CHENDU

COLLEGE OF ENGINEERING AND TECHNOLOGY

MADURANTAKAM.

KANCHEEPURAM-603311.

CS6412 MICROPROCESSOR AND MICROCONTROLLER

LABORATORY MANUAL

Prepared by

N.S.SHINIJA, ASSISTANT PROFESSOR

Faculty of Electronics & Communication Engineering

INDEX

BASIC ARITHMETIC AND LOGICAL OPERATIONS USING 8086 PROGRAMMING

EXPT NO: 01 **DATE:**

AIM:

To write an Assembly Language Program (ALP) for performing the Arithmetic operation of two byte numbers.

APPARATUS REQUIRED:

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

PROBLEM STATEMENT:

Write an ALP in 8086 to add and subtract two byte numbers stored in the memory location 1000H to 1003H and store the result in the memory location 1004H to 1005H.Also provide an instruction in the above program to consider the carry also and store the carry in the memory location 1006H.

ALGORITHM:

(i) **16-bit addition**

- Initialize the MSBs of sum to 0
- Get the first number.
- Add the second number to the first number.
- If there is any carry, increment MSBs of sum by 1.
- Store LSBs of sum.
- Store MSBs of sum.

(ii) 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.

(iii) Multiplication of 16-bit numbers:

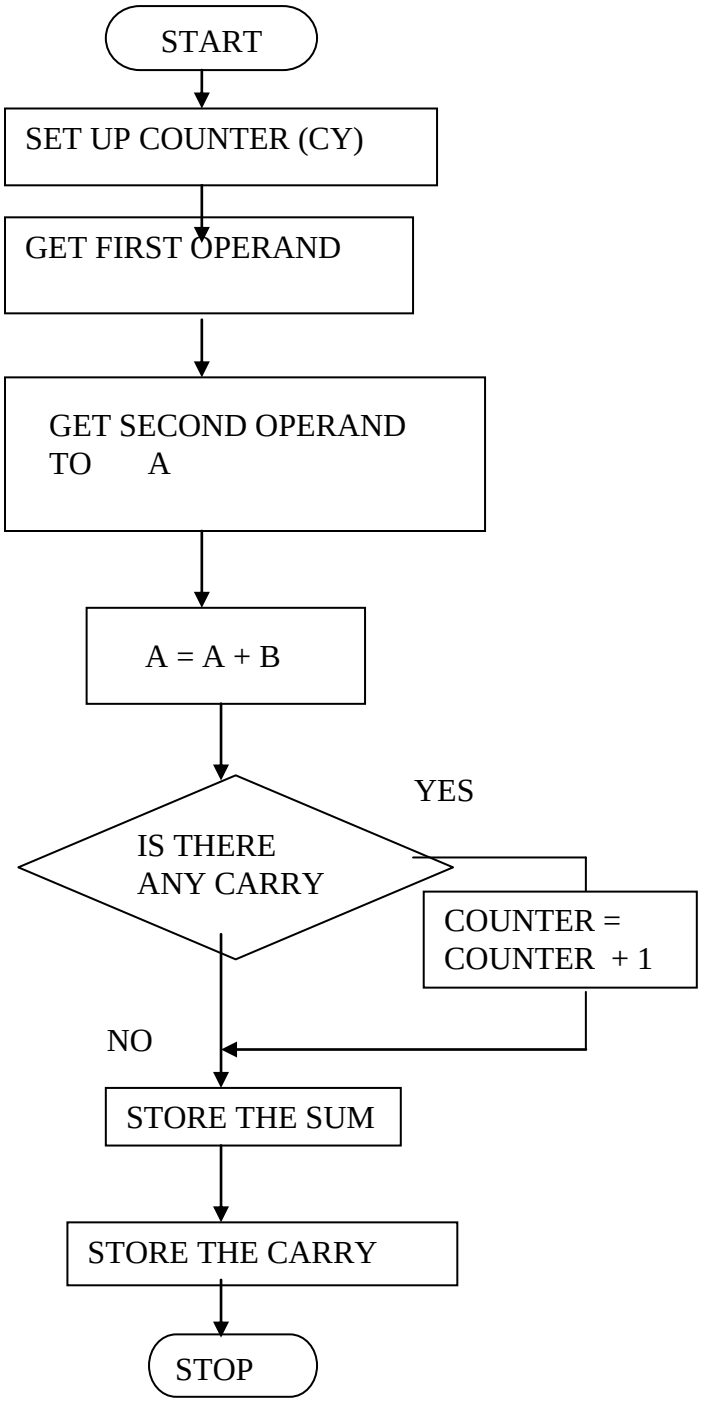
- Get the multiplier.
- Get the multiplicand
- Initialize the product to 0.
- Product = product + multiplicand
- Decrement the multiplier by 1
- If multiplicand is not equal to 0,repeat from step (d) otherwise store the product.

(iv) Division of 16-bit numbers.

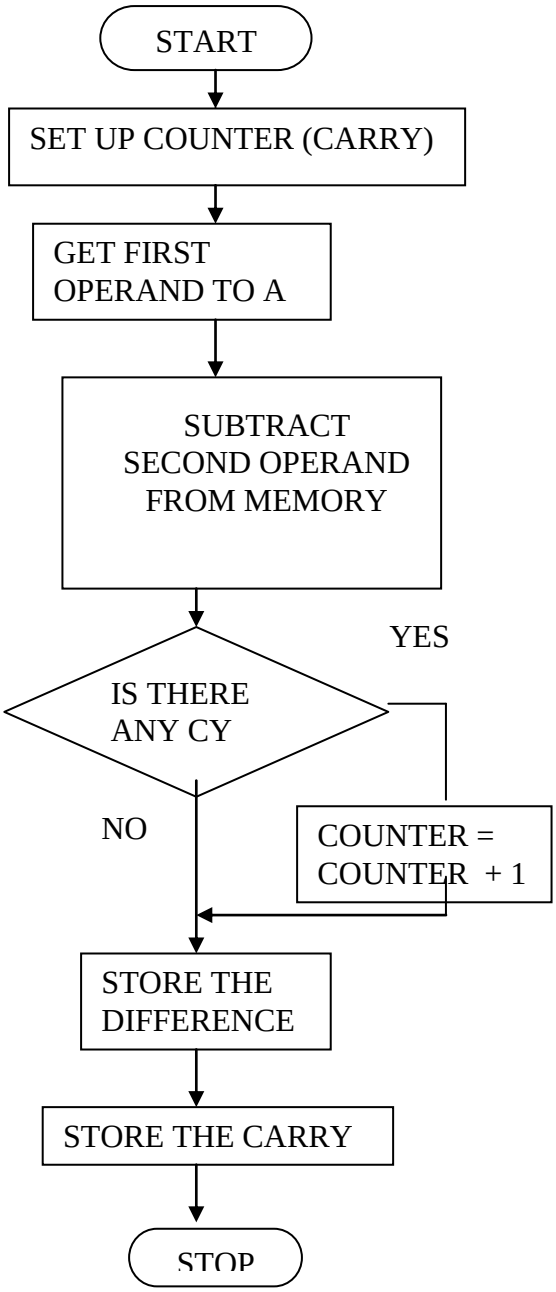
- Get the dividend
- Get the divisor
- Initialize the quotient to 0.
- Dividend = dividend – divisor
- If the divisor is greater, store the quotient. Go to step g.
- If dividend is greater, quotient = quotient + 1. Repeat from step (d)Store the dividend value as remainder.

FLOWCHART

ADDITION



SUBTRACTION



ADDITION

ADDRESS	Opcodes	PROGRAM	COMMENTS
		MOV CX, 0000H	Initialize counter CX
		MOV AX,[1200]	Get the first data in AX reg
		MOV BX, [1202]	Get the second data in BX reg
		ADD AX,BX	Add the contents of both the regs AX & BX
		JNC L1	Check for carry
		INC CX	If carry exists, increment the CX
		L1 : MOV [1206],CX	Store the carry
		MOV [1204], AX	Store the sum
		HLT	Stop the program

SUBTRACTION

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV CX, 0000H	Initialize counter CX
		MOV AX,[1200]	Get the first data in AX reg
		MOV BX, [1202]	Get the second data in BX reg
		SUB AX,BX	Subtract the contents of BX from AX
		JNC L1	Check for borrow
		INC CX	If borrow exists, increment the CX
		L1 : MOV [1206],CX	Store the borrow
		MOV [1204], AX	Store the difference
		HLT	Stop the program

RESULT:.

ADDITION

MEMORY						
DATA						

SUBTRACTION

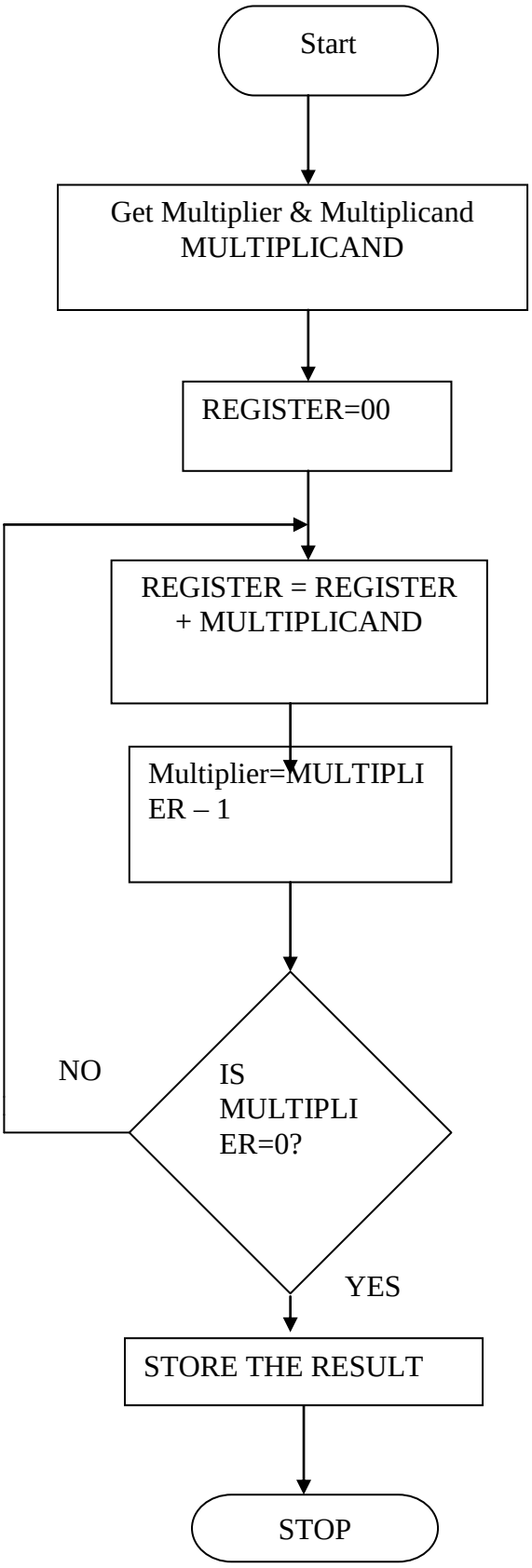
MEMORY						
DATA						

MANUAL CALCULATION

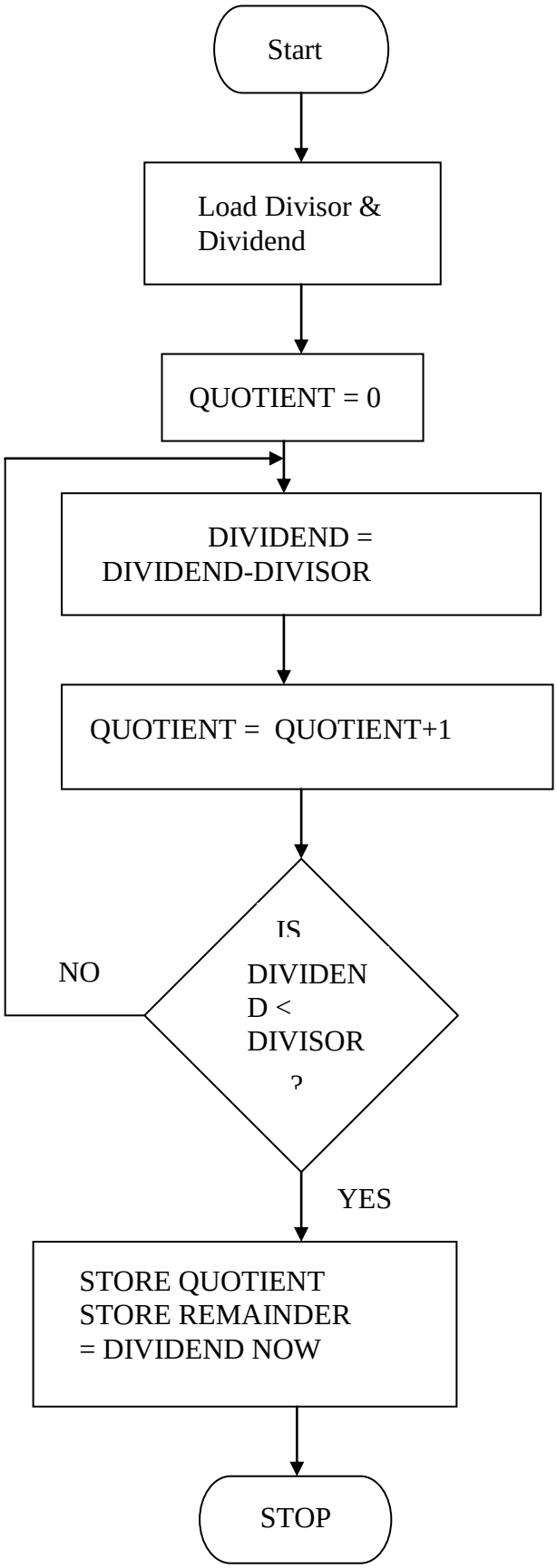
.

FLOWCHART

MULTIPLICATION



DIVISION



MULTIPLICATION

ADDRESS	Opcodes	PROGRAM	COMMENTS
		MOV AX,[1200]	Get the first data
		MOV BX, [1202]	Get the second data
		MUL BX	Multiply both
		MOV [1206],AX	Store the lower order product
		MOV AX,DX	Copy the higher order product to AX
		MOV [1208],AX	Store the higher order product
		HLT	Stop the program

DIVISION

ADDRESS	Opcodes	PROGRAM	COMMENTS
		MOV AX,[1200]	Get the first data
		MOV DX, [1202]	Get the second data
		MOV BX, [1204]	Divide the dividend by divisor
		DIV BX	Store the lower order product
		MOV [1206],AX	Copy the higher order product to AX
		MOV AX,DX	Store the higher order product
		MOV [1208],AX	Stop the program
		HLT	Get the first data

RESULT:.
MULTIPLICATION

MEMORY						
DATA						

DIVISON

MEMORY						
DATA						

MANUAL CALCULATION

Thus Arithmetic operation of two byte numbers are performed and the result is stored.

