**INDEX**

**List of Experiments**

**A. Assembly Language Programs(using MASM)**

1. Programs involving Data Transfer Instructions

i) Block move with and without overlapping.

ii) Block interchange.

2. Programs involving Arithmetic Operation.

i) 16 Bit Addition and Subtraction

ii) Addition of N-bit multi precision numbers (N ≥ 32bits)

iii) Multiplication of two unsigned 32- bit numbers

3. Programs involving Bit Manipulation Instructions.

i) 2 out of 5 Codes

ii) Find the logical 1’s and 0’s in the given data

4. To Find LCM, HCF and Factorial

i) Program to find LCM of a given number

ii) Program to find HCF of a given number

iii) Program to find factorial of a given number

5. Code Conversion

i) BCD to Hexadecimal

ii) Hexadecimal to BCD

6. Programs Involving Branch /Loop Instruction

i) Program to sort the numbers in ascending order (bubble sorting)

ii) Program to find the smallest and largest 16-bit signed number in an array

7. Programs Involving String Manipulation

i) Program to reverse a string

**B. Interface Experiments**

1. Delay calculation and generation of a square wave, triangular waveform using DAC. Display the waveform on a CRO.

2. Interfacing the stepper motor.

**Course Outcomes:**

**1**. Write, compile and debug assembly language program using arithmetic instructions. (POs – 1,

2, 3, 5, PSO – 2)

**2**. Formulate assembly language programs for logical operations and code conversion. (POs – 1,

2, 3, 5, PSO – 2)

**3.** Develop programs using string and loop instructions. (POs – 1, 2, 4, 5, PSO – 2)

**4**. Write assembly language programs to interface stepper motor and DAC to 8086

microprocessor (POs – 1, 2, 4, 5, PSO – 2)

**5**. Apply the concept of parallel computing to perform different operations. (POs – 1, 2, 4, 5,

PSO – 2)

**INTRODUCTION TO MASM**

The Microsoft macro assembler is an x86 high level assembler for DOS and Microsoft windows. It supports wide varieties of macro facilities and structured programming idioms including high level functions for looping and procedures

A program called assembler used to convert the mnemonics of instructions along with the data into the equivalent object code modules, these object code may further converted into executable code using linked and loader programs. This type of program is called as ASSEMBLY LANGUAGE PROGRAMMING. The assembler converts and Assembly language source file to machine code the binary equivalent of the assembly language program. In this respect, the assembler reads an ASCII source file from the disk and program as output.

The Microsoft Macro Assembler (MASM)

The Microsoft Macro Assembler (MASM) is an x86 assembler that uses the Intel syntax for MS-DOS and Microsoft Windows. Beginning with MASM 8.0 there are two versions of the assembler - one for 16-bit and 32-bit assembly sources, and another (ML64) for 64-bit sources only.

Once you are sure that MASM is installed correctly, then you can follow these steps.

1. ***Go to Start, Run, cmd (***or Command Window).
2. ***C :\Document and setting \admin> cd\*** [enter]

If required changes the default drive from C to D.

Once you have located the directory you want, you may move from directory to directory using the CD command (change directory)

1. ***C>D:*** {enter}

Moves you up one level in the path.

1. ***C>D*** [enter]

Takes you back to the root directory (D: in this case).

1. ***D:\>cd*** masm [enter]
2. ***D:\ masm\ edit .asm*** [enter]
3. Type the program
4. Save in the same drive with extension ***filename.asm***
5. ***File***🡪 ***exit*** [enter]
6. ***D:\MASM /ZI filename.asm;*** [enter]
7. ***D:\LINK /CO filename .OBJ;*** [enter]
8. ***D:\CV/43 filename*** [enter]

This will open the program in debugger screen where in you can view the assemble code with the CS and IP values at the left most side and the machine code. Register content and memory content also be viewed using VIEW option of the debugger.

***Execute option in the menu can be used to execute the program either in***

***single steps(F8) or burst execution(F5).***

1. ***Debug Window opens***

***Data File:***

1. ***;L {load} [enter]***
2. ***G [go] [ enter]***
3. ***Q (quit )***
4. ***To Change The Data***
5. ***L(load)***
6. ***EW NUM (*label it for word/ for byte EB num) [enter]**

You get the memory

Eg: 49c2:0000. 24. 67 space 25.68=> i.e memory location 000024 was there it is change to 67 then a space for next location 25 is changed to 68

*20****. DW NUM*** *(dump word)*

21 ***G*** (go)

Follow the procedure as above.

