MICROPROCESSORS AND INTERFACING

LABORATORY MANUAL

(FOR CSE III Year- I Sem)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

(Sponsored by CMR Educational Society)

(Affiliated to JNTU, Hyderabad)

Secunderabad-14.

TABLE OF CONTENTS

1. INTRODUCTION TO MASM

EDITOR

An editor is a program, which allows you to create a file containing the assembly language statements for your program. As you type in your program, the editor stores the ASCII codes for the letters and numbers in successive RAM locations. When you have typed in all of your programs, you then save the file on a floppy of hard disk. This file is called source file. The next step is to process the source file with an assembler. In the MASM /TASM assembler, you should give your source file name the extension, .ASM

ASSEMBLER

An assembler program is used to translate the assembly language mnemonics for instructions to the corresponding binary codes. When you run the assembler, it reads the source file of your program the disk, where you saved it after editing on the first pass through the source program the assembler determines the displacement of named data items, the offset of labels and pails this information in a symbol table. On the second pass through the source program, the assembler produces the binary code for each instruction and inserts the offset etc that is calculated during the first pass. The assembler generates two files on floppy or hard disk. The first file called the object file is given the extension. OBJ. The object file contains the binary codes for the instructions and information about the addresses of the instructions. The second file generated by the assembler is called assembler list file. The list file contains your assembly language statements, the binary codes for each instructions and the offset for each instruction. In MASM/TASM assembler, MASM/TASM source file name ASM is used to assemble the file. Edit source file name LST is used to view the list file, which is generated, when you assemble the file.

LINKER

A linker is a program used to join several object files into one large object file and convert to an **exe** file. The linker produces a link file, which contains the binary codes for all the combined modules. The linker however doesn't assign absolute addresses to the program, it assigns is said to be reloadable because it can be put anywhere in memory to be run. In MASM/TASM, LINK/TLINK source filename is used to link the file.

DEBUGGER

A debugger is a program which allows you to load your object code program into system memory, execute the program and troubleshoot are debug it the debugger allows you to look at the contents of registers and memory locations after your program runs. It allows you to change the contents of register and memory locations after your program runs. It allows you to change the contents of register and memory locations and return the program. A debugger also allows you to set a break point at any point in the program. If you inset a breakpoint the debugger will run the program upto the instruction where the breakpoint is set and stop execution. You can then examine register and memory contents to see whether the results are correct at that point. In MASM/TASM, td filename is issued to debug the file.

DEBUGGER FUNCTIONS:

1. Debugger allows looking at the contents of registers and memory locations.
2. We can extend 8-bit register to 16-bit register which the help of extended register option.
3. Debugger allows setting breakpoints at any point with the program.
4. The debugger will run the program upto the instruction where the breakpoint is set and then stop execution of program. At this point, we can examine registry and memory contents at that point.
5. With the help of dump we can view register contents.
6. We can trace the program step by step with the help of F7.
7. We can execute the program completely at a time using F8

The DOS -Debugger:

The DOS “Debug” program is an example of simple debugger that comes with MS-DOS. Hence it is available on any PC .it was initially designed to give the user the capability to trace logical errors in executable file.

Below, are summarized the basic DOS - Debugger commands

COMMAND	**SYNTAX**
Assemble	A [address]
Compare	C range address
Dump	D [range]

Enter	E address [list]
Fill	F range list
Go	G [=address] [addresses]
Hex	H value1 value2
Input	I port
Load	L[address] [drive][first sector][number]
Move	M range address
Name	N[pathname][argument list]
Output	O port byte
Proceed	P [=address][number]
Quit	Q
Register	R[register]
Search	S range list
Trace	T [=address][value]
Unassembled	u [range]
Write	W[address][drive][first sector][number]

MS-MASM:

Microsoft's Macro Assembler (MASM) is an integrated software package Written by Microsoft Corporation for professional software developers. It consists of an editor, an assembler, a linker and a debugger (Code View). The programmer's workbench combines these four parts into a user-friendly programming environment with built in on line help. The following are the steps used if you are to run MASM from DOS

MICROPROCESSOR LAB EXECUTION PROCEDURE

STEP1: **Opening the DOS prompt**

Click **start** menu button and click on **Run** and then type *cmd at* command prompt immediately DOS window will be appeared

STEP2: **Checking the MASM installation**

To know MASM is installed or not simply type ***masm*** *at the command prompt* upon that it replies masm version vendor (Microsoft), etc... If you get any error there is no masm in that PC

STEP3: Directory changing (create a folder with your branch and no in D drive)

Change the current directory to your won directory suppose your folder in **D** drive type the following commands to change the directory at command prompt type ***D:*** hit enter now you are in ***D* drive** type **cd folder name** hit the enter

Ex. **D cd ece10**

Now we are in folder cse10

STEP4: **writing the program**

At the command prompt type the **edit programname.asm**

Ex. Edit add.asm

Immediately editor window will open and there you have to write the program. Type the program in that window after completion save the Program, to save the program Go to ***file*** opt in the menu bar and select save opt now your code is ready to Assemble.