MOVE A DATA BLOCK WITHOUT OVERLAP

EXP.NO: 02 **DATE:**

AIM:

To convert a given Move a data block without overlap

.

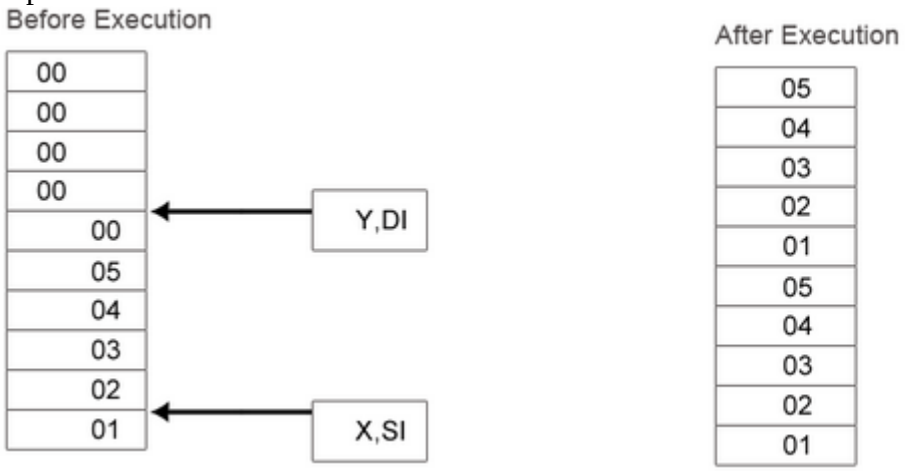
ALGORITHM:

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 Bregister.
6. Store the resultant in memory location.

PROGRAM:

```
DATA SEGMENT
X DB 01H,02H,03H,04H,05H ;Initialize Data Segments Memory Locations
Y DB 05 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA ; Initialize DS to point to start of the memory
MOV DS,AX ; set aside for storing of data
MOV CX,05H ; Load counter
LEA SI,X+04 ; SI pointer pointed to top of the memory block
LEA DI,X+04+03 ; 03 is displacement of over lapping, DI pointed to
;the top of the destination block
CODE ENDS
END START
```

Output:



RESULT:

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

CODE CONVERSION, DECIMAL ARITHMETIC AND MATRIX OPERATIONS.

EXP.NO: 03 CODE CONVERSIONS –DECIMAL TO HEX DATE:

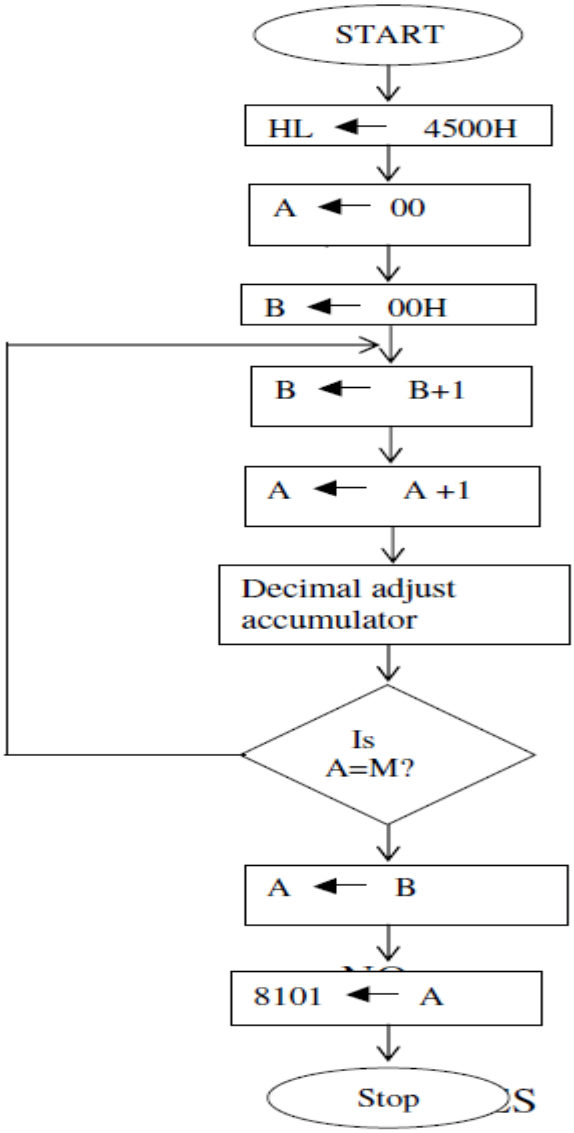
AIM:

To convert a given decimal number to hexadecimal.

ALGORITHM:

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

FLOWCHART :



PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
8000			LXI	H,8100	Initialize HL reg. to 8100H
8001					
8002					
8003			MVI	A,00	Initialize A register.
8004					
8005			MVI	B,00	Initialize B register..
8006					
8007		LOOP	INR	B	Increment B reg.
8008			ADI	01	Increment A reg
8009					
800A			DAA		Decimal Adjust Accumulator
800B			CMP	M	Compare M & A
800C			JNZ	LOOP	If acc and given number are not equal, then go to LOOP
800D					
800E					
800F			MOV	A,B	Transfer B reg to acc.
8010			STA	8101	Store the result in a memory location.
8011					
8012					
8013			HLT		Stop the program

INPUT			OUTPUT		
MEMORY					
DATA					

CODE CONVERSION –HEXADECIMAL TO DECIMAL

AIM:

To convert a given hexadecimal number to decimal.

ALGORITHM:

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

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENTS
8000			LXI	H,8100	Initialize HL reg. to 8100H
8001					
8002					
8003			MVI	A,00	Initialize A register.
8004					
8005			MVI	B,00	Initialize B register.
8006					
8007			MVI	C,00	Initialize C register for carry.
8008					
8009		LOOP	INR	B	Increment B reg.
800A			ADI	01	Increment A reg
800B					
800C			DAA		Decimal Adjust Accumulator
800D			JNC	NEXT	If there is no carry go to NEXT.
800E					

800F					
8010			INR	C	Increment c register.
8011		NEXT	MOV	D,A	Transfer A to D
8012			MOV	A,B	Transfer B to A
8013			CMP	M	Compare M & A
8014			MOV	A,D	Transfer D to A
8015			JNZ	LOOP	If acc and given number are not equal, then go to LOOP
8016					
8017					
8018			STA	8101	Store the result in a memory location.
8019					
801A					
801B			MOV	A,C	Transfer C to A
801C			STA	8102	Store the carry in another memory location.
801D					
801E					
801F			HLT		Stop the program

DECIMAL ARITHMETIC AND MATRIX OPERATIONS

AIM:

To write a program for addition of two matrix by using 8086.

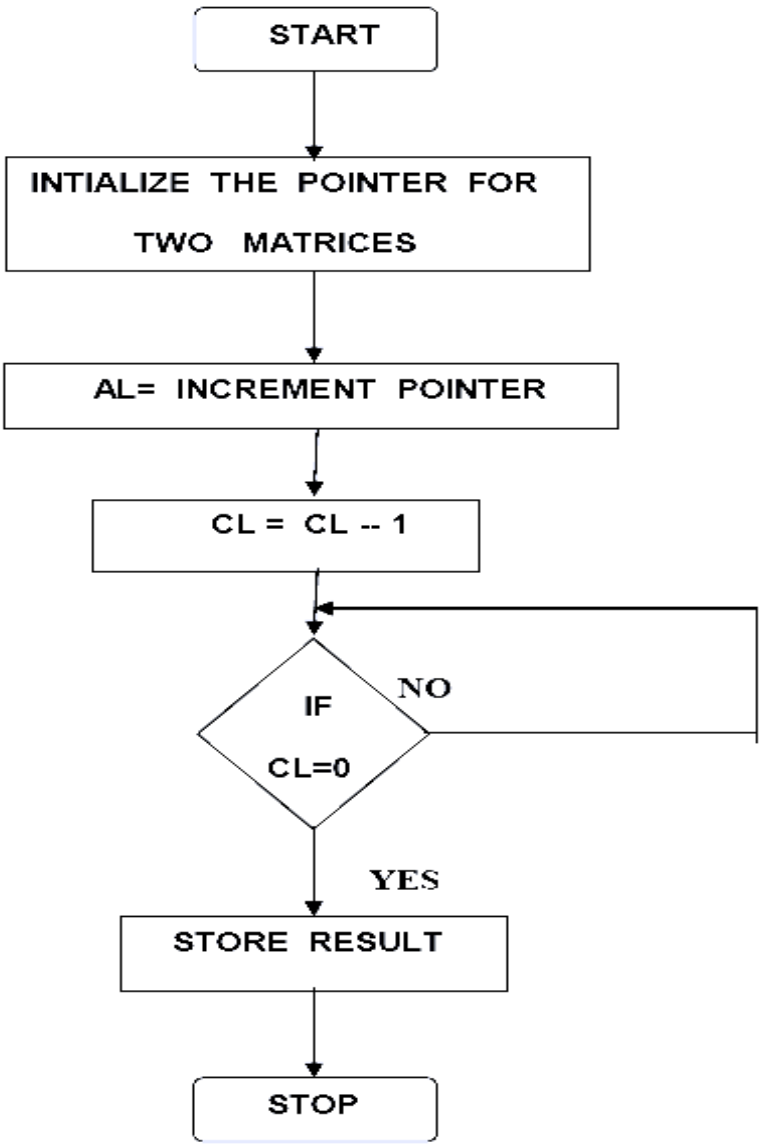
APPARATUS REQUIRED:

8086 Microprocessor Kit

ALGORITH:

1. Initialize the pointer only for data and result
2. Load AL with count
3. Add two matrix by each element
4. Process continues until CL is zero
5. Store result.

FLOWCHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
1500		MOV CL,09	BL,09	Count for 3×3 matrix
1502		MOV SI,2000	BE,00,20	Address in SI
1505		MOV DI,300	BF,00,30	Address in DI
1508		MOV AL,[SI]	BA,04	Load AL with matrix
150A		MOV BL,[DI]	8A,10	Load BL with matrix
150C		ADD AL,BL	00,08	Add two data
150E		MOV [DI],AL	88,05	Store result
1510		INC DI	4F	Increment DI
1511		INC SI	46	Increment SI
1512		DEC CL	FE,C9	Decrement CL
1514		JNZ 1508	75,F2	Loop continues until zero
1516		INT 3	CC	Break point

RESULT:

Thus the output for the addition for two matrix was executed successfully.

FLOATING POINT OPERATIONS, STRING MANIPULATIONS, SORTING AND SEARCHING

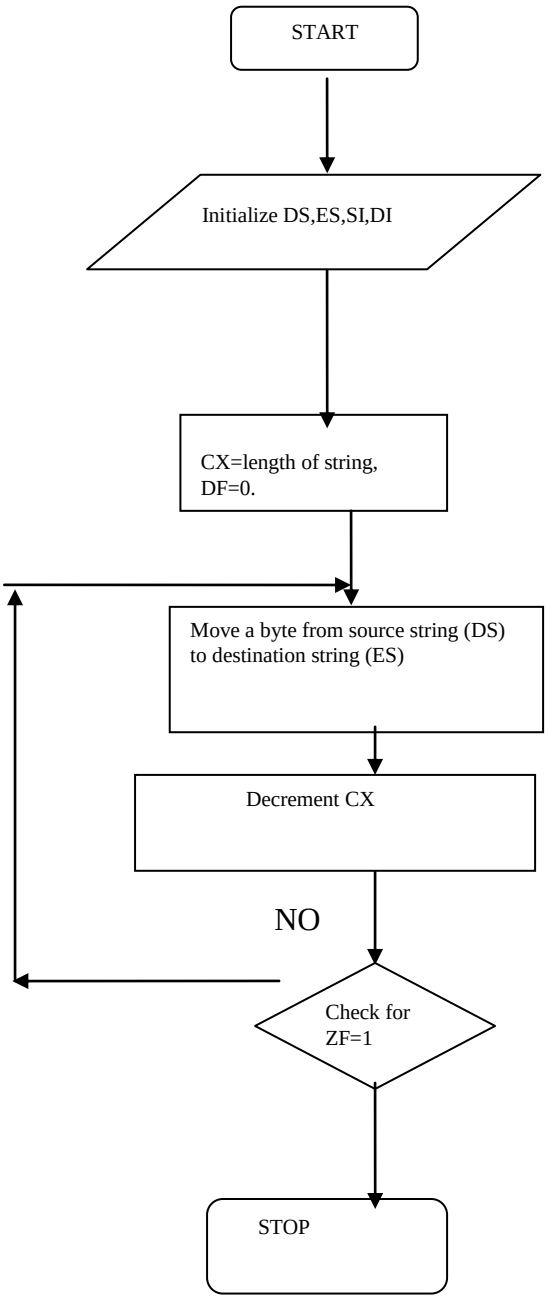
EXPT NO:04 **COPYING A STRING** **DATE:**

AIM:

To move a string of length FF from source to destination.

ALGORITHM:

a. Initialize the data segment .(DS)
b. Initialize the extra data segment .(ES)
c. Initialize the start of string in the DS. (SI)
d. Initialize the start of string in the ES. (DI)
e. Move the length of the string(FF) in CX register.
f. Move the byte from DS TO ES, till CX=0.