**1.Programs involving Data Transfer Instructions**

**; i) Write an ALP to transfer a block of N=5 word from source block to destination block, without overlap**

.MODEL SMALL

.DATA

SRC\_BLK DW 123H, 2345H, 0ABC4H, 2345H, 8765H

DSTN\_BLK DW 5 DUP(0)

.CODE

MOV AX,@DATA ; Load DS with upper 16-bits of Data-seg Base

MOV DS,AX

LEA SI,SRC\_BLK ; SI is src. ptr

LEA DI,DST\_BLK ; DI is dst. ptr

MOV CX,05

LOC1: MOV AX,[SI]

MOV [DI],AX

INC SI

INC SI

INC DI

INC DI

DEC CX

JNZ LOC1

MOV AH, 4CH

INT 21H

END

; DATA

;L

; DW SRC\_BLK L5

; G

; DW DSTN\_ BLK L5

; Q

Space for result :-

**ii) Program to move data (Byte transfer) with overlap**

**;Write an ALP to transfer a block of N= 10 data bytes with overlap, destination block ;ending on the 6th byte position of the source.**

.MODEL SMALL

.DATA

POS EQU 6

DSTN\_BLK DB 10-(POS+1) DUP(0)

SRC\_BLK DB 11H,22H,33H,44H,55H,66H,77H,88H,99H,0AH

.CODE

MOV AX,@DATA

MOV DS, AX

LEA SI, SRC\_BLK

LEA DI, DSTN\_BLK

MOV CX, 10

LOC1: MOV AL,[ SI]

MOV [DI], AL

INC SI

INC DI

LOOP LOC1

MOV AH, 4CH

INT 21H

END

;Data File:

;L

; DB SRC\_BLK LA ; SRC BLK before execution.

; G

; DB DSTN\_ BLK LA

; DB SRC\_BLK LA

; Q

Assignment: **Writ an ALP to transfer a block of N= 10 data bytes with overlap, destination block starting at the 6th byte position of the source.**

Space for result:-

**iii) Program to perform Block Interchange Of two Blocks of Word Dat**a.

**;Write an ALP to interchange a block of N word from source block labeled as ;X\_BLOCK to destination block labeled as Y\_BLOCK.**

.MODEL SMALL

.DATA

X\_BLOCK DW 0123h,3456h,789ah,0bcdeh,0f231h

Y\_BLOCK DW 1122h,3344h,5566h,7788h,0aabbh

.CODE

MOV AX,@DATA ; Load DS with upper 16-bits of Data-seg Base

MOV DS,AX

LEA SI,X\_BLOCK ; SI Points to X- BLOCK

LEA DI,Y\_BLOCK ; DI Points to Y- BLOCK

MOV CX,0005H ; Length of Block=5

LOC1: MOV AX,[SI]

MOV BX,[DI]

MOV [DI],AX

MOV[SI],BX

INC DI

INC DI

INC SI

INC SI

LOOP LOC1 ; Decrements CX, if CX != 0, jump to LOC1.

MOV AH,4CH terminates Program, return control to DOS.

INT 21H

END

; DATA

;L

; DW X- BLOCK L5

; DW X- BLOCK L5

;G

;DW X- BLOCK L5

; DW X- BLOCK L5

; Q

Assignment : a) **Write an ALP to interchange a block of N word from source block labeled as X\_BLOCK to destination block labeled as Y\_BLOCK using stack operation.**

**b) Write an ALP to interchange a block of N word from source block labeled as X\_BLOCK to destination block labeled as Y\_BLOCK using XCHG operation**

Space for result:-

**2. Programs involving Arithmetic Operation**

**i) Write an ALP to perform addition and subtraction of two 16-bit numbers.**

.MODEL SMALL

.DATA

N1 DW 1239H

N2 DW 0ADC2H

RES DW 4 DUP (0)

DIFF DW 4 DUP (0)

