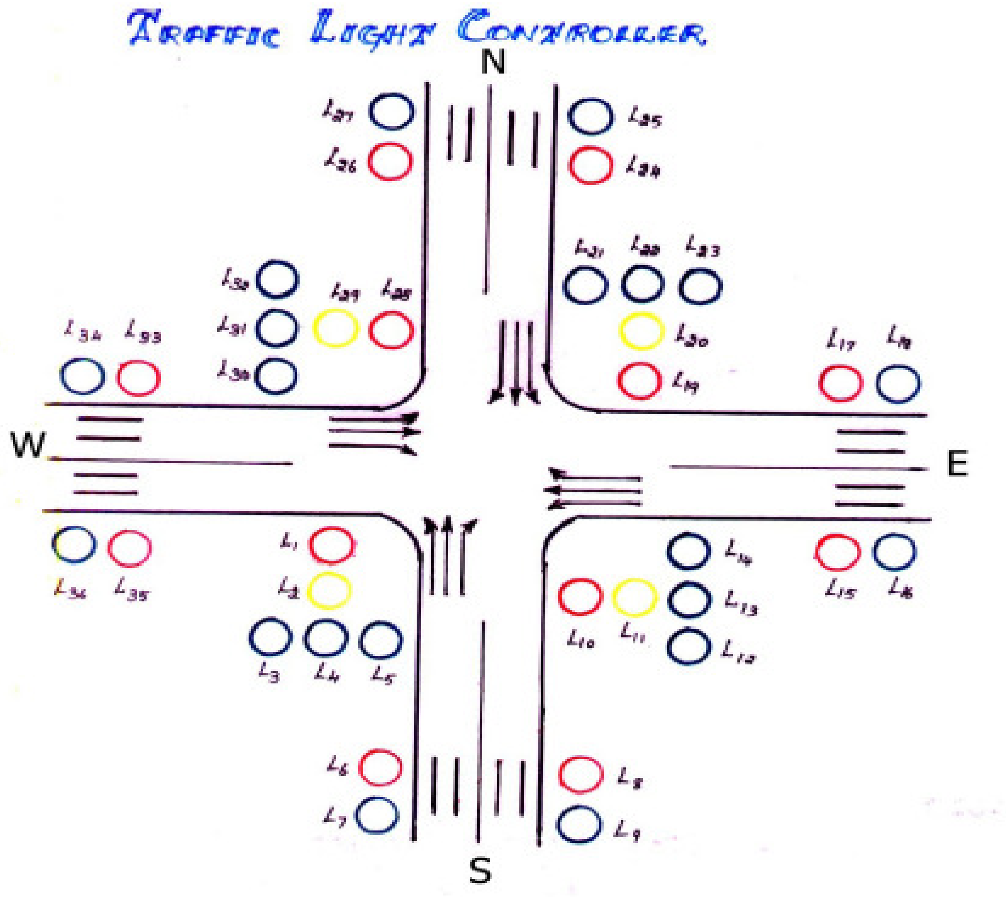
APPARATUS REQUIRED:

SL .No	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8086 kit	1
2.	Power Supply	+5 V dc	1

PROGRAM;

- Log into System.
- Select control type.
- If Automatic mode select then go to step 4th else go to step 8.
- If Automatic control activated.
- Assign time period for green, yellow signal.
- If emergency vehicle is over then go to step 4.
- If rally come then go to step 8.
- Manual control activated.
- Assign time period for green, yellow signal according to that particular road.
- If emergency over then go to step 4.

MODEL GRAPH FOR TRAFFIC LIGHT CONTROL:



ASSEMBLY LANGUAGE PROGRAM FOR TRAFFIC LIGHT CONTROL:

ADDRESS	OPCODE	LABEL	MNEMONICS
1000			MVI A,80
1002			OUT CWR
1004		REPEAT	MVI E, 03
1006			LXI H, C100
1007		NEXTSTAT	MOV A, M
1009			OUT PORRTA
100B			INX H
100E			MOV A, M
1010			OUT PORTB
1012			INX H
1014			MOV A,M
1016			OUT PORT C
1018			CALL DELAY
1019			INX H
101A			DCR E
101C			JNZ NEXTSTAT
101E			JMP REPEAT
1022		DELAY	LXI D, 3000
1024		L2	MVI C,FF
1026		L1	DCR C
1028			JNZ L1
1029			DCR D
1000			MOV A, D
1002			ORA E
1004			JNZ L2
1006			RET

RESULT:

Thus the assembly language program for traffic light control is verified

EX. NO: 13

DATE :

STEPPER MOTOR INTERFACING

AIM:

To write an assembly language program in 8086 to rotate the motor at different speeds.

APPARATUS REQUIRED:

SL.NO	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8086	1
2.	Power Supply	+5 V, dc,+12 V dc	1
3.	Stepper Motor Interface board	-	1
4.	Stepper Motor	-	1

PROBLEM STATEMENT:

Write a code for achieving a specific angle of rotation in a given time and particular number of rotations in a specific time.

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 stepwise manner from one equilibrium position to the next. Two-phase scheme: Any two adjacent stator windings are energized. There are two magnetic fields active in quadrature and none of the rotor pole faces can be in direct alignment with the stator poles. A partial but symmetric alignment of the rotor poles is of course possible.

ALGORITHM:

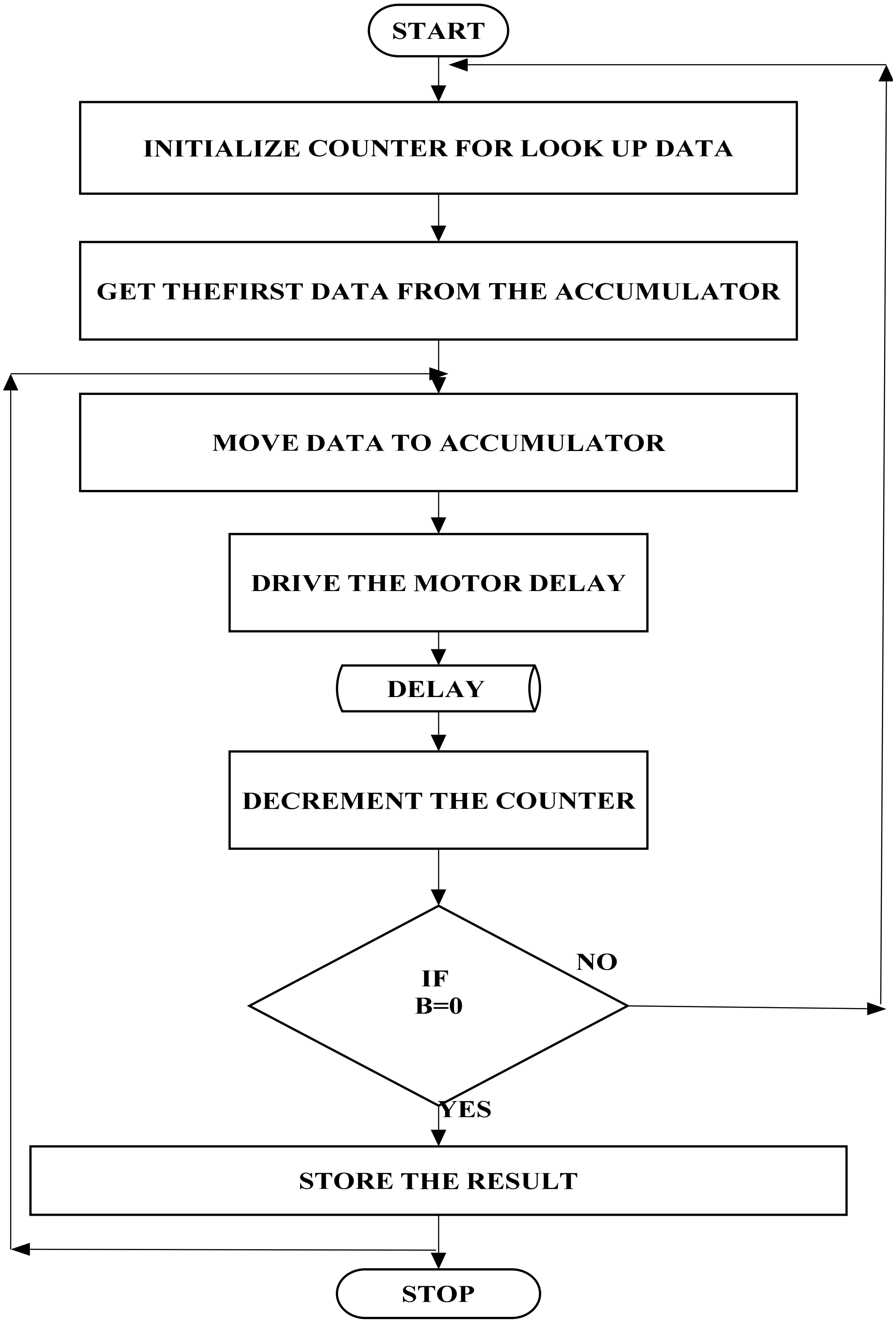
For running stepper motor clockwise and anticlockwise directions

- Get the first data from the lookup table.
- Initialize the counter and move data into accumulator.
- Drive the stepper motor circuitry and introduce delay
- Decrement the counter is not zero repeat from step(iii)
- Repeat the above procedure both for backward and forward directions.

SWITCHING SEQUENCE OF STEPPER MOTOR:

MEMORY LOCATION	A1	A2	B1	B2	HEX CODE
4500	1	0	0	0	09 H
4501	0	1	0	1	05 H
4502	0	1	1	0	06 H
4503	1	0	1	0	0A H

FLOWCHART:


START
INITIALIZE COUNTER FOR LOOK UP DATA
GET THEFIRST DATA FROM THE ACCUMULATOR
MOVE DATA TO ACCUMULATOR
DRIVE THE MOTOR DELAY
DELAY
DECREMENT THE COUNTER
IF
B=0
NO
YES
STORE THE RESULT
STOP

PROGRAM FOR STEPPER MOTOR CONTOL;

ADDRESS	OPCODE	PROGRAM	COMMENTS
1000		*MOV DX,FF26*	Initialize memory location to store the array of number
1002		*MOV AL,80*	Initialize array size
1004		*OUT DX,AL*	Copy the first data in AL
1006		*MOV DX,FF20*	Send it through port address
1007		*MOV AL,05*	Introduce delay
1009		*OUT DX,AL*	Declare DX
100B		*CALL 1100*	JUNP no zero
100E		*MOV AL,07*	Increment DI
1010		*OUT DX,AL*	Go to next memory location
1012		*CALL 1100*	Loop until all the data's have been sent Go to start location for continuous rotation
1014		*MOV AL,06*	Array of data's
1015		*OUT DX,AL*	Output data from DX into AL
1017		*CALL 1100*	Call given address
1018		*MOV AL,04*	Move the data 04 to AL Register
101D		*OUT DX,AL*	Output data from DX into AL
101E		*CALL 1100*	Call given address
1021		*JMP 1006*	Jump the program given address

DELAY SUBROTINE

ADDRESS	OPCODE	PROGRAM	COMMENTS
1100		*MOVBX, 0010*	Initialize memory location to store the array of number
1103		*MOV AL,FF*	Initialize array size
1105		*NOP*	No Operation
1106		*NOP*	No Operation
1107		*NOP*	No Operation
1108		*NOP*	No Operation
1109		*DEC AL*	Decrement AL
110B		*JNZ 1105*	Jump no zero
110D		*DEC BX*	Decrement BX
110E		*JNZ 1103*	Jump no zero
1110		*RET*	Return main program

RESULT:

Thus the assembly language program for rotating stepper motor in both clockwise and anticlockwise directions is written and verified.

EX. NO: 14

DATE :

INTERFACING PRGRAMMABLE KEYBOARD AND DISPLAY CONTROLLER 8279

AIM :

To display the message “2” using Keyboard and Display Controller-8279

APPARATUS REQUIRED:

SL.NO	ITEM	SPECIFICATION	QUANTITY
1.	Microprocessor kit	8086	1
2.	Power Supply	+5 V, dc,+12 V dc	1
3.	8279- Interface board	-	1

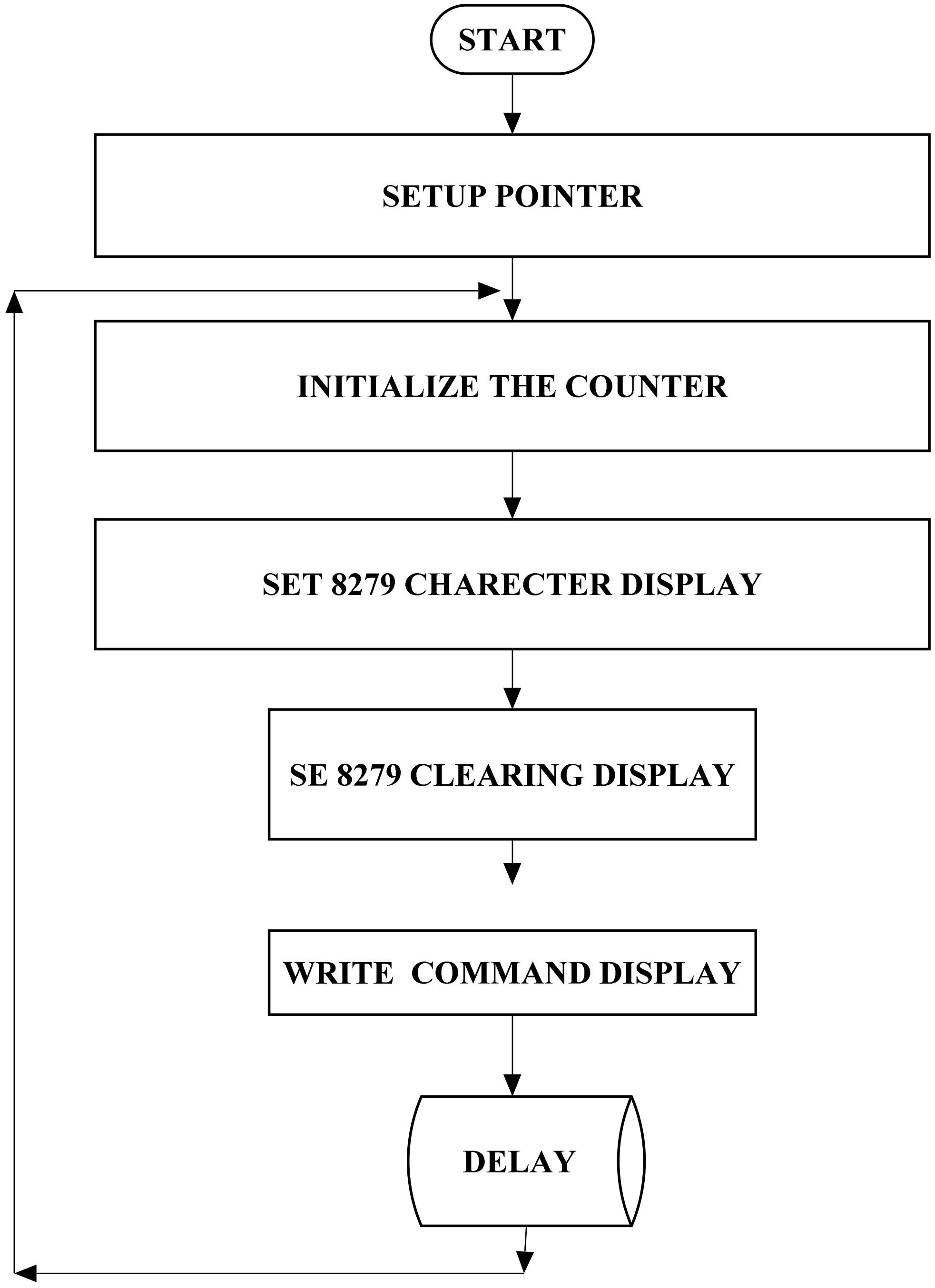
ALGORITHM :