COPYING A STRING

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV SI,1200H	Initialize destination address
		MOV DI,1300H	Initialize starting address
		MOV CX,0006H	Initialize array size
		CLD	Clear direction flag
		REP MOVSB	Copy the contents of source into destination until count reaches zero
		HLT	Stop

RESULT:

INPUT						
MEMORY						
DATA						

OUTPUT						
MEMORY						
DATA						

Thus a string of a particular length is moved from source segment to destination segment

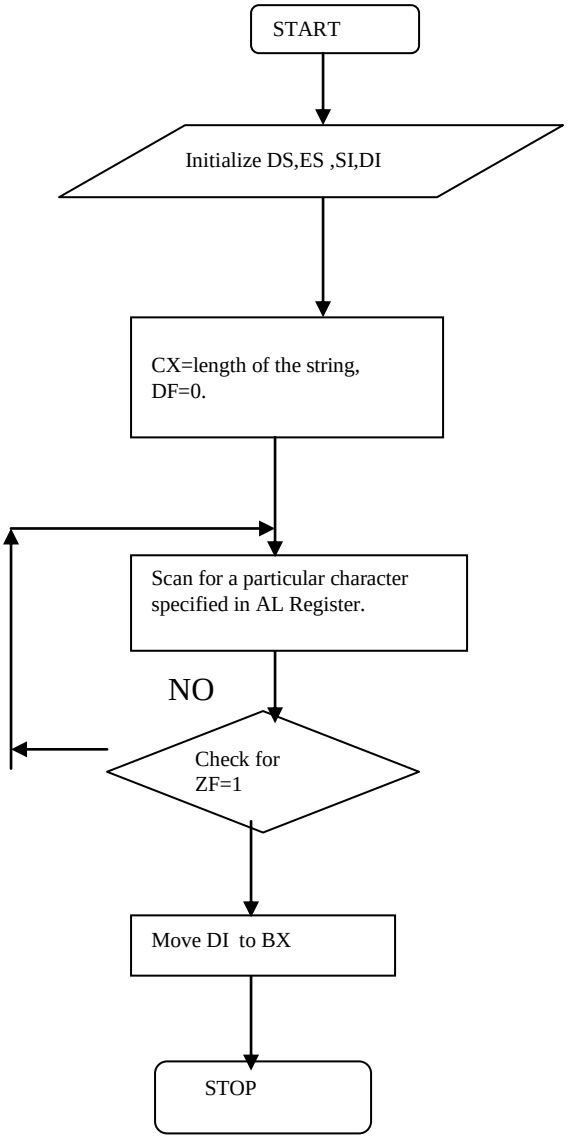
SEARCHING A STRING

AIM:

To scan for a given byte in the string and find the relative address of the byte from the starting location of the string.

ALGORITHM:

a. Initialize the extra segment .(ES)
b. Initialize the start of string in the ES. (DI)
c. Move the number of elements in the string in CX register.
d. Move the byte to be searched in the AL register.
e. Scan for the byte in ES. If the byte is found ZF=0, move the address pointed by ES:DI to BX.



SEARCHING FOR A CHARACTER IN THE STRING

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV DI,1300H	Initialize destination address
		MOV SI, 1400H	Initialize starting address
		MOV CX, 0006H	Initialize array size
		CLD	Clear direction flag
		MOV AL, 08H	Store the string to be searched
		REPNE SCASB	Scan until the string is found
		DEC DI	Decrement the destination address
		MOV BL,[DI]	Store the contents into BL reg
		MOV [SI],BL	Store content of BL in source address
		HLT	Stop

RESULT:

INPUT						
MEMORY						
DATA						

OUTPUT	
MEMORY LOCATION	
DATA	

Thus a given byte or word in a string of a particular length in the extra segment(destination) is found .

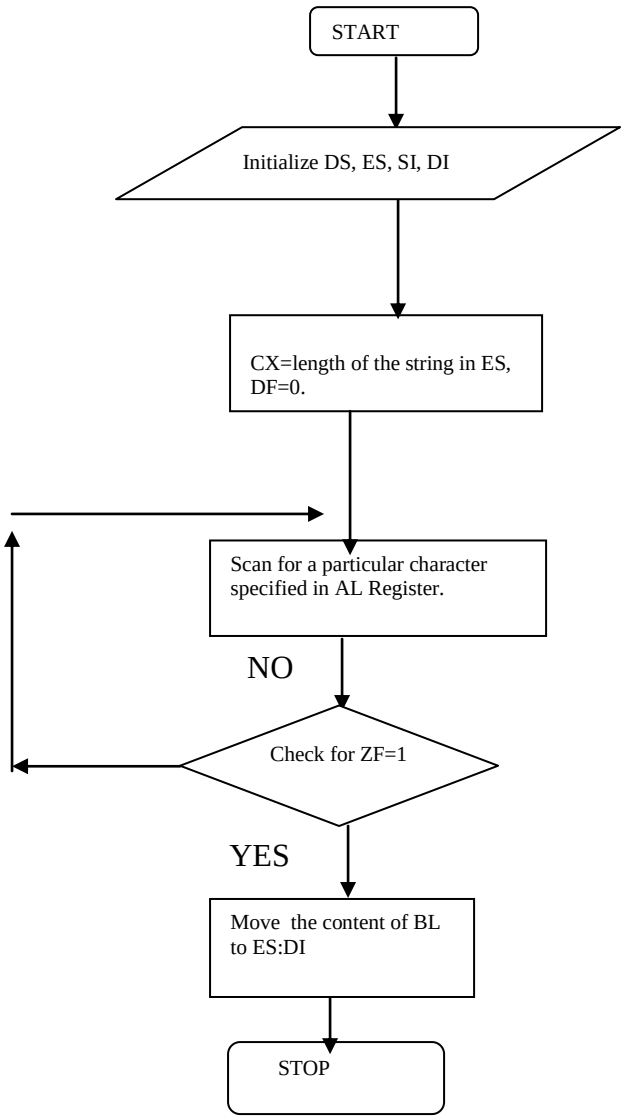
FIND AND REPLACE

AIM:

To find a character in the string and replace it with another character.

ALGORITHM:

a. Initialize the extra segment .(E S)
b. Initialize the start of string in the ES. (DI)
c. Move the number of elements in the string in CX register.
d. Move the byte to be searched in the AL register.
e. Store the ASCII code of the character that has to replace the scanned byte in BL register.
f. Scan for the byte in ES. If the byte is not found, ZF≠1 and repeat scanning.
g. If the byte is found, ZF=1.Move the content of BL register to ES:DI.



FIND AND REPLACE A CHARACTER IN THE STRING

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV DI,1300H	Initialize destination address
		MOV SI,1400H	Initialize starting address
		MOV CX, 0006H	Initialize array size
		CLD	Clear direction flag
		MOV AL, 08H	Store the string to be searched
		MOV BH,30H	Store the string to be replaced
		REPNE SCASB	Scan until the string is found
		DEC DI	Decrement the destination address
		MOV BL,[DI]	Store the contents into BL reg
		MOV [SI],BL	Store content of BL in source address
		MOV [DI],BH	Replace the string
		HLT	Stop

RESULT:

INPUT						
MEMORY						
DATA						

OUTPUT						
MEMORY						
DATA						

Thus a given byte or word in a string of a particular length in the extra segment(destination) is found and is replaced with another character.

ASCENDING & DESCENDING

AIM:

To write an Assembly Language Program (ALP) to sort a given array in ascending and descending order.

APPARATUS REQUIRED:

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

PROBLEM STATEMENT:

An array of length 10 is given from the location. Sort it into descending and ascending order and store the result.

ALGORITHM:

(i) Sorting in ascending order:

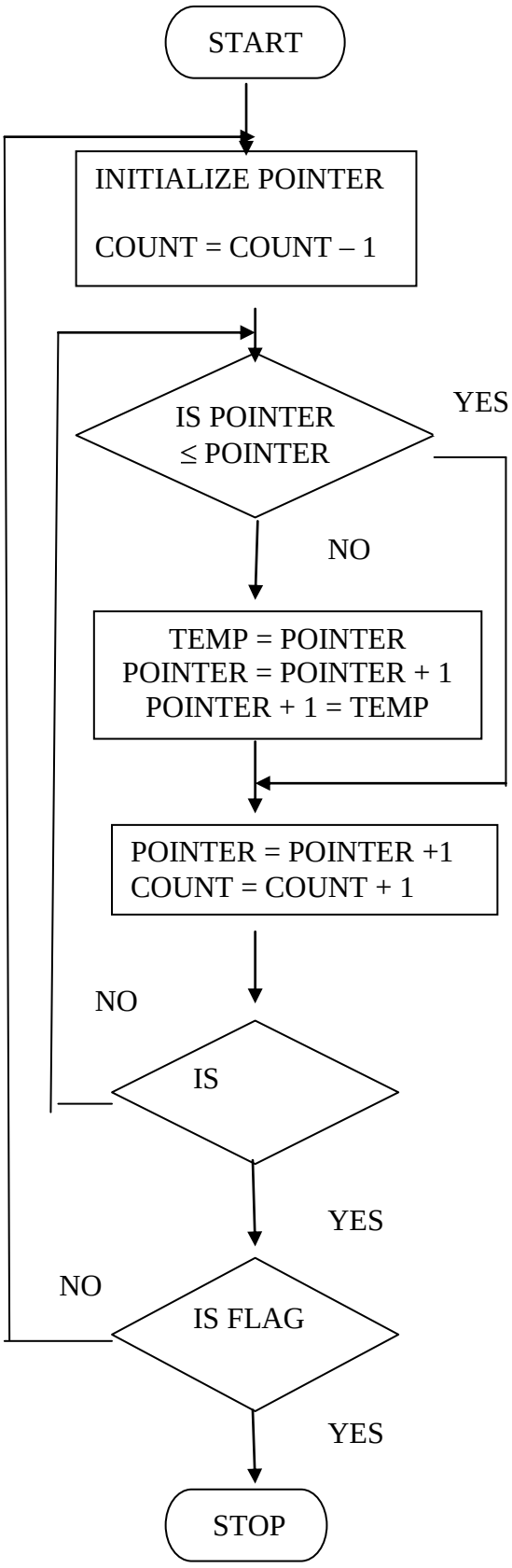
a. Load the array count in two registers C_1 and C_2.
b. Get the first two numbers.
c. Compare the numbers and exchange if necessary so that the two numbers are in ascending order.
d. Decrement C_2.
e. Get the third number from the array and repeat the process until C_2 is 0.
f. Decrement C_1 and repeat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

(ii) Sorting in descending order:

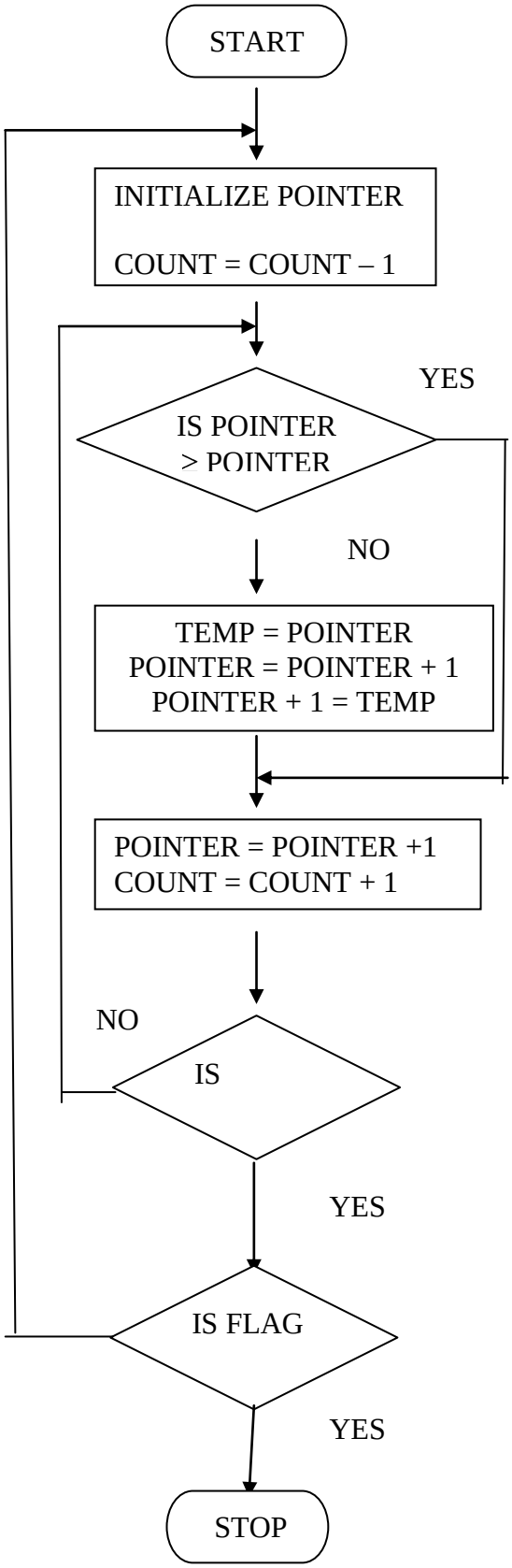
a. Load the array count in two registers C_1 and C_2.
b. Get the first two numbers.
c. Compare the numbers and exchange if necessary so that the two numbers are in descending order.
d. Decrement C_2.
e. Get the third number from the array and repeat the process until C_2 is 0.
f. Decrement C_1 and repeat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

FLOWCHART

ASCENDING ORDER



DESCENDING ORDER



ASCENDING

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV SI,1200H	Initialize memory location for array size
		MOV CL,[SI]	Number of comparisons in CL
		L4 : MOV SI,1200H	Initialize memory location for array size
		MOV DL,[SI]	Get the count in DL
		INC SI	Go to next memory location
		MOV AL,[SI]	Get the first data in AL
		L3 : INC SI	Go to next memory location
		MOV BL,[SI]	Get the second data in BL
		CMP AL,BL	Compare two data's
		JNB L1	If AL < BL go to L1
		DEC SI	Else, Decrement the memory location
		MOV [SI],AL	Store the smallest data
		MOV AL,BL	Get the next data AL
		JMP L2	Jump to L2
		L1 : DEC SI	Decrement the memory location
		MOV [SI],BL	Store the greatest data in memory location
		L2 : INC SI	Go to next memory location
		DEC DL	Decrement the count
		JNZ L3	Jump to L3, if the count is not reached zero
		MOV [SI],AL	Store data in memory location
		DEC CL	Decrement the count
		JNZ L4	Jump to L4, if the count is not reached zero
		HLT	Stop

DESCENDING

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV SI,1200H	Initialize memory location for array size
		MOV CL,[SI]	Number of comparisons in CL
		L4 : MOV SI,1200H	Initialize memory location for array size
		MOV DL,[SI]	Get the count in DL
		INC SI	Go to next memory location
		MOV AL,[SI]	Get the first data in AL
		L3 : INC SI	Go to next memory location

		MOV BL,[SI]	Get the second data in BL
		CMP AL,BL	Compare two data's
		JB L1	If AL > BL go to L1
		DEC SI	Else, Decrement the memory location
		MOV [SI],AL	Store the largest data
		MOV AL,BL	Get the next data AL
		JMP L2	Jump to L2
		L1 : DEC SI	Decrement the memory location
		MOV [SI],BL	Store the smallest data in memory location
		L2 : INC SI	Go to next memory location
		DEC DL	Decrement the count
		JNZ L3	Jump to L3, if the count is not reached zero
		MOV [SI],AL	Store data in memory location
		DEC CL	Decrement the count
		JNZ L4	Jump to L4, if the count is not reached zero
		HLT	Stop

RESULT:.