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,N1

MOV BX,N2

MOV DX,0000H

ADD AX,BX

JNC LAB1

INC DX

LAB1: MOV RES,AX

MOV RES+2,DX

MOV AX, N1

MOV BX, N2

MOV DX, 0000H

SUB AX, BX

JNC LAB1

MOV DX, 00FFH

LAB1: MOV DIFF, DX

MOV DIFF+2, AX

MOV AH,4CH

INT 21H

END

; DATA

;L

; G

; DW N1 L1

; DW N2 L2

; Q

**Space for result :- Space for result :-**

**ii) Write an ALP to add two N multi precision numbers. (X+Y = Z, there X and Y are Multi precision numbers each).**

.MODEL SMALL

.DATA

X DB 23H,34H, 56H,78H, 9AH,0BCH, 0DEH,0F2H; X=F2DEBC9A78563423H

Y DB 0F4H, 0E5H, 66H,77H, 0ABH, 88H,0AAH, 0BBH; Y=0BBAA88AB7766E5F4H

Z DB 9 DUP (0)

.CODE

MOV AX,@DATA ; Load DS with upper 16-bits of Data-seg Base

MOV DS,AX

LEA SI,X ; SI Points to Addend –bytes X

LEA DI,Y ; DI Points to Augend to Y Bytes

LEA BX,Z ; BX Points to Result Z Bytes

MOV CX,08H ; Length of Block=8

LOC1: MOV AL,[SI]

ADC AL, [DI]

MOV [BX], AL

INC DI

INC SI

INC BX

LOOP LOC1 ; Decrements CX, if CX != 0, jump to LOC1.

MOV AL ,00

RCL AL,01

MOV AH,4CH terminates Program, return control to DOS.

INT 21H

END

; DATA

;L

; DB X L8

; DW Y L8

;G

Space for result:-

**iii) a) Write an ALP to perform multiplication of two unsigned 32- bit numbers.**

.MODEL SMALL

.DATA

MPD DW 0C432H, 765BH

MPR DW 3679H, 0B397H

RES DW 4 DUP (0)

.CODE

MOV AX, @DATA

MOV DS, AX ; load DS with upper 16\_bits of Data Seg Base

MOV BX, OFFSET MPD; copy the offset address of MPD to pointer BX reg. MOV AX, [BX] ; copy the content from the offset address to AX- multiplier1

MUL MPR ; First Multiplication

MOV RES, AX ; move the lower word result to RES

MOV RES+2, DX ; And higher word result to RES+2

MOV AX,[BX] ; copy multiplier1 to AX

MUL MPR+2 ; Second multiplication

ADD RES+2, AX

ADC RES+4, DX

JNC L1

INC RES+6

L1: MOV AX,[BX+2] ; get the multiplier2 from the memory location BX+2

MUL MPR ; Third multiplication

ADD RES+2, AX

ADC RES+4, DX

JNC L2

INC RES+6

L2: MOV AX, [BX+2] ; get the multiplier 2

MUL MPR+2 ; fourth multiplication

ADD RES+4, AX

ADC RES+6, DX

MOV AH, 4CH ; DOS interrupt 21H, function 4CH

INT 21H ; Terminates program and returns control to DOS

END

**;To Execute the Program**

; DATA FILE

;L

;G

;DW MPD L2

;DW MPR L2

;DW RES L4

;Q

;RESULT FILE

;>< FILE NAME.DAT

;>L

;>G

;Program terminated normally (-83)

;>DW MPD L2

;>4C74 : 0002 C432 765B

;>DW MPR L2

;>4C74: 0006 3679 B397

;>DW RES L4

;>4C74 : 000A 47A2 FC40 137E 5308

;>Q

Space for result :-

**3) a) Write an ALP to check whether the given 8-bit data is a 2out of 5 code**

.MODEL SMALL

.DATA

NUM DB 0ACH

RES DB 3 DUP(?)