- Display of rolling message “HELP US “
- Initialize the counter
- Set 8279 for 8 digit character display, right entry
- Set 8279 for clearing the display
- Write the command to display
- Load the character into accumulator and display it
- Introduce the delay
- Repeat from step 1.

PROGRAM:

MEMORY LOCATION	OPCODES	PROGRAM	COMMENDS
9000		*MVI C,BA*	Initialize array
9002		*MVI A,12*	Initialize array size
9003		*OUT 71*	Store the control word for display mode
9006		*MVI A,3E*	Send through output port
9009		*OUT 71*	Store the control word to clear display
900B		*MVI A,A0*	Send through output port
900E		*OUT 71*	Store the control word to write display
9011		*MVI B,08*	Send through output port
9013		*MVI A,00*	Get the first data
9016		*OUT 70*	Send through output port
9018		*DCR B*	Give delay
901B		*JNZ 9012*	Go & get next data
901D		*MOV A,C*	Loop until all the data's have been taken
901E		*OUT 70*	Go to starting location
901F		*JMP 9019*	Store 16bit count value

FLOWCHART:


START
SETUP POINTER
INITIALIZE THE COUNTER
SET 8279 CHARECTER DISPLAY
SE 8279 CLEARING DISPLAY
WRITE COMMAND DISPLAY
DELAY

SEGMENT DEFINITION:

DATA BUS	D7	D6	D5	D4	D3	D2	D1	D0
SEGMENTS	d	c	B	A	d	g	f	e

RESULT:

Thus the rolling message "**2**" is displayed using 8279 interface kit.

EX. NO: 15

DATE :

INTERFACING ANALOG TO DIGITAL CONVERTER USING 8086

AIM:

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

APPARATUS REQUIRED:

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

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.

ALGORITHM:

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

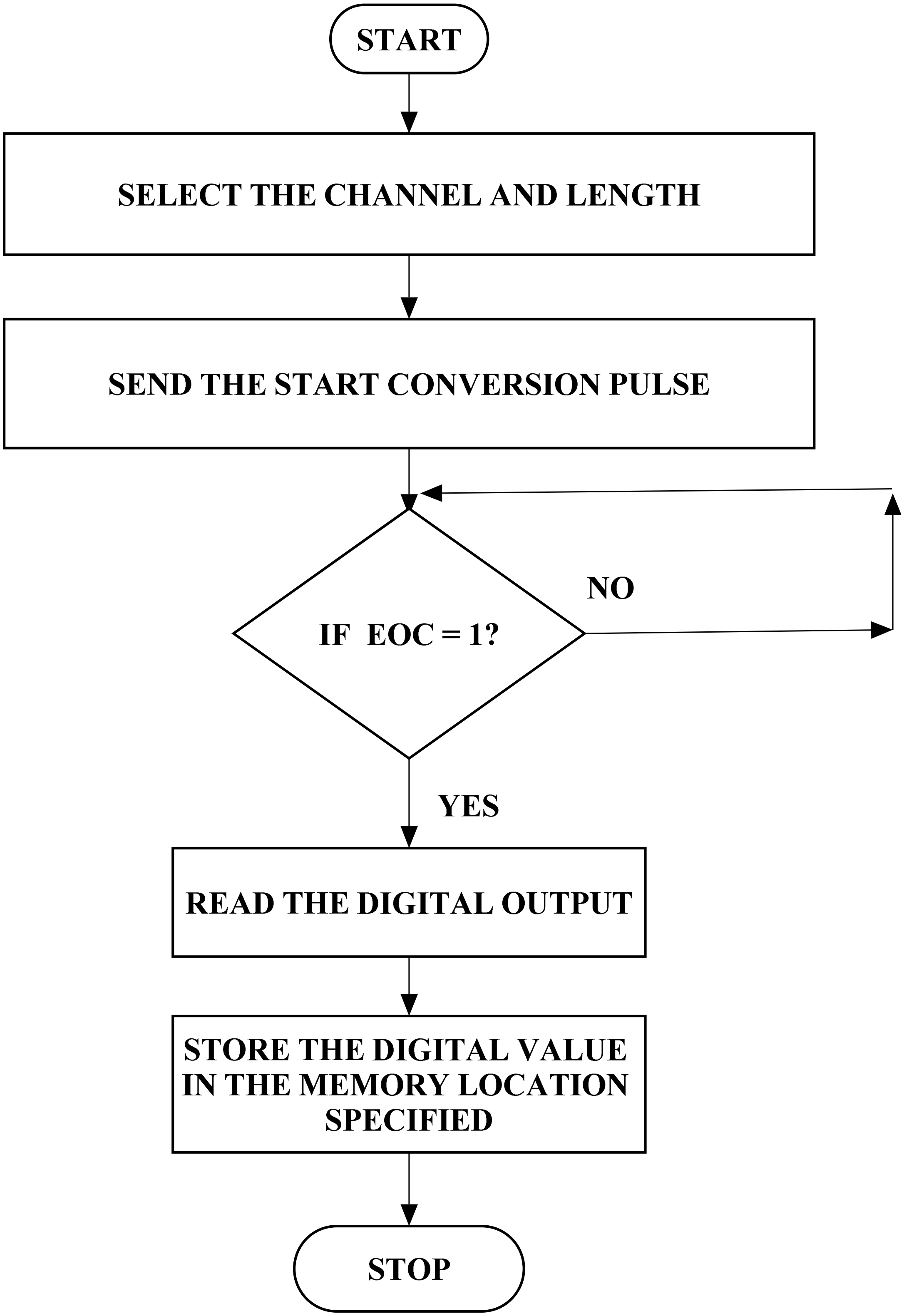
PROGRAM:

MEMORY LOCATION	OPCODES	PROGRAM	COMMENTS
1000		*MOV DX,FF26*	Load accumulator with value for ALE high
1000		*MOV AL,90*	Send through output port
1003		*OUT DX,AL*	Load accumulator with value for ALE low
1006		*MOV DX,FF24*	Send through output port
1009		*MOV AL,FF*	Store the value to make SOC high in the accumulator
100B		*OUT DX,AL*	Send through output port
100E		*MOV AL,00*	Introduce delay
1011		*OUT DX,AL*	
1013		*MOV AL,FF*	
1016		*OUT DX,AL*	
1018		*CALL 1100*	Store the value to make SOC low the accumulator
101B		*MOV DX,FF20*	Send through output port
101D		*IN AL,DX*	Read the EOC signal from port & check for end of conversion
101E		*INT 3*	Stop the program

DELAY SUBROUTINE PROGRAM

2100		***MOV CX,07FF***	Move the data 07ff to CX register
2103		***NOP***	No operation
2104		***NOP***	No operation
2105		***DEC CX***	Decrement CX register
2106		***JNZ 1103***	Jump no zero
2108		***RET***	Return to main address

FLOWCHART:


START
SELECT THE CHANNEL AND LENGTH
SEND THE START CONVERSION PULSE
IF EOC = 1?
NO
YES
READ THE DIGITAL OUTPUT
STORE THE DIGITAL VALUE IN THE MEMORY LOCATION SPECIFIED
STOP

OUTPUT:

HEX CODE IN MEMORY LOCATION	ANALOG VOLTAGE	DIGITAL DATA ON LED DISPLAY

RESULT:

Thus the ADC was interfaced with 8086 and the given analog inputs were converted into its digital equivalent.

EX. NO: 16

DATE :

INTERFACING DIGITAL – TO – ANALOG CONVERTER USING 8086

AIM:

To convert digital inputs into analog outputs and to generate different waveforms.

APPARATUS REQUIRED:

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

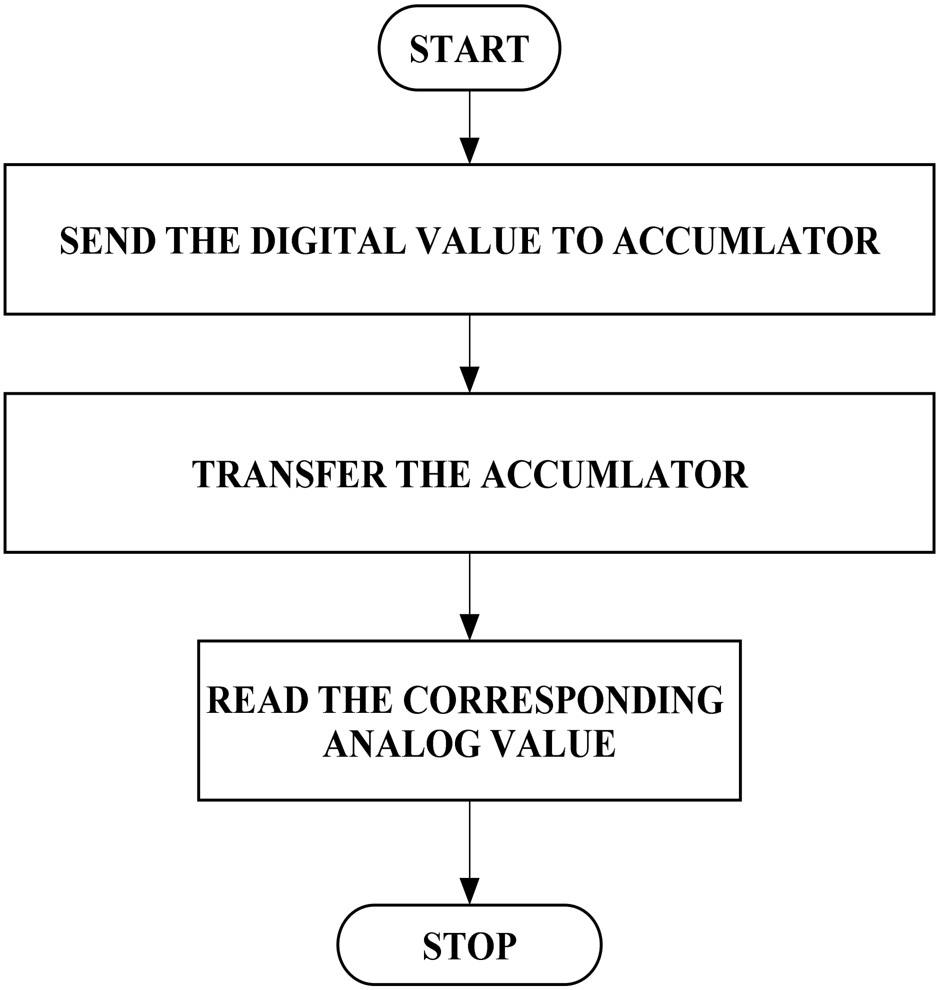
PROBLEM STATEMENT:

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

THEORY:

Since DAC 0800 is an 8 bit DAC and the output voltage variation is between –5v and +5v. The output voltage varies in steps of 10/256 = 0.04 (approximately). The digital data input and the corresponding output voltages are presented in the table. The basic idea behind the generation of waveforms is the continuous generation of analog output of DAC. With 00 (Hex) as input to DAC2 the analog output is –5v. Similarly with FF H as input, the output is +5v. Outputting digital data 00 and FF at regular intervals, to DAC2, results in a square wave of amplitude 5v.Output digital data from 00 to FF in constant steps of 01 to DAC2. Repeat this sequence again and again. As a result a saw-tooth wave will be generated at DAC2 output. Output digital data from 00 to FF in constant steps of 01 to DAC2. Output digital data from FF to 00 in constant steps of 01 to DAC2.

FLOECHART


START
SEND THE DIGITAL VALUE TO ACCUMLATOR
TRANSFER THE ACCUMLATOR
READ THE CORRESPONDING ANALOG VALUE
STOP

ALGORITHM

Measurement of analog voltage

(i) Send the digital value of DAC.
(ii) Read the corresponding analog value of its output.

Waveform generation

Square Waveform:

(i) Send low value (00) to the DAC.
(ii) Introduce suitable delay.
(iii) Send high value to DAC.
(iv) Introduce delay.
(v) Repeat the above
procedure. Saw-tooth waveform:

(i) Load low value (00) to accumulator.
(ii) Send this value to DAC.
(iii) Increment the accumulator.
(iv) Repeat step (ii) and (iii) until accumulator value reaches FF.
(v) Repeat the above procedure from step 1.
Triangular waveform:

(i) Load the low value (00) in accumulator.
(ii) Send this accumulator content to DAC.
(iii) Increment the accumulator.
(iv) Repeat step 2 and 3 until the accumulator reaches FF, decrement the accumulator and send the accumulator contents to DAC.