ASCENDING

MEMORY						
DATA						

DESCENDING

MEMORY						
DATA						

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

LARGEST& SMALLEST

AIM:

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

APPARATUS REQUIRED:

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

PROBLEM STATEMENT:

An array of length 10 is given from the location. Find the largest and smallest number and store the result.

ALGORITHM:

(i) Finding largest number:

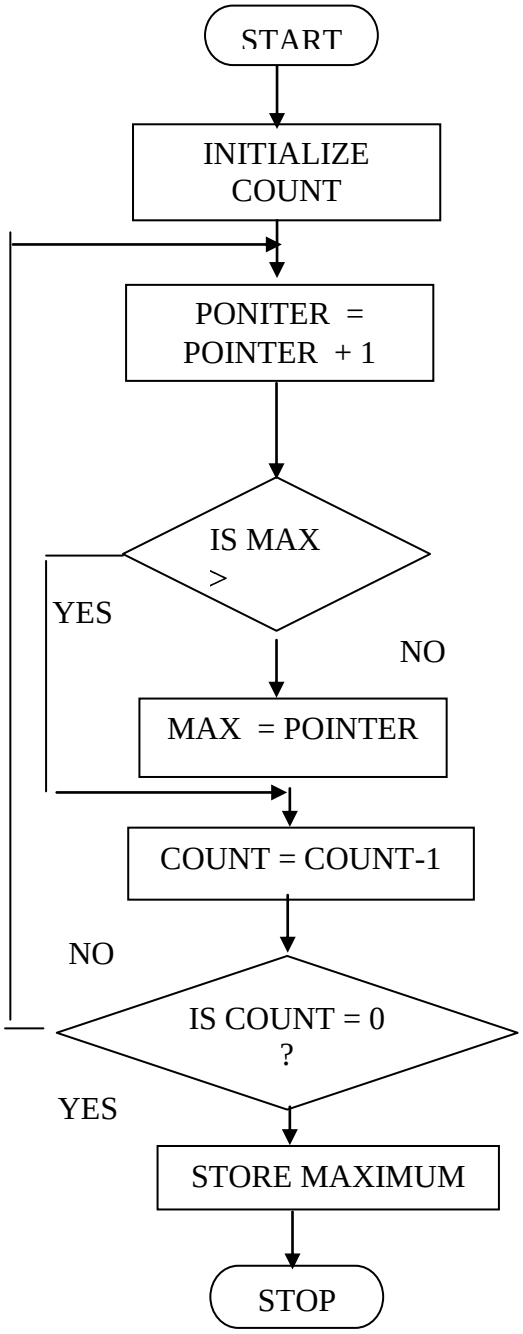
a. Load the array count in a register C_1.
b. Get the first two numbers.
c. Compare the numbers and exchange if the number is small.
d. Get the third number from the array and repeat the process until C_1 is 0.

(ii) Finding smallest number:

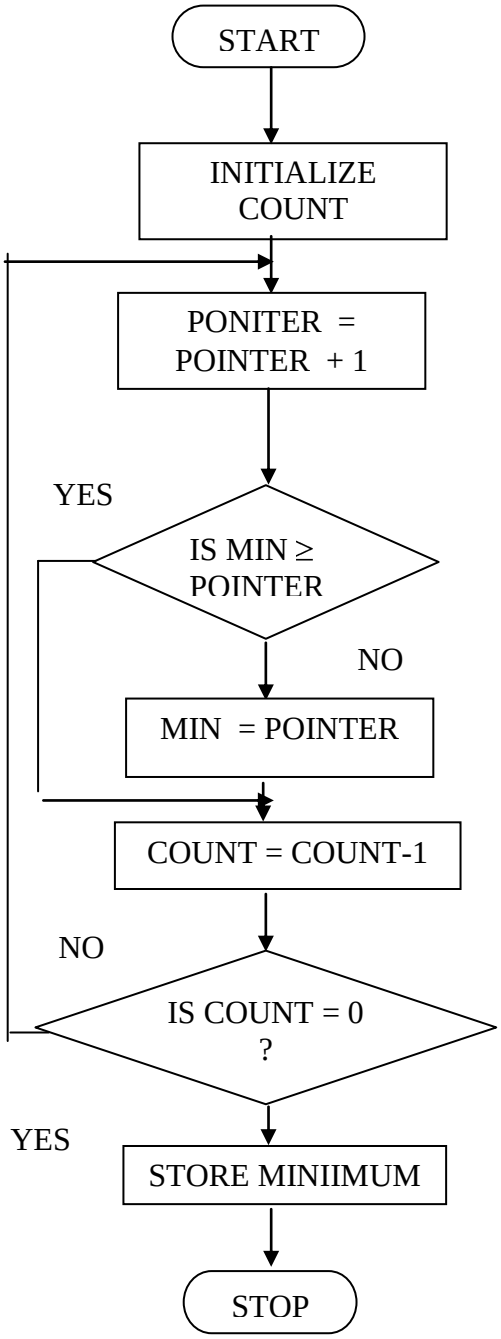
e. Load the array count in a register C1.
f. Get the first two numbers.
g. Compare the numbers and exchange if the number is large.
h. Get the third number from the array and repeat the process until C1 is 0.

FLOWCHART

LARGEST NUMBER IN AN ARRAY



SMALLEST NUMBER IN AN ARRAY



LARGEST

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV SI,1200H	Initialize array size
		MOV CL,[SI]	Initialize the count
		INC SI	Go to next memory location
		MOV AL,[SI]	Move the first data in AL
		DEC CL	Reduce the count
		L2 : INC SI	Move the SI pointer to next data
		CMP AL,[SI]	Compare two data's
		JNB L1	If AL > [SI] then go to L1 (no swap)
		MOV AL,[SI]	Else move the large number to AL
		L1 : DEC CL	Decrement the count
		JNZ L2	If count is not zero go to L2
		MOV DI,1300H	Initialize DI with 1300H
		MOV [DI],AL	Else store the biggest number in 1300 location
		HLT	Stop

SMALLEST

ADDRESS	OPCODES	PROGRAM	COMMENTS
		MOV SI,1200H	Initialize array size
		MOV CL,[SI]	Initialize the count
		INC SI	Go to next memory location
		MOV AL,[SI]	Move the first data in AL
		DEC CL	Reduce the count
		L2 : INC SI	Move the SI pointer to next data
		CMP AL,[SI]	Compare two data's
		JB L1	If AL < [SI] then go to L1 (no swap)
		MOV AL,[SI]	Else move the large number to AL
		L1 : DEC CL	Decrement the count
		JNZ L2	If count is not zero go to L2
		MOV DI,1300H	Initialize DI with 1300H
		MOV [DI],AL	Else store the biggest number in 1300 location
		HLT	Stop

RESULT:.

LARGEST

MEMORY						
DATA						

SMALLEST

MEMORY						
DATA						

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

PASSWORD CHECKING, PRINT RAM SIZE AND SYSTEM DATE

EXPT NO:05 **DATE:**

AIM:

To write an Assembly Language Program (ALP) for performing the Arithmetic operation of two byte numbers

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
```

```
;Today.asm Display month/day/year.
;   Feb 1st, 2012
;CIS 206  Ken Howard
        .MODEL small
        .STACK 100h
        .DATA
        mess1 DB  10, 13, 'Today is $'  ; 10=LF, 13=CR
        .CODE

 Today PROC
        MOV AX, @data
       MOV DS, AX
        MOV DX, OFFSET mess1    ; Move string to DX
       MOV AH, 09h             ; 09h call to display string (DX > AH > DOS)
       INT 21H               ; Send to DOS
       ; CX year, DH month, DL day
       MOV AH, 2AH             ; Get the date (appendix D)
       INT 21H               ; Send to DOS
        PUSH CX                ; Move year to the stack
       MOV CX, 0              ; Clear CX

 MOV CL, DL
       PUSH CX               ; Move day to stack
        MOV CL, DH              ; Move month > CL
       PUSH CX               ; Move month to stack
  MOV DH, 0             ; Clear DH
  ; ************************* DISPLAY MONTH ************************
```

```
      ; Set up for division
        ; Dividend will be in DX/AX pair (4 bytes)
                        ; Quotient will be in AX
                        ; Remainder will be in DX
       MOV DX, 0              ; Clear DX
      POP AX                ; Remove month from stack into AX
       MOV CX, 0              ; Initialize the counter
      MOV BX, 10             ; Set up the divisor
dividem:
       DIV BX                ; Divide (will be word sized)
      PUSH DX                ; Save remainder to stack
       ADD CX, 1              ; Add one to counter
      MOV DX, 0              ; Clear the remainder
      CMP AX, 0              ; Compare quotient to zero
      JNE dividem            ; If quoient is not zero, go to "dividem:"
divdispm:

      POP DX                ; Remove top of stack into DX
       ADD DL, 30h            ; ADD 30h (2) to DL
      MOV AH, 02h            ; 02h to display AH (DL)
      INT 21H               ; Send to DOS
       LOOP divdispm          ; If more to do, divdispm again
                        ; LOOP subtracts 1 from CX. If non-zero, loop.
       MOV DL, '/'            ; Character to display goes in DL
      MOV AH, 02h            ; 02h to display AH (DL)
      INT 21H               ; Send to DOS
; ************************** DISPLAY DAY ************************
      ; Set up for division
                        ; Dividend will be in DX/AX pair (4 bytes)
                        ; Quotient will be in AX
                        ; Remainder will be in DX
      MOV DX, 0              ; Clear DX
      POP AX                ; Remove day from stack into AX
      MOV CX, 0              ; Initialize the counter
      MOV BX, 10             ; Set up the divisor
divided:
       DIV BX                ; Divide (will be word sized)
      PUSH DX                ; Save remainder to stack

      ADD CX, 1              ; Add one to counter
      MOV DX, 0              ; Clear the remainder
      CMP AX, 0              ; Compare quotient to zero
      JNE divided            ; If quoient is not zero, go to "divided:"
divdispd:
       POP DX                ; Remove top of stack
      ADD DL, 30h            ; ADD 30h (2) to DL
      MOV AH, 02h            ; 02h to display AH (DL)
      INT 21H               ; Send to DOS
       LOOP divdispd          ; If more to do, divdispd again
                        ; LOOP subtracts 1 from CX. If non-zero, loop.
```

```
      MOV DL, '/'          ; Character to display goes in DL
      MOV AH, 02h          ; 02h to display AH (DL)
      INT 21H              ; Send to DOS
 ; ************************* DISPLAY YEAR ************************
 ; Set up for division
                           ; Dividend will be in DX/AX pair (4 bytes)
                           ; Quotient will be in AX
                           ; Remainder will be in DX
       MOV DX, 0           ; Clear DX
      POP AX               ; Remove month from stack into AX
       MOV CX, 0           ; Initialize the counter
      MOV BX, 10           ; Set up the divisor
 dividey:
       DIV BX              ; Divide (will be word sized)
      PUSH DX              ; Save remainder to stack
      ADD CX, 1            ; Add one to counter
      MOV DX, 0            ; Clear the remainder
      CMP AX, 0            ; Compare quotient to zero
      JNE dividey          ; If quoient is not zero, go to "dividey:"
divdispy:
      POP DX               ; Remove top of stack into DX
     ADD DL, 30h           ; ADD 30h (2) to DL
      MOV AH, 02h          ; 02h to display AH (DL)
      INT 21H              ; Send to DOS
     LOOP divdispy         ; If more to do, divdisp again
                           ; LOOP subtracts 1 from CX. If non-zero, loop.
     MOV al, 0             ; Use 0 as return code
     MOV AH, 4ch           ; Send return code to AH
      INT 21H              ; Send return code to DOS to exit.

Today   ENDP               ; End procedure
END     Today              ; End code. Start using "Today" procedure.
```

MVI A, 80H: Initialize 8255, port A and port B
OUT 83H (CR): in output mode
START: MVI A, 09H
OUT 80H (PA): Send data on PA to glow R1 and R2
MVI A, 24H
OUT 81H (PB): Send data on PB to glow G3 and G4
MVI C, 28H: Load multiplier count (40_{10}) for delay
CALL DELAY: Call delay subroutine
MVI A, 12H
OUT (81H) PA: Send data on Port A to glow Y1 and Y2
OUT (81H) PB: Send data on port B to glow Y3 and Y4
MVI C, 0AH: Load multiplier count (10_{10}) for delay
CALL: DELAY: Call delay subroutine
MVI A, 24H
OUT (80H) PA: Send data on port A to glow G1 and G2
MVI A, 09H
OUT (81H) PB: Send data on port B to glow R3 and R4
MVI C, 28H: Load multiplier count (40_{10}) for delay

CALL DELAY: Call delay subroutine
MVI A, 12H
OUT PA: Send data on port A to glow Y1 and Y2
OUT PB: Send data on port B to glow Y3 and Y4
MVI C, 0AH: Load multiplier count (10ıo) for delay
CALL DELAY: Call delay subroutine
JMP START
Delay Subroutine:
DELAY: LXI D, Count: Load count to give 0.5 sec delay
BACK: DCX D: Decrement counter
MOV A, D
ORA E: Check whether count is 0
JNZ BACK: If not zero, repeat
DCR C: Check if multiplier zero, otherwise repeat
JNZ DELAY
RET: Return to main program

Ram size:

```
        ORG 0000H
      CLR  PSW3
       CLR PSW4
        CPL A
        ADD A, #01H
        MOV A,R3
AGAIN:  SJMP AGAIN
```

RESULT:

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

Counters and Time Delay

EXP.NO: 06 **DATE:**

AIM:

To write an assembly language program in 8086 to Counters and Time Delay

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

PROGRAM:

```
.MODEL SMALL
.DATA

MSGIN DB 'Enter delay duration (0-50): $'
MSG1 DB 'This is Microprocessor!$'
 DELAYTIME DW 0000H

.CODE

MOV DX,@DATA
MOV DS,DX
LEA DX,MSGIN
MOV AH,09H
INT 21H

IN1:
   MOV AH,01H
   INT 21H
   CMP AL,0DH   ;
   JE NXT
   SUB AL,30H
   MOV DL,AL
   MOV AX,BX
   MOV CL,0AH
   MUL CL
   MOV BX,AX
   AND DX,00FFH
   ADD BX,DX
   MOV DELAYTIME,BX
   LOOP IN1
```

```
NXT:    MOV CX,DELAYTIME
        MOV DL,10
        MOV AH,02H
        INT 21H

LEA SI,MSG1

LP:   PUSH DX
      MOV DL,[SI]
      CMP DL,'$'
      JE NXT2
     MOV AH,02H
INT 21H
ADD SI,1
POP DX
MOV DI,DELAYTIME
MOV AH, 0
INT 1Ah
MOV BX, DX