.CODE

MOV AX,@DATA

MOV DS, AX

LEA SI, NUM

LEA DI, RES

CALL FINDP

MOV AH,4CH

INT 21H

FINDP PROC NEAR

MOV AX,[SI]

MOV BX,AX

TEST AL,0E0H

JNZ N2O5

MOV CX, 05

BACK0: ROR AL.01

JNC LOC1

INC COUNT

LOC1: LOOP BACK0

CMP COUNT,02

JZ LOC2

N2O5: MOV BYTE PTR [DI],0FFH

LOC2: RET

FINDP ENDP

END

; DATA FILE

;L

;G

;DB NUM L1

;DB RES L3

;Q

Space for result:-

**b)** **Write an ALP to Find the number of 1’s and 0’s, in the given 8-bit data**

.MODEL SMALL

.DATA

NUM DB 7DH

RES DB 2 DUP(?)

N\_1S DB 00H

N\_0S DB 08H

.CODE

MOV AX,@DATA

MOV DS,AX

LEA SI,NUM

LEA DI,RES

CALL FINDP

MOV AH,4CH

INT 21H

FINDP PROC NEAR

MOV AX,[SI]

MOV BX,AX

MOV CX,08

BACK: ROL AL,1

JNC LOC1

INC N\_1S

LOC1: LOOP BACK

MOV AL,N\_1S

MOV BYTE PTR [DI],AL

SUB N\_0S,AL

MOV AL,N\_0S

MOV BYTE PTR 1[DI],AL

RET

FINDP ENDP

END

; DATA FILE

;L

;G

;DB NUM L1

;DB RES L2

; DB N\_1S L1

; DB N\_0S L1

;Q

Space for result:-

**4. Find LCM,HCF and Factorial**

**;i) Write an ALP to find LCM of two 16-bit unsigned integers.**

.MODEL SMALL

.DATA

X DW 50

Y DW 04

Z DW 2 DUP(?)

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,X

MOV BX,AX

MOV DX,0

LOC2: PUSH AX

PUSH DX

DIV Y

CMP DX,0

POP DX

POP AX

JE LOC1

ADD AX,BX

ADC DX,0

JMP LOC2

LOC1: MOV Z,AX

MOV Z+2,DX

MOV AH,4CH

INT 21H

ALIGN 16

END

;DATA FILE: L

; G

; DD Z L1

; Q

Space For Result:

**ii) Write an ALP to find HCF(GCD) of two 16-bit unsigned integers.**

.MODEL SMALL

.DATA

X DW 50

Y DW 04

Z DW ?

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,X

MOV BX,Y

CMP AX,BX

JAE PROCEED

LOC2: XCHG AX,BX

PROCEED: MOV DX,0

DIV BX

CMP DX,0

MOV AX,DX

JNZ LOC2

MOV Z,BX

MOV AH,4CH

INT 21H

END

; DATA FILE: L

; G

; DW Z L1

; Q

Space For Result:

**iii) Write an ALP to find factorial of a number (N≤ 9) using recursion.**

.MODEL SMALL

.STACK 64

.DATA

NUM DW 3

RES DW 2 DUP(0)

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,NUM

CALL FACT

MOV AH,4CH

I NT 21H

FACT PROC NEAR

CMP AX,01H

JA CONT

MOV RES,01

RET

CONT: PUSH AX

DEC AX

CALL FACT

POP AX

MUL RES

MOV RES,AX

MOV RES+2,DX

RET

FACT ENDP

END

;DATA FILE: L

; G

; DD RES L1

; EW NUM 9 ; Enter nos. upto 9

; G

; DD RES L1

; Q

; G

; Program terminated normally (-128)

; DD RES L1

; 4C71:0010 0005:8980

; Q

Space For Result:

**5. Code Conversion**

**i) Write an ALP to convert a given Hexadecimal number --------- to its equivalent BCD number (16 bit). Display the result.**

.MODEL SMALL

.STACK 200H

.DATA

HEX\_NO DW 0AB89H

BCD DB 5 DUP (0)

DIVISOR\_1 DW 2710H

DIVISOR\_2 DW 03E8H

DIVISOR\_3 DW 64H

DIVISOR\_4 DB 0AH

MSG DB ‘THE EQUIVALENT BCD NO IS: $'

CRLF DB 0DH, 0AH, '$'