MEASUREMENT OF ANALOG VOLTAGE

DIGITAL DATA	ANALOG VOLTAGE

PROGRAME FOR DAC

MEMORY LOCATION	OPCODES	PROGRAM	COMMENTS
1000		***MOV DX,FF26***	Load accumulator with value for ALE high
1000		***MOV AL,80***	Send through output port
1003		***OUT DX,AL***	Load accumulator with value for ALE low
1006		***MOV DX,FF22***	Send through output port
1009		***MOV AL,FF***	Store the value to make SOC high in the accumulator
100B		***OUT DX,AL***	Send through output port
100E		***CALL 2100***	Introduce delay
1011		***MOV AL,00***	
1013		***OUT DX,AL***	
1016		***CALL 2100***	
1018		***JMP 2009***	Store the value to make SOC low the accumulator

DELAY SUOUTINEBR

2100		*MOV CX,07FF*	Move the data 07ff to CX register
2103		*NOP*	No operation
2104		*NOP*	No operation
2105		*DEC CX*	Decrement CX register
2106		*JNZ 2103*	Jump no zero
2108		*RET*	Return to main address

RESULT

Thus the DAC was interfaced with 8086 and different waveforms have been generated.

EX. NO: 17

DATE :

8 BIT ADDITION USING ARITHMETIC OPERATION 8051 MICROCONTROLLER

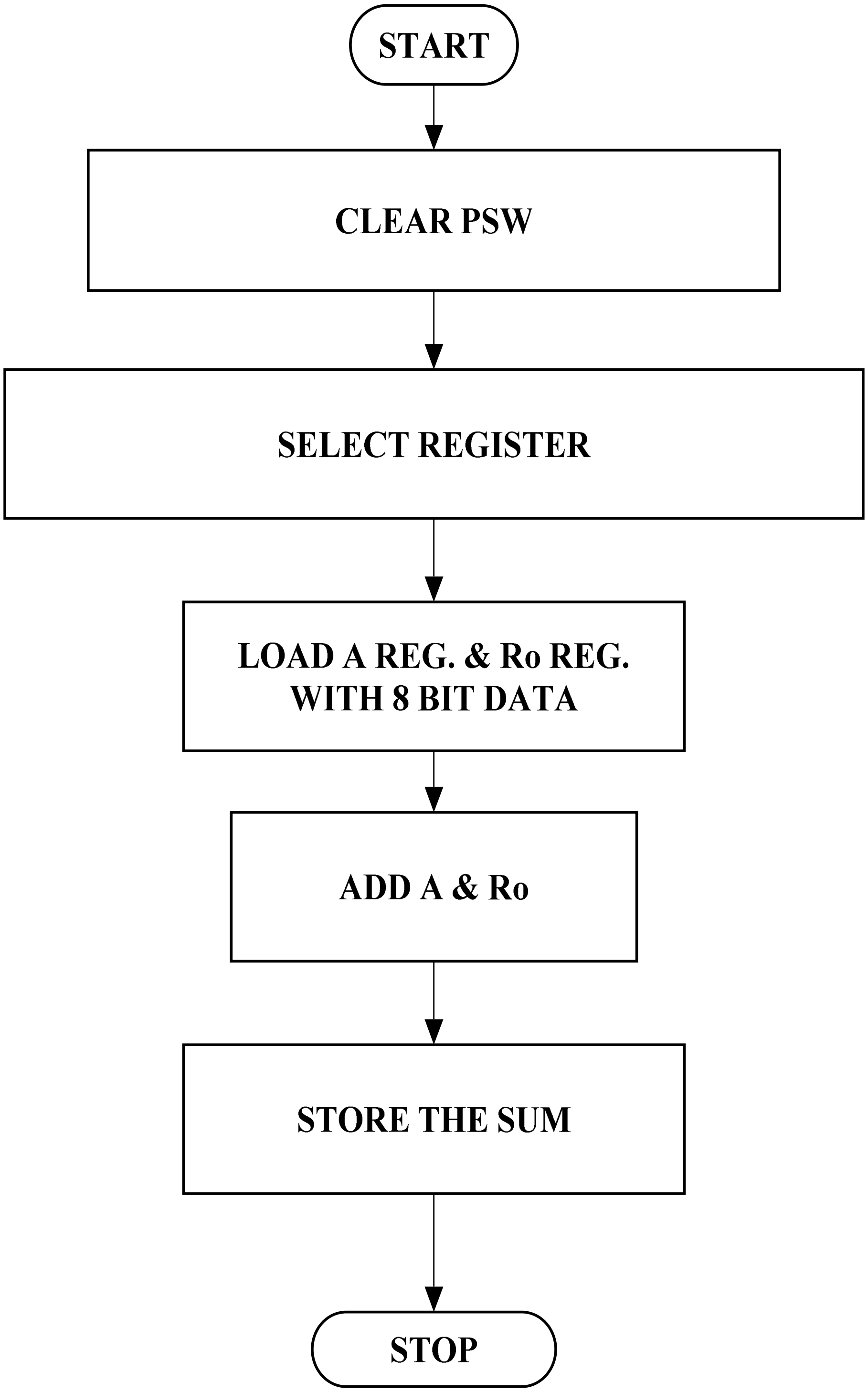
AIM:

To write an ALP program to add two 8-bit numbers using 8051 microcontroller.

ALGORITHM:

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

FLOW CHART


START
CLEAR PSW
SELECT REGISTER
LOAD A REG. & Ro REG.
WITH 8 BIT DATA
ADD A & Ro
STORE THE SUM
STOP

PROGRAM

ADDRESS	OPCODE	MNEMONIC	COMMENTS
8100		***MOV DPTR,#8300H***	Get the data1 in Accumulator
8101		***MOV X A,@DPTR***	Add the data1 with data2
8103		***MOV B,A***	Move the data A into B
8105		***INC DPTR***	Initialize the memory Location
8108		***MOV X A,@DPTR***	Move the data DPTR into A
8109		***ADD A,B***	Add A and B
8110		***INC X @DPTR,A***	Increment data
8111		***MOV X @DPTR,A***	Move the data A into B
8112		***LJMP 0000***	Stop the program

OUTPUT:

INPUT		**OUTPUT**	
MEMORY DATA		MEMORY DATA	

RESULT:

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

EX. NO: 18

DATE :