Delay:
MOV AH, 0
INT 1Ah
SUB DX, BX
CMP DI, DX
JA Delay

LOOP LP

  NXT2:  MOV AH,4CH
         INT 21H


END
```

RESULT:

Thus the output for the Counters and Time Delay was executed successfully

TRAFFIC LIGHT CONTROL

EXP.NO: 07 **DATE:**

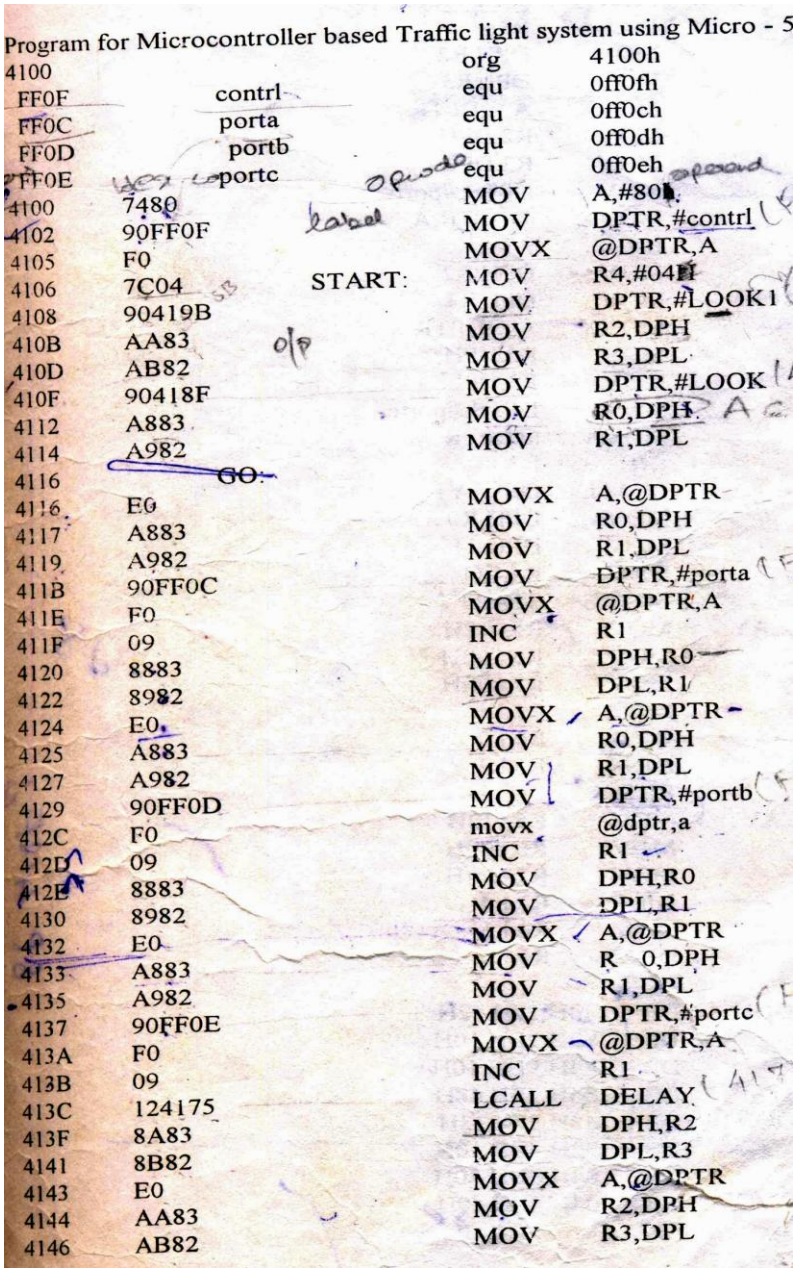
AIM:

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

APPARATUS REQUIRED:

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

PROGRAM:



Program for Microcontroller based Traffic light system using Micro - 5

```
4100                        org     4100h
FF0F        contrl          equ     0ff0fh
FF0C        porta           equ     0ff0ch
FF0D        portb           equ     0ff0dh
FF0E        portc           equ     0ff0eh
4100  7480                  MOV     A,#80H
4102  90FF0F                MOV     DPTR,#contrl
4105  F0                    MOVX    @DPTR,A
4106  7C04        START:    MOV     R4,#04H
4108  90419B                MOV     DPTR,#LOOK1
410B  AA83                  MOV     R2,DPH
410D  AB82                  MOV     R3,DPL
410F  90418F                MOV     DPTR,#LOOK
4112  A883                  MOV     R0,DPH
4114  A982                  MOV     R1,DPL
4116
4116  E0                    MOVX    A,@DPTR
4117  A883                  MOV     R0,DPH
4119  A982                  MOV     R1,DPL
411B  90FF0C                MOV     DPTR,#porta
411E  F0                    MOVX    @DPTR,A
411F  09                    INC     R1
4120  8883                  MOV     DPH,R0
4122  8982                  MOV     DPL,R1
4124  E0                    MOVX    A,@DPTR
4125  A883                  MOV     R0,DPH
4127  A982                  MOV     R1,DPL
4129  90FF0D                MOV     DPTR,#portb
412C  F0                    movx    @dptr,a
412D  09                    INC     R1
412E  8883                  MOV     DPH,R0
4130  8982                  MOV     DPL,R1
4132  E0                    MOVX    A,@DPTR
4133  A883                  MOV     R 0,DPH
4135  A982                  MOV     R1,DPL
4137  90FF0E                MOV     DPTR,#portc
413A  F0                    MOVX    @DPTR,A
413B  09                    INC     R1
413C  124175                LCALL   DELAY
413F  8A83                  MOV     DPH,R2
4141  8B82                  MOV     DPL,R3
4143  E0                    MOVX    A,@DPTR
4144  AA83                  MOV     R2,DPH
4146  AB82                  MOV     R3,DPL
```

```
4148    90FF0C          MOV     DPTR,#porta
414B    F0              MOVX    @DPTR,A
414C    0B              INC     R3
414D    8A83            MOV     DPH,R2
414F    8B82            MOV     DPL,R3
4151    E0              MOVX    A,@DPTR
4152    AA83            MOV     R2,DPH
4154    AB82            MOV     R3,DPL
4156    90FF0D          MOV     DPTR,#portb
4159    F0              MOVX    @DPTR,A
415A    0B              INC     R3
415B    8A83            MOV     DPH,R2
415D    8B82            MOV     DPL,R3
415F    E0              MOVX    A,@DPTR
4160    AA83            MOV     R2,DPH
4162    AB82            MOV     R3,DPL
4164    90FF0E          MOV     DPTR,#portc
4167    F0              MOVX    @DPTR,A
4168    0B              INC     R3
4169    124182          LCALL   DELAY1
416C    8883            MOV     DPH,R0
416E    8982            MOV     DPL,R1
4170    DCA4            DJNZ    R4,GO
4172    124106          LCALL   START
4175    7D12    DELAY:  MOV     R5,#12H
4177    7EFF    L3:     MOV     R6,#0ffH
4179    7FFF    L2:     MOV     R7,#0ffH
417B    DFFE    L1:     DJNZ    R7,L1
417D    DEFA            DJNZ    R6,L2
417F    DDF6            DJNZ    R5,L3
4181    22              RET
4182    7D12    DELAY1: MOV     R5,#12H
4184    7EFF    L6:     MOV     R6,#0ffH
4186    7FFF    L5:     MOV     R7,#0ffH
4188    DFFE    L4:     DJNZ    R7,L4
418A    DEFA            DJNZ    R6,L5
418C    DDF6            DJNZ    R5,L6
418E    22              RET
418F    44 27 12  LOOK: DB      44H,27H,12H
4192    92 2B 10        DB      92H,2BH,10H
4195    84 9D 10        DB      84H,9DH,10H
4198    84 2E 48        DB      84H,2EH,48H
419B    48 27 12  LOOK1:DB      48H,27H,12H
419E    92 4B 10        DB      92H,4BH,10H
41A1    84 9D 20        DB      84H,9DH,20H
41A4    04 2E 49        DB      04H,2EH,49H
                        END
```

STEPPER MOTOR INTERFACING

EXP.NO: 08 **DATE:**

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:

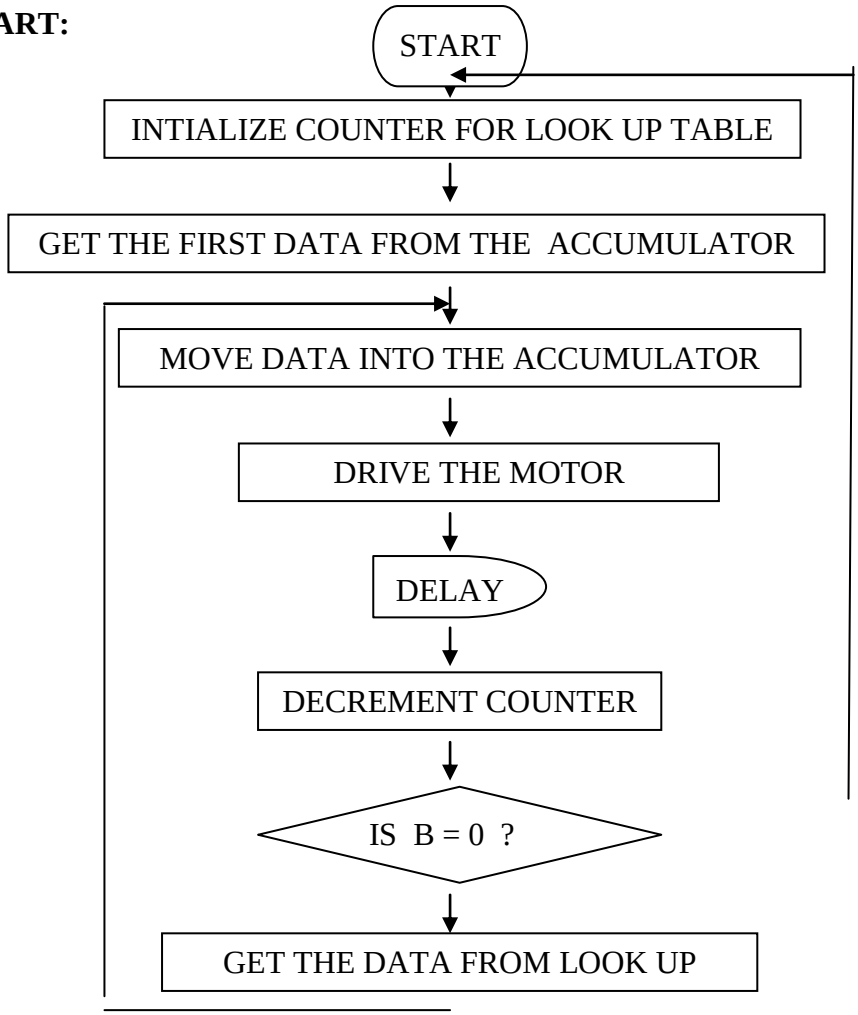
For running stepper motor clockwise and anticlockwise directions

(i) Get the first data from the lookup table.
(ii) Initialize the counter and move data into accumulator.
(iii) Drive the stepper motor circuitry and introduce delay
(iv) Decrement the counter is not zero repeat from step(iii)
(v) 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:



PROGRAM TABLE

ADDRESS	OPCODE	PROGRAM	COMMENTS
		START : MOV DI, 1200H	Initialize memory location to store the array of number
		MOV CX, 0004H	Initialize array size
		LOOP 1 : MOV AL,[DI]	Copy the first data in AL
		OUT 0C0,AL	Send it through port address
		MOV DX, 1010H	Introduce delay
		L1 : DEC DX	
		JNZ L1	
		INC DI	Go to next memory location
		LOOP LOOP1	Loop until all the data's have been sent
		JMP START	Go to start location for continuous rotation
		1200 : 09,05,06,0A	Array of data's

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

DIGITAL CLOCK

EXP.NO: 09 **DATE:**

Aim

To display the digital clock specifically by displaying the hours, minutes and seconds using 8086 kits

Apparatus required

S.No	Item	Specification
1	Microprocessor kit	**8086**
2	Power Supply	5V

Preliminary Settings

Org 1000h

Store time value in memory location 1500- Seconds
1501- Minutes
1502- Hours

Digital clock program

Memory location	Opcode	Label	Mnemonics
1000		START	
1000	E87500		CALL CONVERT
1003	E86200		CALL DISPLAY
1006	C6C0B0	DELAY	MOV AL,0B0H
1009	E616		OUT 16H,AL
100B	C6C107		MOV CL,07H
100E	C6C088	S2	MOV AL,88H
1011	E614		OUT 14H,AL
1013	C6C080		MOV AL,80H
1016	E614		OUT 14H,AL
1018	C6C080	S1	MOV AL,80H
101B	E616		OUT 16H,AL
101D	90		NOP
101E	90		NOP
101F	90		NOP
1020	90		NOP
1021	E414		IN AL,14H
1023	88C2		MOV DL,AL
1025	E414		IN AL,14H
1027	08D0		OR AL,DL
1029	75ED		JNZ S1
102B	FEC9		DEC CL

102D	75DF		JNZ S2
102F	C7C60015		MOV SI,1500H
1033	8A04		MOV AL,[SI]
1035	FEC0		INC AL
1037	8804		MOV [SI],AL
1039	80F83C		CMP AL,3CH
103C	75C2		JNZ START
103E	C6C000		MOV AL,00H
1041	8804		MOV [SI],AL
1043	46		INC SI
1044	8A04		MOV AL,[SI]
1046	FEC0		INC AL
1048	8804		MOV [SI],AL
104A	80F83C		CMP AL,3CH
104D	75B1		JNZ START
104F	C6C0000		MOV AL,0
1052	8804		MOV [SI],AL
1054	46		INC SI
1055	8A04		MOV AL,[SI]
1057	FEC0		INC AL
1059	8804		MOV [SI],AL
105B	80F818		CMP AL,18H
105E	75A0		JNZ START
1060	C6C000		MOV AL,0
1063	8804		MOV [SI],AL
1065	E998FF		JMP START
1068	C6C406	DISPLAY	MOV AH,06H
106B	C7C20016		MOV DX,1600H
106F	C6C501		MOV CH,01H
1072	C6C100		MOV CL,0H
1075	CD05		INT 5
1077	C3		RET
1078	C7C60015	CONVERT	MOV SI,1500H
107C	C7C30816		MOV BX,1608H
1080	C6C024		MOV AL,24H
1083	8807		MOV [BX],AL
SECONDS			
1085	8A04		MOV AL,[SI]
1087	C6C400		MOV AH,0
108A	C6C60A		MOV DH,0AH
108D	F6F6		DIV DH
108F	80C430		ADD AH,30H
1092	4B		DEC BX
1093	8827		MOV [BX],AH
1095	4B		DEC BX
1096	80C030		ADD AL,30H

1099	8807		MOV [BX],AL
109B	4B		DEC BX
109C	C6C03A		MOV AL,3AH
109F	8807		MOV [BX],AL
10A1	4B		DEC BX
MINUTES			
10A2	46		INC SI
10A3	8A04		MOV AL,[SI]
10A5	C6C400		MOV AH,0
10A8	C6C60A		MOV DH,0AH
10AB	F6F6		DIV DH
10AD	80C430		ADD AH,30H
10B0	8827		MOV [BX],AH
10B2	4B		DEC BX
10B3	80C030		ADD AL,30H
10B6	8807		MOV [BX],AL
10B8	4B		DEC BX
10B9	C6C03A		MOV AL,3AH
10BC	8807		MOV [BX],AL
10BE	4B		DEC BX
HOURS			
10BF	46		INC SI
10C0	8A04		MOV AL,[SI]
10C2	C6C400		MOV AH,0
10C5	C6C60A		MOV DH,0AH
10C8	F6F6		DIV DH
10CA	80C430		ADD AH,30H
10CD	8827		MOV [BX],AH
10CF	4B		DEC BX
10D0	80C030		ADD AL,30H
10D	8807		MOV [BX],AL
10D	C3		RET
10D	C3		GETC
10D	E402		IN AL,02H
10D	80E0FF		AND AL,0FFH
10D	80F8F0		CMP AL,0F0H
10E	75F6		JNE GETC

Result

Thus the digital clock program has been written and executed using 8086 microprocessor kit and the output of digital clock was displayed as [hours: minutes: seconds] successfully.

INTERFACING PRGRAMMABLE KEYBOARD AND DISPLAY CONTROLLER- 8279

EXP.NO:10 **DATE:**

AIM :

To display the rolling message " HELP US " in the display.

APPARATUS REQUIRED:

8086 Microprocessor kit, Power supply, Interfacing board.

ALGORITHM :

Display of rolling message " HELP US "

1. Initialize the counter
2. Set 8279 for 8 digit character display, right entry
3. Set 8279 for clearing the display
4. Write the command to display
5. Load the character into accumulator and display it
6. Introduce the delay
7. Repeat from step 1.

1. <u>Display Mode Setup:</u> Control word-10 H

0	**0**	**0**	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**D**	**D**	**K**	**K**	**K**

DD

00- 8Bit character display left entry
01- 16Bit character display left entry
10- 8Bit character display right entry
11- 16Bit character display right entry

KKK- Key Board Mode

000-2Key lockout.

<u>2.Clear Display:</u> Control word-DC H

1	**1**	**0**	**1**	**1**	**1**	**0**	**0**
1	**1**	**0**	**CD**	**CD**	**CD**	**CF**	**CA**

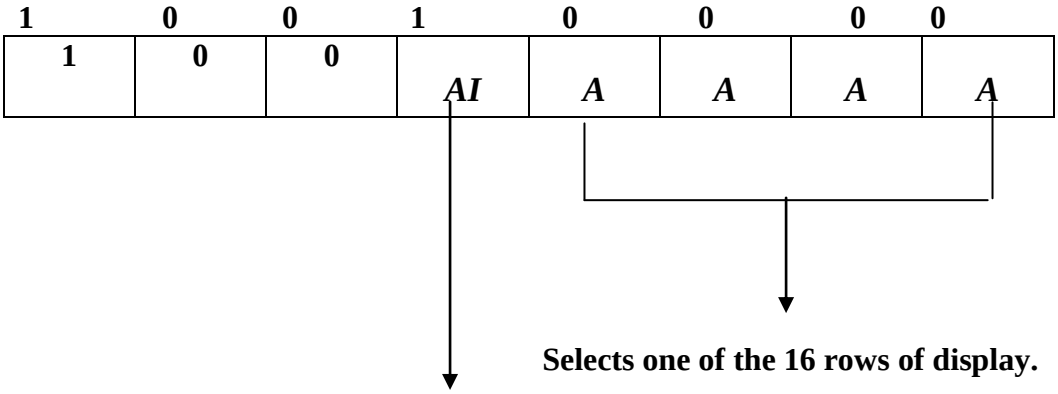
11 **A0-3; B0-3 =FF**

1-Enables Clear display
0-Contents of RAM will be displayed

1-FIFO Status is cleared

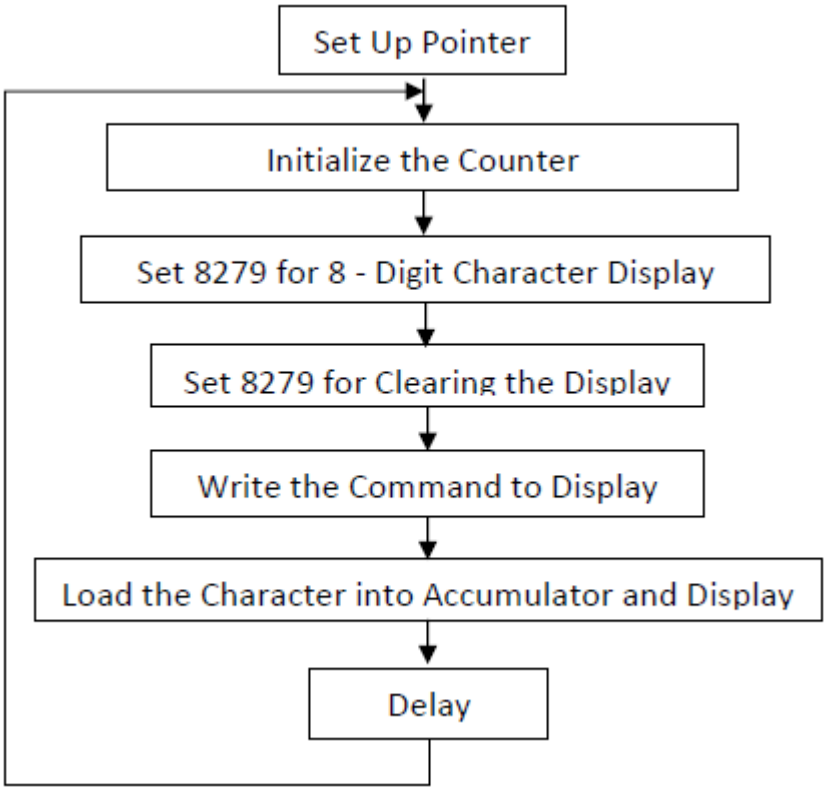
1-Clear all bits
(Combined effect of CD)

3. Write Display: Control word-90H



Auto increment = 1, the row address selected will be incremented after each of read and write operation of the display RAM.

FLOWCHART



SEGMENT DEFINITION

DATA BUS	D7	D6	D5	D4	D3	D2	D1	D0
SEGMENTS	d	c	b	a	dp	g	f	e

PROGRAM TABLE

PROGRAM	COMMENTS
START : MOV SI,1200H	Initialize array
MOV CX,000FH	Initialize array size
MOV AL,10	Store the control word for display mode
OUT C2,AL	Send through output port
MOV AL,CC	Store the control word to clear display
OUT C2,AL	Send through output port
MOV AL,90	Store the control word to write display
OUT C2,AL	Send through output port
L1 : MOV AL,[SI]	Get the first data
OUT C0,AL	Send through output port
CALL DELAY	Give delay
INC SI	Go & get next data
LOOP L1	Loop until all the data's have been taken
JMP START	Go to starting location
DELAY : MOV DX,0A0FFH	Store 16bit count value
LOOP1 : DEC DX	Decrement count value
JNZ LOOP1	Loop until count values becomes zero
RET	Return to main program

LOOK-UP TABLE:

1200	98	68	7C	C8
1204	FF	1C	29	FF

RESULT:

MEMORY LOCATION	7-SEGMENT LED FORMAT								HEX DATA
	d	c	b	a	dp	e	g	f	
1200H	1	0	0	1	1	0	0	0	98
1201H	0	1	1	0	1	0	0	0	68
1202H	0	1	1	1	1	1	0	0	7C
1203H	1	1	0	0	1	0	0	0	C8
1204H	1	1	1	1	1	1	1	1	FF
1205H	0	0	0	0	1	1	0	0	1C
1206H	0	0	1	0	1	0	0	1	29
1207H	1	1	1	1	1	1	1	1	FF

Thus the rolling message "HELP US" is displayed using 8279 interface kit.

PRINTER STATUS

EXP.NO:11 **DATE:**

AIM:

To display the Printer Status in the display

APPARATUS REQUIRED:

8086 Microprocessor kit, Power supply, interfacing board.

PROGRAM:

```
XOR AX, AX
XOR BX, BX

   ;this divides my 3digit number by 100 giving me my, hundredth digit
MOV AX, RES
MOV BX, 100
DIV BX

   ;prints the hundredth digit
ADD AL, '0'
MOV DL, AL
PUSH AX ; save AX on the stack
MOV AH, 02h
INT 21h
POP AX ; restore ax

   ;divides the remainder by 10 giving me my tens digit
MOV BX, 10
DIV BX

   ;prints my tens digit
ADD AL, '0'
MOV DL, AL
PUSH AX ; save AX on the stack
MOV AH, 02h
INT 21h
POP AX ; restore ax

   ;print my last remainder which is my ones
ADD AH, '0'
MOV DL, AH
MOV AH, 02h
INT 21h
```

RESULT:

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

SERIAL INTERFACE AND PARALLEL INTERFACE

EXP.NO: 12 **DATE:**

Aim

To connect two 8086 microprocessor kits and to serially communicate with each other by considering transmitter and receiver kits.

Apparatus required

S.No	Item	Specification	Quantity
1	Microprocessor kit	8086	2
2	Power Supply	+5 V, dc, +12 V dc	1
3	9 - 9 Cable	RS 232 C	1

Procedure

1. Take two no of 8086 microprocessor kits.
2. Enter the transmitter program in transmitter kit.
3. Enter the receiver program in receiver kit.
4. Interface the two kits with 9-9 serial cable in the serial port of the microprocessor kits. (LCD kit means PC-PC cable. LED kit means kit-kit cable)
5. Enter the data in transmitter kit use the memory location 1500.
6. Execute the receiver kit.
7. Execute the transmitter kit.
8. Result will be available in receiver kit memory location 1500.

Transmitter Program

Memory address	Opcode	Label	Mnemonics
1000	C7C60015		MOV SI,1500H
1004	C6C036		MOV AL,36H
1007	E616		OUT 16H,AL
1009	C6C040		MOV AL,40H
100C	E610		OUT 10H,AL
100E	C6C001		MOV AL,01H
1011	E610		OUT 10H,AL
1013	C6C105	RELOAD	MOV CL,05H
1016	E40A	CHECK	IN AL,0AH
1018	80E004		AND AL,04H
101B	74F9		JZ CHECK
101D	8A04		MOV AL,[SI]
101F	E608		OUT 08H,AL
1021	46		INC SI
1022	80F83F		CMP AL,3FH
1025	75EC		JNZ RELOAD
1027	FEC9		DEC CL
1029	75EB		JNZ CHECK
102B	CD02		INT 02

Receiver Program

Memory address	Opcode	Label	Mnemonics
1000	C7C60015		MOV SI,1500H
1004	C6C036		MOV AL,36H
1007	E616		OUT 16H,AL
1009	C6C040		MOV AL,40H
100C	E610		OUT 10H,AL
100E	C6C001		MOV AL,01H
1011	E610		OUT 10H,AL
1013	C6C105	RELOAD	MOV CL,05H
1016	E40A	CHECK	IN AL,0AH
1018	80E002		AND AL,02H
101B	74F9		JZ CHECK
101D	E408		IN AL,08H
101F	8804		MOV [SI],AL
1021	46		INC SI
1022	80F83F		CMP AL,3FH
1025	75EC		JNZ RELOAD
1027	FEC9		DEC CL
1029	75EB		JNZ CHECK
102B	CD02		INT 02
102D	CD02		INT 02
			CODE END

Result

Thus the serial communication between two 8086 microprocessor kits has been established and the data is transmitted in one kit and received in the other kit successfully

PARALLEL COMMUNICATION BETWEEN TWO 8086 MICROPROCESSORS KITS

Aim

To connect two 8086 microprocessor kits and to establish parallel communication with each other by considering transmitter and receiver kits.

Apparatus required

S.No	Item	Specification	Quantity
1	Microprocessor kit	8086	2
2	Power Supply	+5 V, dc, +12 V dc	1
3	26 Core cable	---	1

Procedure

1. Take two 8086 microprocessor kits.
2. Enter the transmitter program in transmitter kit.
3. Enter the receiver program in receiver kit.
4. Interface the two kits with 26-core cable on PPI-1.
5. Execute the receiver kit.
6. Execute the transmitter kit.
7. Go and see the memory location 1200 in receiver to get same eight data.
8. Data is available in transmitter kit in the memory location.
9. Change the data & execute the following procedure & get the result in receiver kit.

Transmitter program

Memory address	Opcode	Label	Mnemonics
1000	C6C082		MOV AL,82H
1003	E626		OUT 26H,AL
1005	C6C03F		MOV AL,3FH
1008	E620		OUT 20H,AL
100A	E422	LOOP	IN AL,22H
100C	80E83F		SUB AL,3FH
100F	75F9		JNZ LOOP
1011	C6C024		MOV AL,24H
1014	E620		OUT 20H,AL
1016	E80200		CALL DELAY
1019	CD02		INT 02
101B	C6C35F	DELAY	MOV BL,05H
101E	C6C2FF	LION	MOV DL,0FFH
1021	FECA	L2	DEC DL
1023	75FC		JNZ L2
1025	FECB		DEC BL
1027	75F5		JNZ LION
1029	C3		RET

Receiver Program

Memory address	Opcode	Label	Mnemonics
1000	C6C090		MOV AL,90H
1003	E626		OUT 26H,AL
1005	E420	CHECK	IN AL,20H
1007	80E83F		SUB AL,3FH
100A	75F9		JNZ CHECK
100C	C6C03F		MOV AL,3FH
100F	E622		OUT 22H,AL
1011	C6C108		MOV CL,08H
1014	E81200		CALL DELAY
1017	C7C60012		MOV SI,1200H
101B	E420	L1	IN AL,20H
101D	8804		MOV [SI],AL
101F	E80700		CALL DELAY
1022	46		INC SI
1023	FEC9		DEC CL
1025	75F4		JNZ L1
1027	CD02		INT 02
1029	C6C305	DELAY	MOV BL,05H
102C	C6C2FF	LION	MOV DL,0FFH
102F	FECA	L2	DEC DL
1031	75FC		JNZ L2
1033	FFCB		DEC BL
1035	75F5		JNZ LION
1037	C3		RET
1038	C3		RET

Result

Thus the serial communication between two 8086 microprocessor kits has been established and the data is transmitted in one kit and received in the other kit successfully.

A/D AND D/A INTERFACE AND WAVEFORM GENERATION

EXPT NO:13 **DATE:**

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**	8086	1
2.	**Power Supply**	+5 V dc,+12 V dc	1
3.	**ADC Interface board**	-	1

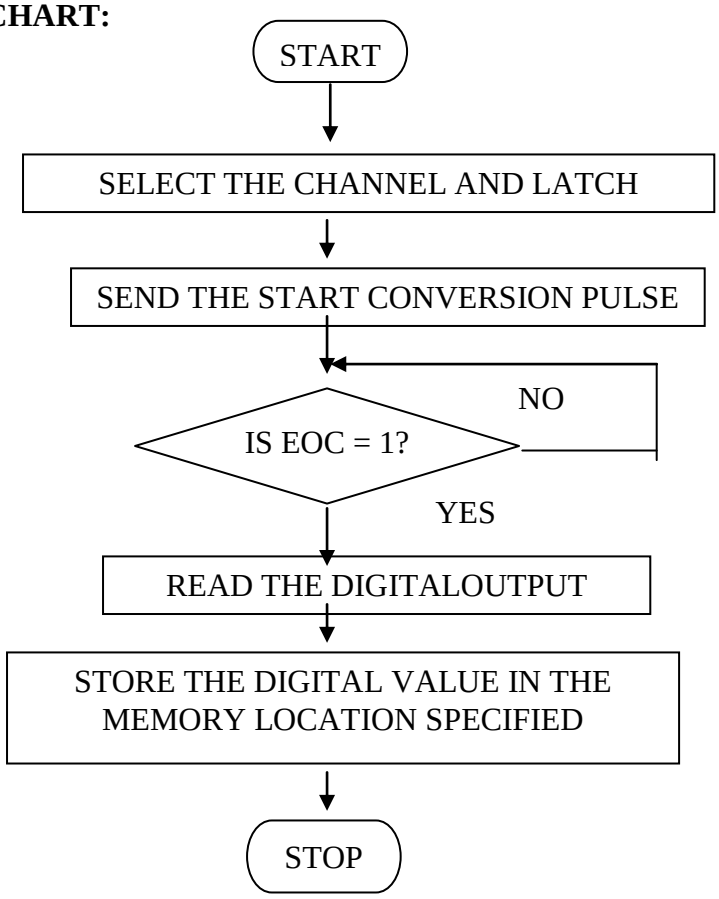
THEORY:

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. 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:

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

FLOW CHART:



PROGRAM TABLE

PROGRAM	COMMENTS
MOV AL,00	Load accumulator with value for ALE high
OUT 0C8H,AL	Send through output port
MOV AL,08	Load accumulator with value for ALE low
OUT 0C8H,AL	Send through output port
MOV AL,01	Store the value to make SOC high in the accumulator
OUT 0D0H,AL	Send through output port
MOV AL,00	Introduce delay
MOV AL,00	
MOV AL,00	
MOV AL,00	Store the value to make SOC low the accumulator
OUT 0D0H,AL	Send through output port
L1 : IN AL, 0D8H	Read the EOC signal from port & check for end of conversion
AND AL,01	
CMP AL,01	
JNZ L1	If the conversion is not yet completed, read EOC signal from port again
IN AL,0C0H	Read data from port
MOV BX,1100	Initialize the memory location to store data
MOV [BX],AL	Store the data
HLT	Stop

RESULT:

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

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

INTERFACING DIGITAL – TO – ANALOG CONVERTER

AIM :

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

APPARATUS REQUIRED:

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

PROBLEM STATEMENT:

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

THEORY:

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

ALGORITHM:

Measurement of analog voltage:

(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:

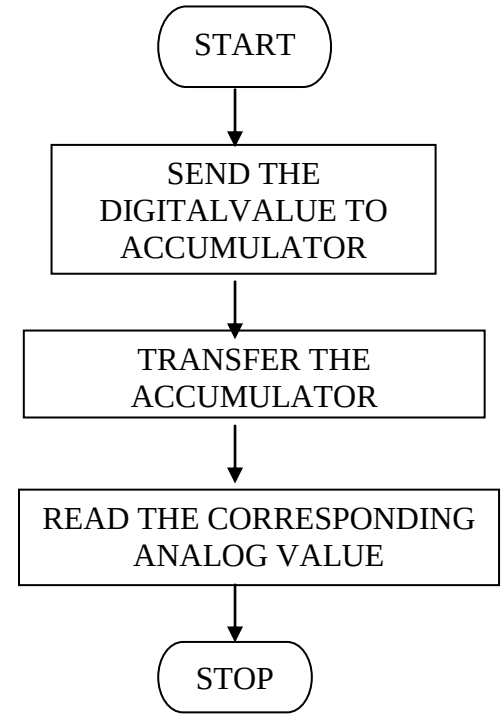
(i) Load low value (00) to accumulator.
(ii) Send this value to DAC.
(iii) Increment the accumulator.
(iv) Repeat step (ii) and (iii) until accumulator value reaches FF.
(v) Repeat the above procedure from step 1.

Triangular waveform:

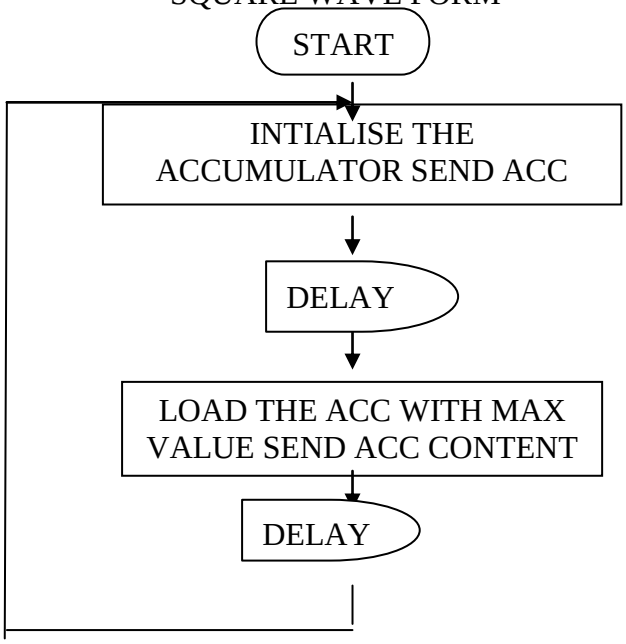
(i) Load the low value (00) in accumulator.
(ii) Send this accumulator content to DAC.
(iii) Increment the accumulator.
(iv) Repeat step 2 and 3 until the accumulator reaches FF, decrement the accumulator and send the accumulator contents to DAC.
(v) Decrementing and sending the accumulator contents to DAC.
(vi) The above procedure is repeated from step (i)