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AX,HEX\_NO

XOR DX,DX

DIV DIVISOR\_1

MOV BCD+4,AL

MOV AX, DX

XOR DX, DX

DIV DIVISOR\_2

MOV BCD+3, AL

MOV AX, DX

XOR DX,DX

DIV DIVISOR\_3

MOV BCD+2,AL

MOV AX,DX

MOV DX,00

DIV DIVISOR\_4

MOV BCD+1 ,AL

MOV AL, AH

MOV AH,00

MOV BCD,AL

MOV AH,9

MOV DX,OFFSET MSG

INT 21H

CALL DISPLAY

MOV AH, 4CH

INT 21H

DISPLAY PROC NEAR

MOV CX,5

MOV DI,4

LEA BX,BCD

NEXT: MOV DL,[BX+DI]

OR DL,30H

MOV AH,2

INT 21H

DEC DI

LOOP NEXT

MOV AH,9

MOV DX, OFFSET CRLF

INT 21H

RET

DISPLAY ENDP

END

Space for the Result:

**ii) Write an ALP to convert a given BCD number to its equivalent Hexadecimal number (16 bit).**

.MODEL SMALL

.STACK 200

.DATA

BCD\_INPUT DB 45H,67H

HEX\_VALUE DW 0

FACTORS DB 0AH,64H,0E8H,03H

NB\_SEP DB 10H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV SI,OFFSET BCD\_INPUT

MOV DI,OFFSET HEX\_VALUE

CALL BCD\_HEX

MOV AH,4CH

INT 21H

BCD\_HEX PROC NEAR

MOV AX,WORD PTR[SI]

OR AH,00

JZ BYTE\_CONV

MOV BL,AL

MOV AL, AH

MOV AH,0

DIV NB\_SEP

MOV CH,AH

MOV AH,0

MUL WORD PTR FACTORS+2

MOV WORD PTR HEX\_VALUE,AX

MOV AL,CH

MUL FACTORS+1

ADD HEX\_VALUE,AX

MOV AL,BL

MOV AH,0

BYTE\_CONV: DIV NB\_SEP

MOV DL,AH

MOV DH,0

MUL FACTORS

ADD AX,DX

ADD HEX\_VALUE,AX

RET

BCD\_HEX ENDP

END

;RESULT FILE:

; L

; G

; Program terminated normally (6)

; DW BCD\_INPUT L1

;DW HEX-VALUE L1

; Q

Space For Result:

**6. Programs Involving Branch /Loop Instruction**

**i)** **Write an ALP to sort N = 10 words of numbers in ascending order using bubble sort algorithm. Display the result**

.MODEL SMALL

.DATA

X DW DATA 3598H, 5745H, 79A5H, 0B245H,01H, 4005H, 0F12H, 2005H, 0DCABH,1000H, 3456H

Y DW (Y-X)/2

ASC\_TABLE DB ‘0123456789ABCDEF’

NB\_SEP DB 10H

CRLF DB 0DH,0AH,’$’

.CODE

MOV AX,@DATA

MOV DS,AX

NEXT\_PASS:LEA SI,X

MOV CX,Y

DEC CX

MOV BL,00

LOC2: MOV AX,[SI]

CMP AX,[SI+2]

JBE LOC1

XCHG AX, [SI+2]

XCHG AX,[SI]

MOV BL,0FFH

LOC1: INC SI

INC SI

DEC CX

JNZ LOC2

CMP BL,00H

JNE NEXT\_PASS

CALL DISPLAY

MOV AH,4CH

INT 21H

ALIGN 16

DISPLAY PROC NEAR

LEA BP,X

MOV CX,Y

LOC: MOV AX,DS:[DP]

PUSH AX

XCHG AH,AL

CALL DISP

POP AX

CALL DISP

INC BP

INC BP

MOV DL,20H

MOV AH,2

INT 21H

LOOP LOC

MOV AH,9

MOV DX,OFFSET CRLF

INT 21H

RET

DISPLAY ENDP

DISP PROC NEAR

MOV BX,OFFSET ASC\_TABLE

MOV AH,0

PUSH CX

DIV NB\_SEP

MOV CL,AH

XLAT

MOV AH,2

MOV DL,AL

INT 21H

MOV AL,CL

XLAT

MOV DL,AL

MOV AH,2

INT 21H

POP CX

RET

DISP ENDP

END

;DATADILE: L

; ;G

; ;Q

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Assignment : **Write an ALP to sort N = 10 words of numbers in descending order using bubble sort algorithm. Display the result.**

**;ii)** **Write an ALP to find the largest and smallest element in array of N,16-bit unsigned integer. Store the largest and small elements with their position.**

.MODEL SMALL

.DATA

ARRAY DW 0ABCDH,34H,0FFFFH,7867H ,2345H,5678H,012H,01H,7896H,1CBDH

L\_POS DW ?