8 BIT SUBTRACTION USING ARITHMETIC OPERATION

8051 MICROCONTROLLER

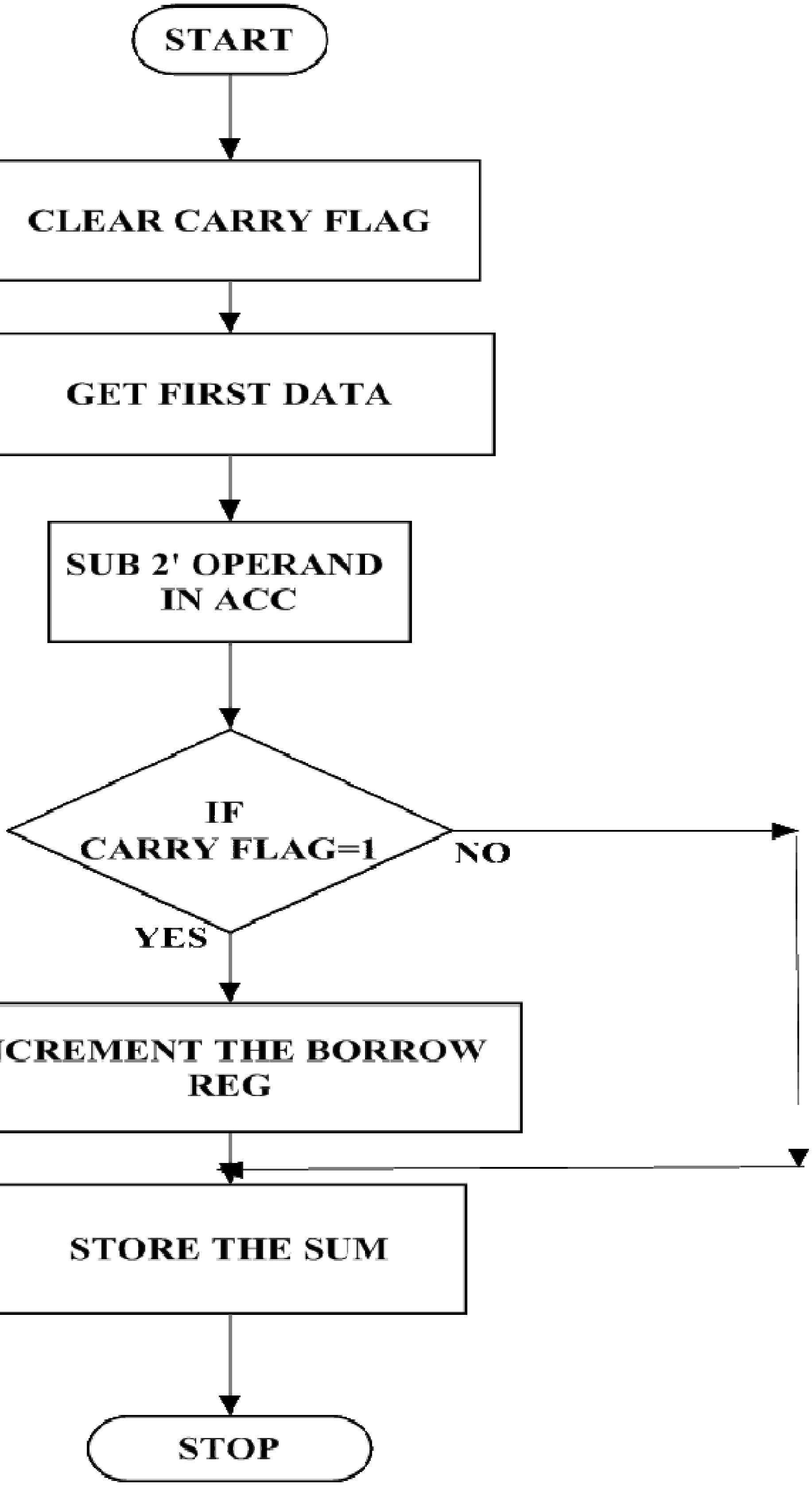
AIM:

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

ALGORITHM:

- Clear the carry flag.
- Initialize the register for borrow.
- Get the first operand into the accumulator.
- Subtract the second operand from the accumulator.
- If a borrow results increment the carry register.
- Store the result in memory.

FLOECHART:


START
CLEAR CARRY FLAG
GET FIRST DATA
SUB 2' OPERAND
IN ACC
IF
CARRY FLAG=1
NO
YES
INCREMENT THE BORROW
REG
STORE THE SUM
STOP

8 BIT SUBTRACTION

ADDRESS	OPCODE	MNEMONIC	COMMENTS
8100		***MOV DPTR,#8300H***	Get the data1 in Accumulator
8101		***MOV X A,@DPTR***	Add the data1 with data2
8103		***MOV B,A***	Move the data A into B
8105		***INC DPTR***	Initialize the memory Location
8108		***MOV X A,@DPTR***	Move the data DPTR into A
8109		***SUB B A,B***	Sub A and B
8110		***INC X @DPTR,A***	Increment data
8111		***MOV X @DPTR,A***	Move the data A into B
8112		***LJMP 0000***	Stop the program

OUTPUT:

INPUT		OUTPUT	
Memory Data		Memory Data	

RESULT:

Thus the 8051 ALP for subtraction of two 8 bit numbers is executed**.**

EX. NO: 19

DATE :

8 BIT MULTIPLICATION USING ARITHMETION OPERATION 8051 MICROCONTROLLER

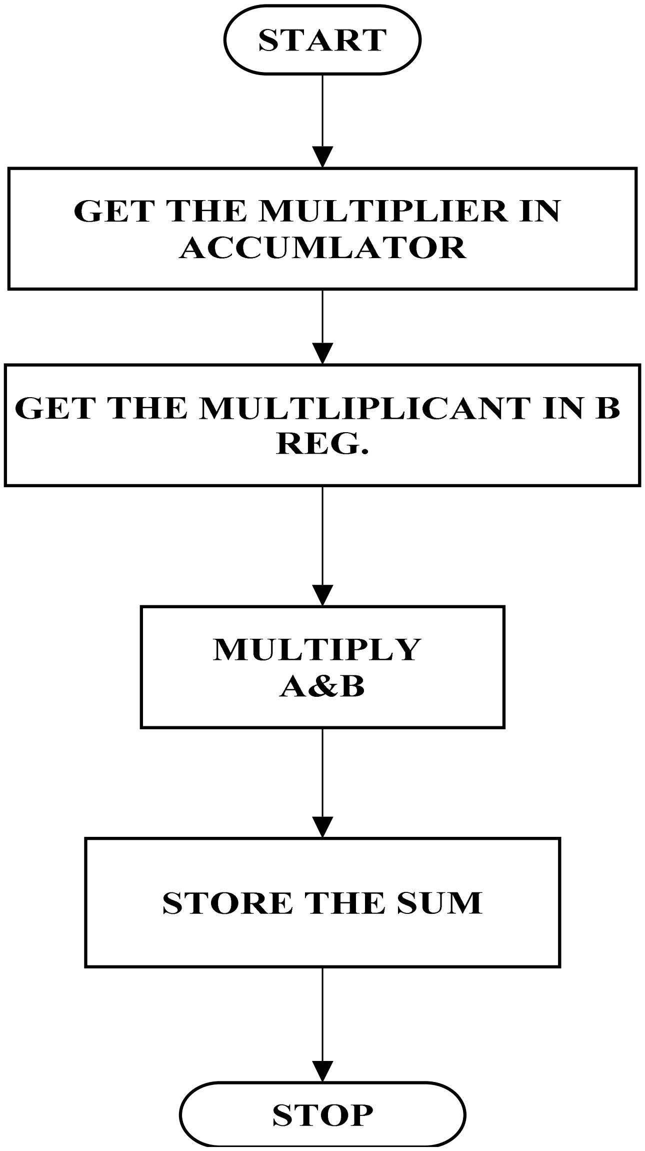
AIM:

To perform multiplication of two 8 bit data and store the result in memory.