FLOWCHART:

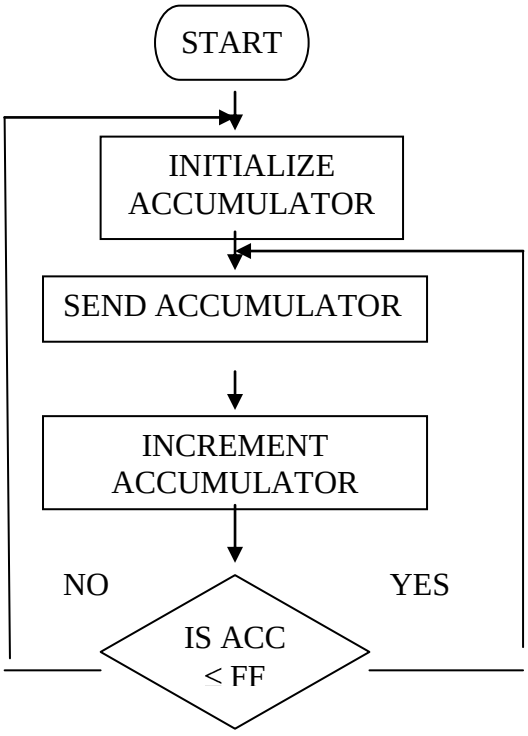
MEASUREMENT OF ANALOG VOLTAGE



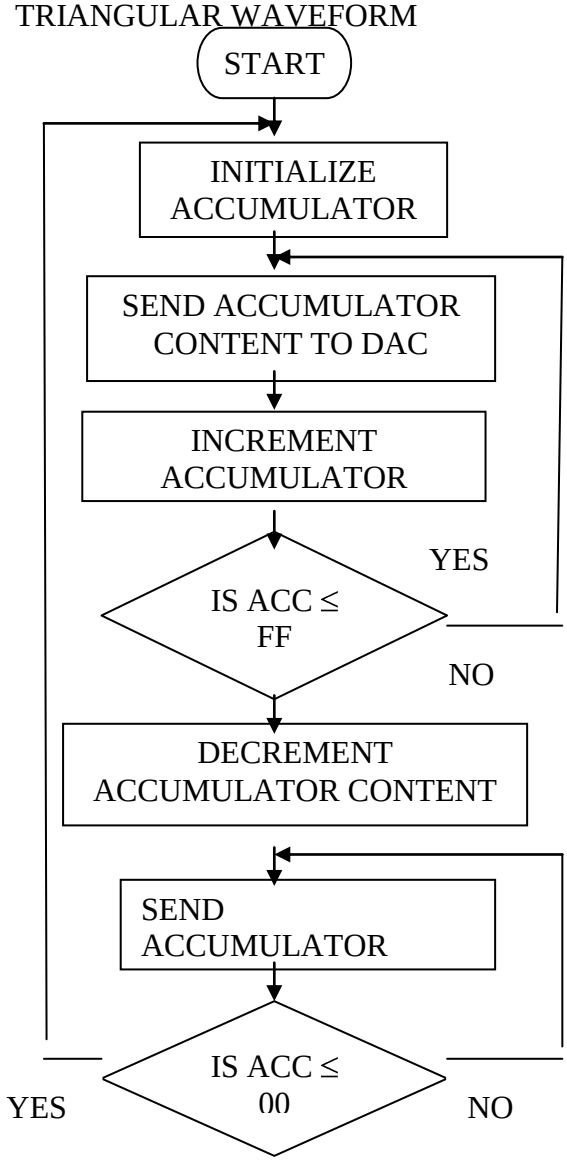
SQUARE WAVE FORM



SAWTOOTH WAVEFORM



TRIANGULAR WAVEFORM



MEASUREMENT OF ANALOG VOLTAGE:

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

DIGITAL DATA	ANALOG VOLTAGE

PROGRAM TABLE: Square Wave

PROGRAM	COMMENTS
L2 : MOV AL,00H	Load 00 in accumulator
OUT C0,AL	Send through output port
CALL L1	Give a delay
MOV AL,FFH	Load FF in accumulator
OUT C0,AL	Send through output port
CALL L1	Give a delay
JMP L2	Go to starting location
L1 : MOV CX,05FFH	Load count value in CX register
L3 : LOOP L3	Decrement until it reaches zero
RET	Return to main program

PROGRAM TABLE: Saw tooth Wave

PROGRAM	COMMENTS
L2 : MOV AL,00H	Load 00 in accumulator
L1 : OUT C0,AL	Send through output port
INC AL	Increment contents of accumulator
JNZ L1	Send through output port until it reaches FF
JMP L2	Go to starting location

PROGRAM TABLE: Triangular Wave

PROGRAM	COMMENTS
L3 : MOV AL,00H	**Load 00 in accumulator**
L1 : OUT C0,AL	**Send through output port**
INC AL	**Increment contents of accumulator**
JNZ L1	**Send through output port until it reaches FF**
MOV AL,0FFH	**Load FF in accumulator**
L2 : OUT C0,AL	**Send through output port**
DEC AL	**Decrement contents of accumulator**
JNZ L2	**Send through output port until it reaches 00**
JMP L3	**Go to starting location**

RESULT:WAVEFORM GENERATION:

WAVEFORMS	AMPLITUDE	TIMEPERIOD
Square Waveform		
Saw-tooth waveform		
Triangular waveform		

MODEL GRAPH:

Square Waveform **Saw-tooth waveform**

Triangular waveform

Thus the DAC was interfaced with 8085 and different waveforms have been generated.

BASIC ARITHMETIC AND LOGICAL OPERATIONS

EXPT NO:14 **DATE:**

8 BIT ADDITION

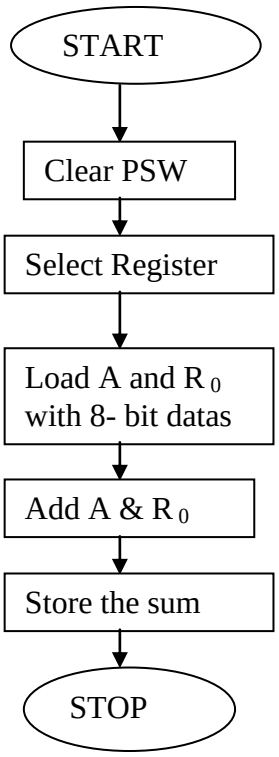
AIM:

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

ALGORITHM:

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

FLOW CHART:



8 Bit Addition (Immediate Addressing)

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		CLR	C	C3	Clear CY Flag
4101		MOV	A,# data1	74,data1	Get the data1 in Accumulator
4103		ADDC	A, # data 2	24,data2	Add the data1 with data2
4105		MOV	DPTR, # 4500H	90,45,00	Initialize the memory location
4108		MOVX	@ DPTR, A	F0	Store the result in memory location
4109	L1	SJMP	L1	80,FE	Stop the program

RESULT:

OUTPUT	
MEMORY LOCATION	DATA
4500	

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

8 BIT SUBTRACTION

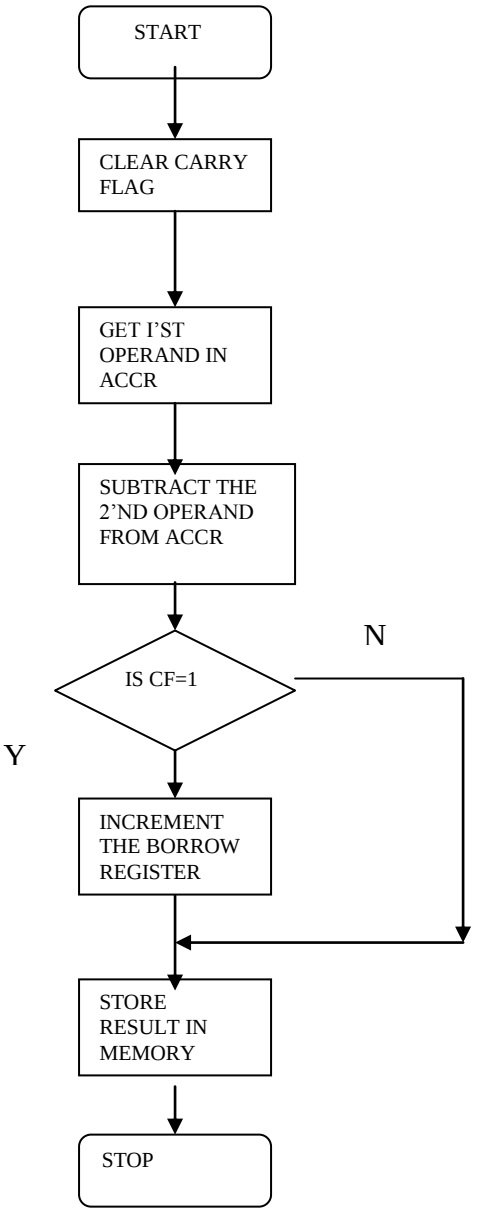
AIM:

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

ALGORITHM:

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

FLOWCHART:



8 Bit Subtraction (Immediate Addressing)

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		CLR	C	C3	Clear CY flag
4101		MOV	A, # data1	74, data1	Store data1 in accumulator
4103		SUBB	A, # data2	94,data2	Subtract data2 from data1
4105		MOV	DPTR, # 4500	90,45,00	Initialize memory location
4108		MOVX	@ DPTR, A	F0	Store the difference in memory location
4109	L1	SJMP	L1	80,FE	Stop

RESULT:

OUTPUT	
MEMORY LOCATION	DATA
4500	

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

8 BIT MULTIPLICATION

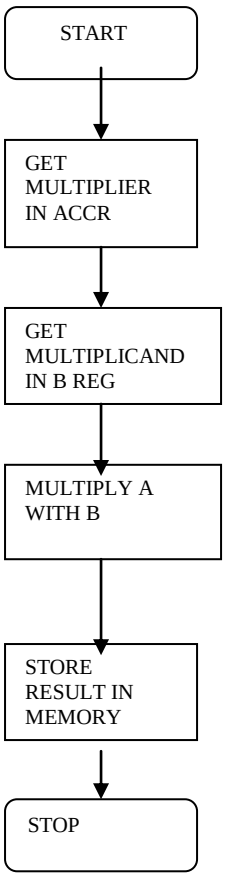
AIM:

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

ALGORITHM:

a. Get the multiplier in the accumulator.
b. Get the multiplicand in the B register.
c. Multiply A with B.
d. Store the product in memory.

FLOWCHART:



8 Bit Multiplication

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		MOV	A ,#data1	74, data1	Store data1 in accumulator
4102		MOV	B, #data2	75,data2	Store data2 in B reg
4104		MUL	A,B	F5,F0	Multiply both
4106		MOV	DPTR, # 4500H	90,45,00	Initialize memory location
4109		MOVX	@ DPTR, A	F0	Store lower order result
401A		INC	DPTR	A3	Go to next memory location
410B		MOV	A,B	E5,F0	Store higher order result
410D		MOV	@ DPTR, A	F0	
410E	STOP	SJMP	STOP	80,FE	Stop

RESULT:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
4500		4502	
4501		4503	

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

8 BIT DIVISION

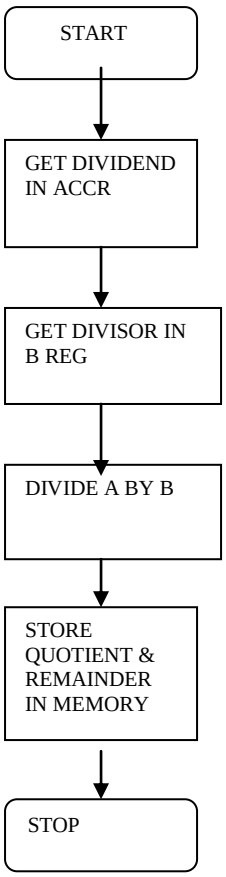
AIM:

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

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.

FLOWCHART:



8 Bit Division

ADDRESS	LABEL	MNEMONIC	OPERAND	HEX CODE	COMMENTS
4100		MOV	A, # data1	74,data1	Store data1 in accumulator
4102		MOV	B, # data2	75,data2	Store data2 in B reg
4104		DIV	A,B	84	Divide
4015		MOV	DPTR, # 4500H	90,45,00	Initialize memory location
4018		MOVX	@ DPTR, A	F0	Store remainder
4109		INC	DPTR	A3	Go to next memory location
410A		MOV	A,B	E5,F0	Store quotient
410C		MOV	@ DPTR, A	F0	
410D	STOP	SJMP	STOP	80,FE	Stop

RESULT:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
4500 (dividend)		4502 (remainder)	
4501 (divisor)		4503 (quotient)	

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

LOGICAL AND BIT MANIPULATION

AIM:

To write an ALP to perform logical and bit manipulation operations using 8051 microcontroller.

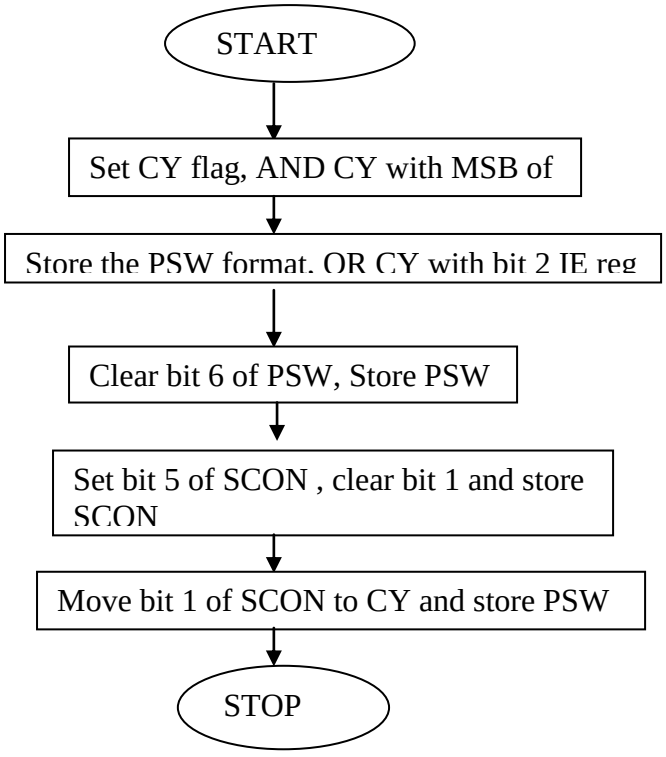
APPARATUS REQUIRED:

8051 microcontroller kit

ALGORITHM:

1. Initialize content of accumulator as FFH
2. Set carry flag (cy = 1).
3. AND bit 7 of accumulator with cy and store PSW format.
4. OR bit 6 of PSW and store the PSW format.
5. Set bit 5 of SCON.
6. Clear bit 1 of SCON.
7. Move SCON.1 to carry register.
8. Stop the execution of program.

FLOWCHART:



PROGRAM TABLE

ADDRESS	HEX CODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100	90,45,00		MOV	DPTR,#4500	Initialize memory location
4103	74,FF		MOV	A,#FF	Get the data in accumulator
4105	D3		SETB	C	Set CY bit
4016	82,EF		ANL	C, ACC.7	Perform AND with 7th bit of accumulator
4018	E5,D0		MOV	A,DOH	Store the result
410A	F0		MOVX	@DPTR,A	
410B	A3		INC	DPTR	Go to next location
410C	72,AA		ORL	C, IE.2	OR CY bit with 2nd bit if IE reg
410E	C2,D6		CLR	PSW.6	Clear 6th bit of PSW
4110	E5,D0		MOV	A,DOH	Store the result
4112	F0		MOVX	@DPTR,A	
4113	A3		INC	DPTR	Go to next location
4114	D2,90		SETB	SCON.5	Set 5th of SCON reg
4116	C2,99		CLR	SCON.1	Clear 1st bit of SCON reg
4118	E5,98		MOV	A,98H	Store the result
411A	F0		MOVX	@DPTR,A	
411B	A3		INC	DPTR	Go to next location
411C	A2,99		MOV	C,SCON.1	Copy 1st bit of SCON reg to CY flag
411E	E5,D0		MOV	A,DOH	Store the result
4120	F0		MOVX	@DPTR,A	
4122	80,FE	L2	SJMP	L2	Stop

RESULT:

MEMORY LOCATION	SPECIAL FUNCTION REGISTER FORMAT								BEFORE EXECUTION	AFTER EXECUTION
4500H (PSW)	**CY**	**AC**	**FO**	**RS1**	**RS0**	**OV**	**-**	**P**	**00H**	**88H**
4501H (PSW)	**CY**	**AC**	**FO**	**RS1**	**RS0**	**OV**	**-**	**P**	**40H**	**88H**
4502H (SCON)	**SM0**	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**	**0FH**	**20H**
4503H (PSW)	**CY**	**AC**	**FO**	**RS1**	**RS0**	**OV**	**-**	**P**	**FFH**	**09H**

Thus the bit manipulation operation is done in 8051 microcontroller.

Square and Cube program, Find 2's complement of a number

EXPT NO: 15 **DATE:**

AIM:

To convert Square and Cube program, Find 2's complement of a number using 8051 micro controller

RESOURCES REQUIERED:

- 8051 microcontroller kit
- Keyboard
- Power supply

PROGRAM:

```
org 0000h; sets the program counter to 0000h
 mov a,#n;assign value 'n' in decimal to A which is converted to it's
equivalent hexadecimal value
 mov b,#n;assign value 'n' in decimal to B  which is converted to it's
equivalent hexadecimal value
 mov r0,#n;assign value 'n' in decimal to R0 which is converted to it's
equivalent hexadecimal value
  mul ab;  multiplying 'A' with 'B'
  mov 40h,a;  lower byte is stored in address 40h
  mov 41h,b;  higher byte is stored in address 41h
  mov r1,a;  move value of 'A' to R1
  mov a,b;  move value of 'B' to 'A'
  mov b,r0;  move value of R0 to b
  mul ab;  multiply 'A' and 'B'
  mov b,a;  lower byte obtained is moved from 'A' to 'B'
  mov r2,b;  move value of 'B' to R2
  mov a,r1;  move value of R1 to 'A'
  mov b,r0;  move value of R0 to 'B'
  mul ab;  multiplying 'A' and 'B'
  mov 50h,a;  Lower byte obtained is stored in address 50h
```

```
        mov r3,b;  higher byte obtained is stored in R3
        mov a,r2;  move value from R2 to 'A'
        add a,r3;  add value of 'A' with R3 and store the value in 'A'
        mov b,a;  move value from 'A' to 'B'
        mov 51h,b;  store value obtained in address 51h
       end
```

```
SQUARE  PGM USING 8051
ORG 00h
02        LJMP MAIN
03 DELAY:
04 ;MOV R0,#2
05 MOV TMOD, #01H
06 MOV TH0, #HIGH (-50000)
07 MOV TL0, #LOW (-50000)
08 SETB TR0
09 JNB TF0,
10 CLR TF0
12 ;DJNZ R0,DELAY
13 RET
14 MAIN:
15 MOV DPTR,#300H
16 MOV A,#0FFH
17 MOV P1,A
18 BACK:
19 LCALL DELAY
20 MOV A,P1
21 MOVC A,@A+DPTR
22 ;MOV P2,#00H
23 ;LCALL DELAY
24 MOV P2,A
25 SJMP BACK
26 ORG 300H
27 XSQR_TABLE:
28 DB 0,1,4,9,16,25,36,49,64,81
29 END
```

Thus the Square and Cube program, Find 2’s complement of a number is done in 8051 microcontroller

Unpacked BCD to ASCII

EXPT NO:16 **DATE:**

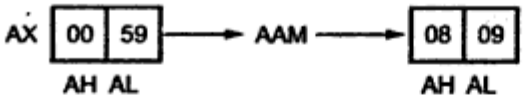
AIM:

To convert BCD number into ASCII by using 8051 micro controller

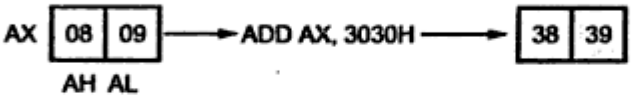
RESOURCES REQUIERED:

- 8051 microcontroller kit
- Keyboard
- Power supply

Algorithm :



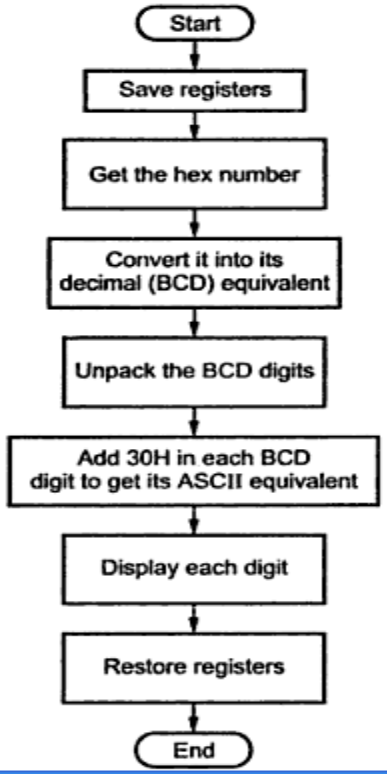
Note : 59H → 89 Decimal



Note : 38H and 39H are the ASCII equivalents of 8 and 9 respectively

1. Save contents of all registers which are used in the routine.
2. Get the data in AL register and make AH equal to 00.
3. Use AAM instruction to convert number in its decimal equivalent in the unpacked format.
4. Add 30H in each digit to get its ASCII equivalent.
5. Display digit one by one using function 2 of INT 21H.
6. Restore contents of registers.

FLOWCHART:



Routine : Convert Binary to ASCII for number less than 100

Passing Parameter : Hex number in AL register.

```
; Routine to convert binary number into its
; Decimal and then ASCII equivalent, and then display the number

BTA PROC NEAR

     PUSH DX            ; Save registers
     PUSH BX
     PUSH AX

     MOV AH, 00H        ; Clear AH
     AAM                ; Convert to BCD
     ADD AX, 3030H      ; Convert to ASCII
     MOV BX,AX          ; Save result
     MOV DL, BH         ; Load first digit (MSD)
     MOV AH, 02         ; Load function number
     INT 21H            ; Display first digit (MSD)

MOV DL, BL              ; Load second digit (LSD)
INT 21H                 ; Display second digit (LSD)

POP AX                  ; Restore registers
POP BX
POP DX
RET
ENDP
```

RESULT:

The given number is converted into ASCII using 8051 microcontroller kit.