LARGEST DW ?

S\_POS DW ?

SMALLEST DW ?

COUNT EQU 10

.CODE

MOV AX,@DATA

MOV DS,AX

MOV ES,AX

LEA SI,ARRAY

MOV BP,SI

MOV CX,COUNT-1

MOV BX,[SI]

MOV DI,BX

INC SI

INC SI

LOOP1: MOV AX,[SI]

CMP BX,AX

JAE UP

MOV BX,AX

MOV DX,SI

UP: CMP DI,AX

JBE CONT

MOV DI,AX

MOV S\_POS,SI

CONT: INC SI

INC SI

LOOP LOOP1

MOV LARGEST ,BX

MOV SMALLEST,DI

SUB DX,BP

SHR DX,1

MOV L\_POS,DX

SUB S\_POS,BP

SHR S\_POS,1

MOV AH,4CH

INT 21H

END;

DATA FILE L

; DW ARRAY LA

; G

; DW LARGEST L1

; DW L\_POS L1

; DW SMALLEST L1

; DW S\_POS L1

; Q

Space for Result:

**7. Programs Involving String Manipulation**

**i) Write an ALP to reverse a string , check whether the given is a palindrome or not Display the same message.**

.MODEL SMALL

.DATA

STR1 DB 'WE ARE HAVING GNIVAH ERA EW.'

CRLF DB 0DH,0AH,'$'

LEN EQU CRLF-STR1

STR2 DB LEN DUP(?),'$'

MSG\_PAL DB 0DH,0AH,'THE STRING IS PALINDROME$'

MSG\_NOTPAL DB 0DH,0AH,'THE STRING IS NOT A PALINDORM$'

.CODE

MOV AX,@DATA

MOV DS,AX

MOV ES,AX

LEA SI,STR1

LEA DI,STR2+LEN-1

MOV CX,LEN

MOV BX,DI

CLD

NEXT: LODSB

STOSB

DEC BX

MOV DI,BX

LOOP NEXT

MOV AH,9

MOV DX,OFFSET STR1

INT 21H

MOV DX,OFFSET STR2

INT 21H

MOV SI,OFFSET STR1

MOV DI,OFFSET STR2

MOV CX,LEN-1

CLD

REPE CMPSB

JZ PAL

MOV AH,9

LEA DX,MSG\_NOTPAL;Z=0 NO MATCH

INT 21H

JMP EXIT

PAL: MOV AH,9

LEA DX,MSG\_PAL

INT 21H

EXIT: MOV AH,4CH

INT 21H

ALIGN 16

END

; DATAFILE: L

; G

; DB STR1 LCOUNT

; DB STR2 LCOUNT

; DB STR3 LCOUNT2

; DB STR4 LCOUNT2

; DW WORDSTR5 LCOUNT1

; DW WORDSTR6 LCOUNT1

; Q

Space for result:

**Interfacing Programs**

Steps

1. Type the program in the notepad with extension .asm

2. START🡪 RUN 🡪 CMD

on Command window

3. c>cd sa(folder name where files are stored)

4. c:\sa>masm86

Input File: file name .asm

5. c:\sa>LINK86

Input File: file name.asm

(Hex file is created)

Then go to

6. START🡪 Program 🡪 ESA Trainer 🡪win86

7. Download button is enabled

8. Browse 🡪 hex file🡪ok

9. CMD 🡪 G(enter)

10. 2000(enter)

**9. Interfacing Experiments**

**i) Generation of a square wave, triangular wave generation and stair case waveform using DAC. Display the waveform on a CRO.**