ALGORITHM:

- Get the multiplier in the accumulator.
- Get the multiplicand in the B register.
- Multiply A with B.
- Store the product in memory.

FLOWCHART:


START
GET THE MULTIPLIER IN ACCUMLATOR
GET THE MULTLIPLICANT IN B REG.
MULTIPLY A&B
STORE THE SUM
STOP

8 BIT MULTIPLICATION

ADDRESS	OPCODE	MNEMONIC	COMMENTS
8100		***MOV DPTR,#8300H***	Get the data1 in Accumulator
8101		***MOV X A,@DPTR***	Add the data1 with data2
8103		***MOV B,A***	Move the data A into B
8105		***INC DPTR***	Initialize the memory Location
8108		***MOV X A,@DPTR***	Move the data DPTR into A
8109		***ADD A,B***	Sub A and B
8110		***INC DPTR***	Increment data
8111		***MOV X @DPTR,A***	Move the data A into B
8112		***SJMP 0000***	Stop the program

OUTPUT:

INPUT		OUTPUT	
Memory Location	Data	Memory location	Data
4500		4502	
4501		4503	

RESULT:

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

EX. NO: 20

DATE :

8 BIT DIVISION USING ARITHMETIC OPERATION 8051 MICROCONTROLLER

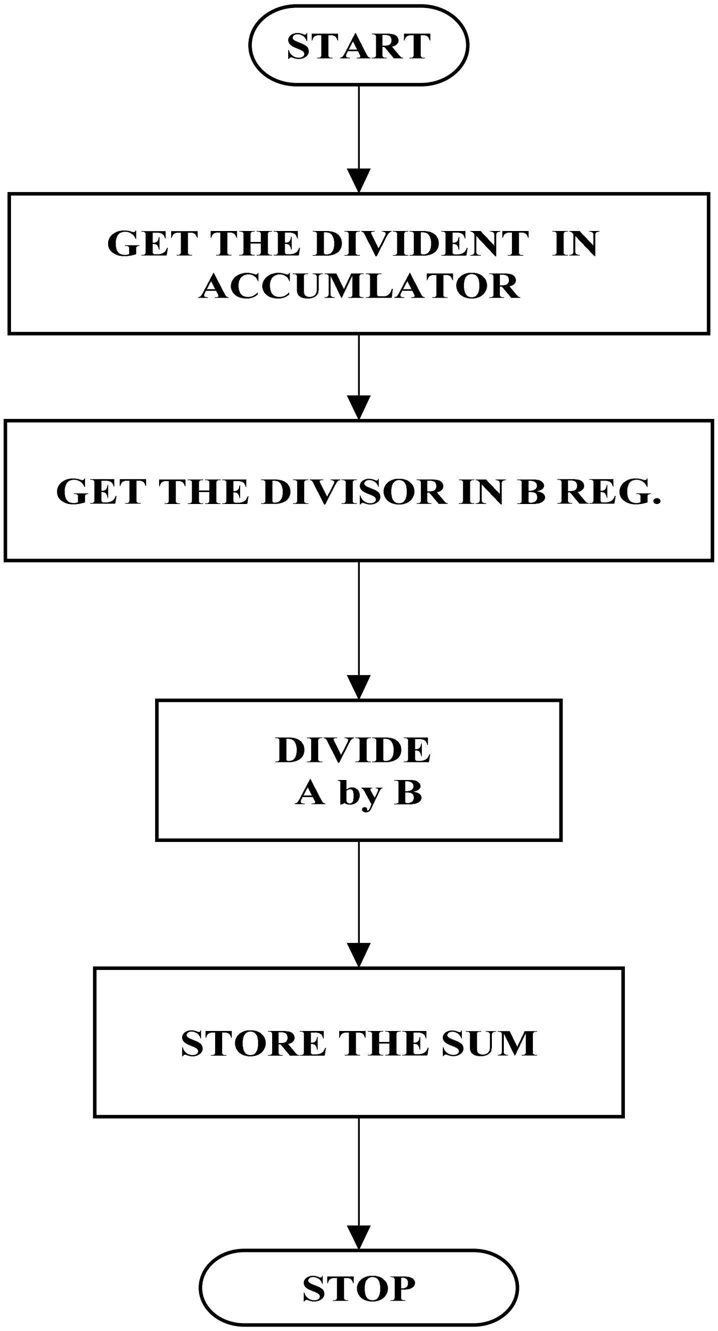
AIM:

To perform division of two 8 bit data and store the result in memory

ALGORITHM:

- Get the Dividend in the accumulator.
- Get the Divisor in the B register.
- Divide A by B.
- Store the Quotient and Remainder in memory.

FLOWCHART:


START
GET THE DIVIDENT IN ACCUMLATOR
GET THE DIVISOR IN B REG.
DIVIDE A by B
STORE THE SUM
STOP

8 BIT DIVISION

ADDRESS	OPCODE	MNEMONIC	COMMENTS
8100		***MOV DPTR,#8300H***	Get the data1 in Accumulator
8101		***MOV X A,@DPTR***	Add the data1 with data2
8103		***MOV B,A***	Move the data A into B
8105		***INC DPTR***	Initialize the memory Location
8108		***MOV X A,@DPTR***	Move the data DPTR into A
8109		***DIV A,B***	Div A and B
8110		***INC DPTR***	Increment data
8111		***MOV X @DPTR,A***	Move the data A into B
8112		***SJMP 0000***	Stop the program

OUTPUT:

INPUT		OUTPUT	
Memory Location	Data	Memory location	Data
4500		4502	
4501		4503	

RESULT:

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

EX. NO: 21

DATE :

LOGICAL OPERATIONS USING 8051 MICROCONTROLLER

AIM:

To perform logical operation using 8051 microcontroller AND, OR & EX-OR.

ALGORITHM:

- Get the input value and store data in the accumulator.
- Get the second values and store the B register.
- Logical operation to perform the given number
- Store the output value in memory.

PROGRAM FOR "*AND*" LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			***MOV DPTR,#9000h***	Move DPTR to 9000 Address
8003			***MOVX A,@DPTR***	Move XA register to DPTR
8007			***ANL A,#20***	AND Operation
800D			***INC DPTR***	Increment DPTR
800B			***MOV X @DPTR,A***	Move DPTR register to accumulator
8010			***SJMP 8008***	Copy the lower order data

PROGRAM FOR "*OR*" LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			***MOV DPTR,#9000***	Move DPTR to 9000 Address
8003			***MOVX A,@DPTR***	Move XA register to DPTR
8007			***ORL A,#20***	OR Operation
800D			***INC DPTR***	Increment DPTR
800B			***MOV X @DPTR,A***	Move DPTR register to accumulator
8010			***SJMP 8008***	Copy the lower order data

PROGRAM FOR "*EX- OR*" LOGIC

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
8000			MOV DPTR,#9000	Move DPTR to 9000 Address
8003			MOVX A,@DPTR	Move XA register to DPTR
8007			XRL A,#50	EX-OR Operation
800D			INC DPTR	Increment DPTR
800B			MOV X @DPTR,A	Move DPTR register to accumulator
8010			SJMP 8008	Copy the lower order data

OUTPUT:

GATE	INPUT	OUTPUT
AND		
OR		
EX-OR		

RESULT:

Thus the assembly language program to perform logical operations AND, OR & EX-OR using 8051 Performed and the result is stored.