**a. Write an ALP to interface 8086 processor with 8255 PPI to generate the triangular wave.**

**; Assume the interface is connected over J4 of trainer**

**; This program generates a Square or Triangular wave at Xout or Yout**

**; The program can be executed in Stand alone or Serial mode**

**; Execute the program from memory location 2000H**

**OUTPUT 2500AD**

**ORG 2000H**

**MOV AX,0000H**

**MOV CS,AX**

**MOV ES,AX**

**MOV DX,0FFE6H ;Initialise all 8255**

**MOV AL,80H ;ports as O/P ports**

**OUT DX,AL**

**START: MOV CX,0FFH ;set count**

**MOV AL,00H ;start from 0**

**UP: INC AL ;increment data for**

**MOV DX,0FFE0H ;+ive going slope and**

**OUT DX,AL ;output at port A & B**

**MOV DX,0FFE2H**

**OUT DX,AL**

**LOOP UP**

**MOV CX,0FFH ;set count**

**MOV AX,CX ;start from FFh**

**DOWN: DEC AL ;decrement data for**

**MOV DX,0FFE0H ;-ive going slope at**

**OUT DX,AL ;port A & B**

**MOV DX,0FFE2H**

**OUT DX,AL**

**LOOP DOWN**

**JMP SHORT START ;repeat continuously**

**END**

**b. Write an ALP to interface 8086 processor with 8255 PPI to generate the square wave.**

; Assume the interface is connected over J4 of trainer

; This program generates a Square or Triangular wave at Xout or Yout

; The program can be executed in Stand alone or Serial mode

; Execute the program from memory location 2000H

OUTPUT 2500AD

ORG 2000H

MOV AX,0000H

MOV CS,AX

MOV ES,AX

MOV DX,0FFE6H ;Initialise all 8255

MOV AL,80H ;ports as O/P ports

OUT DX,AL

START:

MOV AL,0FFH ;start from FFH

MOV DX,0FFE0H ;+ive going slope and

OUT DX,AL ;output at port A & B

MOV DX,0FFE2H

OUT DX,AL

CALL DELAY

MOV AL,00H ;set count decrement data for

MOV DX,0FFE0H ;-ive going slope at

OUT DX,AL ;port A & B

MOV DX,0FFE2H

OUT DX,AL

CALL DELAY

JMP SHORT START ;repeat continuously

DELAY: MOV CX,094DH ;Delay routine

SS: LOOP SS

RET

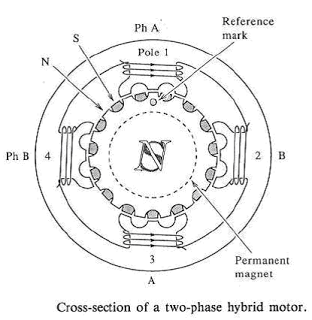
END

**ii) STEPPER MOTOR INTERFACE**

**Apparatus:-**  
Microprocessor trainer kit, stepper motor kit, power supply, data cable etc  
  
THEORY:-  
Stepper motor is a device used to obtain an accurate position control of rotating shafts. A stepper motor employs rotation of its shaft in terms of steps, rather than continuous rotation as in case of AC or DC motor. To rotate the shaft of the stepper motor, a sequence of pulses is needed to be applied to the windings of the stepper motor, in proper sequence. The numbers of pulses required for complete rotation of the shaft of the stepper motor are equal to the number of internal teeth on its rotor. The stator teeth and the rotor teeth lock with each other to fix a position of the shaft. With a pulse applied to the winding input, the rotor rotates by one teeth position or an angle x. the angle x may be calculated as.

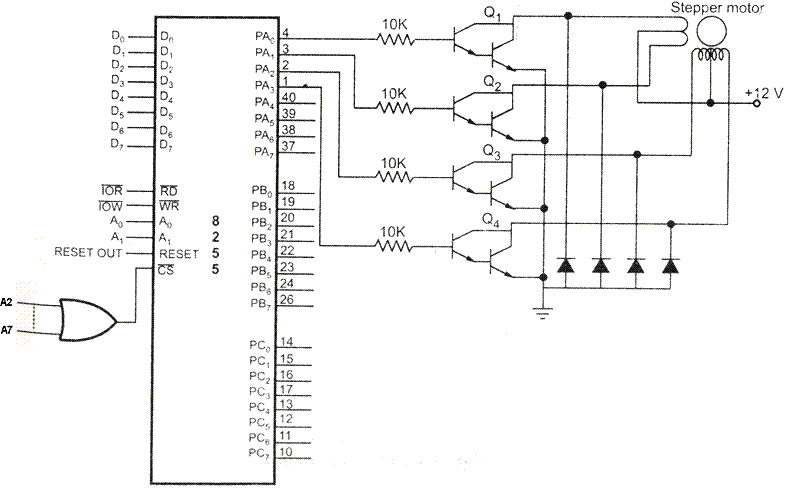
x = 3600 / no. of rotor teeth

After the rotation of the shaft through angle x, the rotor locks it self with the next tooth in the sequence on the internal surface of the stator. The typical schematic of a typical stepper motor with four windings is as shown below.