EX. NO: 22

DATE :

FIND 2'S COMPLEMENT OF A NUMBER

AIM:

To Finding 2's complement of a number using 8051 micro controller

RESOURCES REQUIERED:

- 8051 microcontroller kit
- Keyboard
- Power supply

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
9000			*MOV DPRT,#9000*	Get the first data in AX register,
9003			*MOVX A,@DPTR*	Move the second data in DX register.
9007			*CPL A*	Compliment the higher order data**.**
900D			*ADD A,#01*	Move ax register into address
900B			*INC DPTR*	Inc DPTR
9010			*MOVX @DPTR,A*	Copy the lower order data
9012			*LJMP*	Store the higher order data.

OUTPUT:

INPUT DATA	OUTPUT DATA

RESULT;

Thus the program of finding 2’s complement of a number is done in 8051 microcontroller

EX. NO: 23

DATE :

COVERSION OF BCD TO ASCII

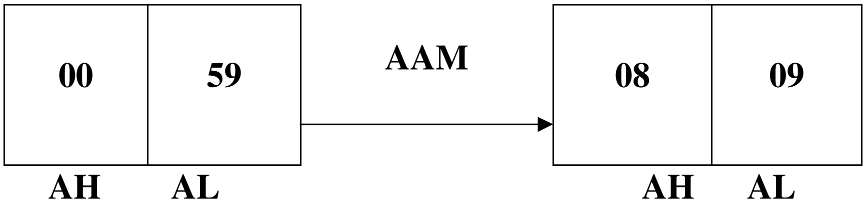
AIM:

To convert BCD number into ASCII by using 8051 micro controller

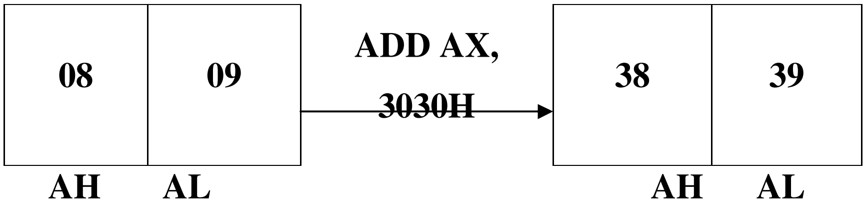
RESOURCES REQUIERED:

- 8051 microcontroller kit
- Keyboard
- Power supply

ALGORITHM:



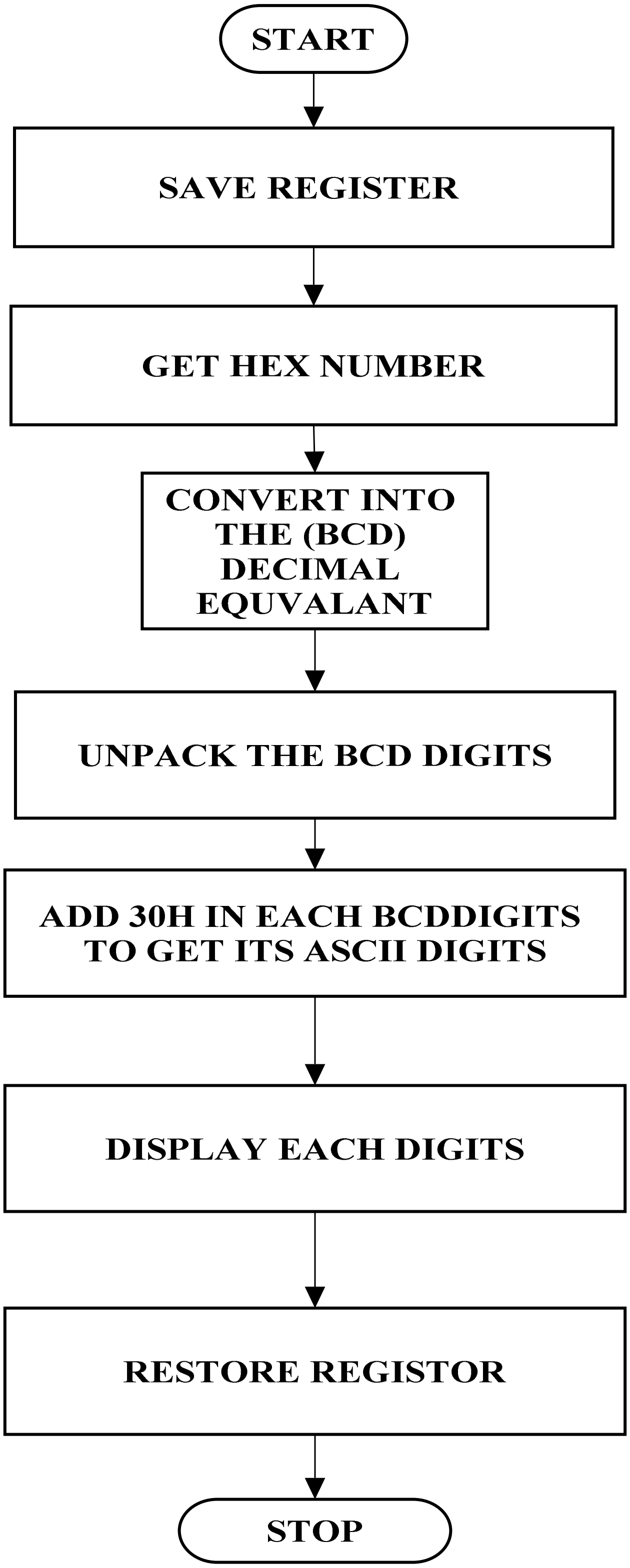
NOTE; 59H TO 89 DECIMAL



NOTE; 38h and 39h are the ASCII equivalents of 8 and 9 respectively

- Save contents of all registers which are used in the routine
- Get the data in al register and make AH equal to 00.
- Use AAM instruction to convert number in its decimal equivalent in the unpacked format.
- Add 30h in each digit to get its ASCII equivalent.
- Display one by one using function 2 of INT 21h.
- Routine content of register.

FLOWCHART:


START
SAVE REGISTER
GET HEX NUMBER
CONVERT INTO THE (BCD) DECIMAL EQUVALANT
UNPACK THE BCD DIGITS
ADD 30H IN EACH BCDDIGITS TO GET ITS ASCII DIGITS
DISPLAY EACH DIGITS
RESTORE REGISTOR
STOP

PROGRAM;

ROUTINE: convert binary for number less than 100 passing parameter

; Hex number in al register.

; Routine to convert binary number into its

; Decimal and then ADCII equivalent, and display the number

```
BTA  PROC  NEAR
     PUSH DX
     PUSH BX
     PUSH AX

     MOV AX, 00H

     AAM
     ADD AX, 3030H
     MOV BX, AX
     MOV DL,   BH
     MOV AH, 02
     INT 21H
MOV DL, BL
INT 21H

POP AX
POP BX
POP DX
RET
END P
```

RESULT:

Thus the given number is BCD number converted into ASCII using 8051 microcontroller kit.