The stepper motors have been designed to work with digital circuits. Binary level pulses of 0-5V are required at its winding inputs to obtain the rotation of the shafts. The sequence of the pulses can be decided, depending upon the required motion of the shaft.

WORKING:-  
8255 is interfaced with 8086 in I/O mapped I/O. port C (PC0, PC1, PC2, PC3) is used to give pulse sequence to stepper motor. The 8255 provides very less current which will not be able to drive stepper motor coils so each of the winding of a stepper motor needs to be interfaced using high speed switching Darlington transistors with max 1A, 80V rating with heat sink, with the output port of 8255. Output the sequence in correct order to have the desired direction to rotate the motor.



**ii)a)** **Write an ALP to interface 8086 processor with 8255 PPI to rotate the stepper motor in anti-clockwise direction**

; Assume the interface is connected over J4 of the trainer.

; This program illustrates the control of direction of

; Rotation of the Stepper motor depending upon user choice.

; The program executes in a continuous loop.

; The program can be executed in STAND-ALONE MODE or SERIAL

; MODE of operation.

; The program starts at memory location 0:2000H

; Please refer ESA 86/88E user's manual for mnemonic

; Syntax suitable to trainer

OUTPUT 2500AD

ORG 2000H

MOV AX,0000H ;Initialise Segment

MOV ES,AX ;Registers

MOV DX,0FFE6H ;Initialise

MOV AL,80H ;all 8255 Ports as o/p

OUT DX,AL

START: MOV AL,11H ;Output value

MOV DX,0FFE0H ;to Port A

R1: OUT DX,AL

CALL DELAY ;Introduce delay

RCL AL,1 ;Rotate bits in

JMP SHORT R1 ;data byte right & repeat

DELAY: MOV CX,800H ;Delay routine

SS: LOOP SS

RET

END

**ii)b)** **Write an ALP to interface 8086 processor with 8255 PPI to rotate the stepper motor in clockwise direction**

; Assume the interface is connected over J4 of the trainer.

; This program illustrates the control of direction of

; rotation of the Stepper motor depending upon user choice.

; The program executes in a continuous loop.

; The program can be executed in STAND-ALONE MODE or SERIAL

; MODE of operation.

; The program starts at memory location 0:2000H

; Please refer ESA 86/88E user's manual for mnemonic

; syntax suitable to trainer.

OUTPUT 2500AD

ORG 2000H

MOV AX,0000H ;Initialise Segment

MOV ES,AX ;Registers

MOV DX,0FFE6H ;Initialise

MOV AL,80H ;all 8255 Ports as o/p

OUT DX,AL

START: MOV AL,11H ;Output value

MOV DX,0FFE0H ;to Port A

R1: OUT DX,AL

CALL DELAY ;Introduce delay

RCR AL,1 ;Rotate bits in

JMP SHORT R1 ;data byte right & repeat

DELAY: MOV CX,800H ;Delay routine

SS: LOOP SS

RET

END

PROCEDURE:-

Step 1

1. ***Go to Start, Run, cmd (***or Command Window).
2. ***C :\Document and setting \admin> cd\*** [enter]
3. ***C > cd folder name***
4. C> masm86

***input file: filename*** ( enter until debug is completed)

5. C> Link86

***input file: filename(Hex file is generated)***

1. Connect power supply 5V & GND to both microprocessor trainer kit & Stepper motor interfacing kit.
2. Connect data bus between microprocessor trainer kit & Stepper motor interfacing kit.
3. Download the HEX file.
4. Enter the program to rotate Stepper motor in clockwise & anticlockwise.
5. Before executing the program press reset then in cmd type G 2000 ENTER for clockwise, and for anti-clockwise type g 2000 ENTER

Observe the rotation of stepper motor