STEP5: A**ssembling, Linking and Executing the program**

Go to *file* opt click *exit* opt now DOS prompt will be displayed to assemble the program type the following commands at the DOS prompt

Masm Program Name, Program Name, Program Name, Program Name hit the enter

Ex: Masm add, add, add, add enter

OR

Ex: Masm add.asm

If there are any errors in the program assembler reports all of them at the command prompt with line no's, if there are now bugs your ready to link the program. To link the program type the following line at command prompt Link program name**,,,,,** (5 commas)

Ex: Link add,,,,,

OR

Ex: link add.obj

After linking you are ready to execute the program. To execute the program type the following command

Debug program name.exe hit the enter

Ex: Debug add.exe

Now you entered into the execution part of the program here you have to execute the program instruction by instruction (debugging) first of all press the ***r*** key(register) ***hit*** the enter key it'll displays all the registers and their initial values in HEXDECIMAL note down the values of all the register which are used in the program. To execute the next instruction press ***t*** key (TRACE) hit the enter it'll execute that instruction and displays the contents of all the register. You have to do this until you reach the last instruction of the program. After execution you have to observe the results (in memory or registers based on what you have written in the program).

STEP6: Copying list file (common for all programs):

A list file contains your code starting address and end address along with your program .For every program assembler generates a list file at your folder, programname.lst (ex. Add.lst) you should copy this to your lab observation Opening a list file

Edit programname.lst

Ex. Edit add.lst

EXPERIMENT NO-1

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to add, subtract and multiply two 16 bit unsigned numbers. Store the result in extra segment.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

Programs:

(A) 16-BIT ADDITION

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA
DATA SEGMENT
        NUM1 DW 23C5H
        NUM2 DW 6789H
DATA ENDS
EXTRA SEGMENT
        RESULT DW   0000H
EXTRA ENDS
CODE SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            MOV AX,EXTRA
            MOV ES,AX
            SUB AX,AX
            MOV AX, NUM1
            MOV BX, NUM2
            ADD AX,BX
            MOV RESULT,AX
            INT 3H
CODE ENDS
END START
```

Result:

INPUT: NUM1 23C5H
NUM2 6789H

OUTPUT: RESULT 8B4EH

(B) 16-BIT SUBTRACTION

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
DATA SEGMENT
        NUM1 DW   23C5H
        NUM2 DW  1789H
        RESULT DW   0000H
DATA ENDS
EXTRA SEGMENT
        RESULT DW   0000H
EXTRA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        SUB AX,AX
        MOV AX, NUM1
        MOV BX, NUM2
        SUB AX,BX
        MOV RESULT,AX
        INT 3H
CODE ENDS
END START
```

Result:

INPUT: NUM1 23C5H
NUM2 1789H

OUTPUT: RESULT 0C3CH

(C) 16-BIT MULTIPICATION

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA

DATA SEGMENT
        NUM1 DW 23C5H
        NUM2 DW 1789H
DATA ENDS

EXTRA SEGMENT
        RESULT DW   0000H
EXTRA ENDS
CODE SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            MOV AX,EXTRA
            MOV ES,AX
            SUB AX,AX
            MOV AX, NUM1
            MOV BX, NUM2
             MUL AX,BX
            MOV DI,OFFSET RESULT
            MOV [DI],AX
            MOV RESULT,AX
            INT 3H
CODE ENDS
END START
```

Result:

INPUT: NUM1 23C5H

NUM2 1789H

OUTPUT: DX:AX 0349D76DH

EXPERIMENT NO-2

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to divide a 32 bit unsigned number by a 16 bit unsigned numbers. Store the result in stack segment.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

PROGRAM:

```
ASSUME CS:CODE,DS:DATA,SS:STACK
DATA SEGMENT
        NUM1 DD   2A2CFA6H
        NUM2 DW  590H
DATA ENDS

STACK SEGMENT
        QUOITENT DW ?
        REMAINDER DW ?
STACK ENDS

CODE SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            MOV AX,STACK
            MOV SS,AX
            SUB AX,AX
            MOV SI,OFFSET  NUM1
            MOV AX,[SI]
            INC SI
            INC SI
```

```
        MOV DX,[SI]
        MOV BX, NUM2
        DIV BX
        MOV QUOITENT,AX
        MOV REMAINDER,DX
        INT 3H
CODE ENDS
END START
```

Result:

INPUT: NUM1 2A2CFA6H

NUM2 590H

OUTPUT: AX: DX:

EXPERIMENT NO-3

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to sort the given array of numbers in ascending and decending order.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

Programs:

(A) ASCENDING ORDER

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        ARRAY DB 0500H,0300H,0001H,0004H,0002H
        COUNT DB 05H
DATA  ENDS
CODE  SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            SUB AX,AX
            MOV DX,COUNT-1
            AGAIN:MOV CX,DX
            MOV SI,OFFSET ARRAY
BACK:       MOV AX,[SI]
            CMP AX,[SI+2]
            JBE FORW
            XCHG AX,[SI+2]
            MOV [SI],AX
FORW:       INC SI
            INC SI
            LOOP BACK
            DEC DX
            JNZ AGAIN
            INT 03H
CODE ENDS
END START
```

Result:

INPUT:

OUTPUT:

DECENDING ORDER

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        ARRAY DB 0500H,0300H,0001H,0004H,0002H
        COUNT DB 05H
DATA  ENDS
CODE  SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            SUB AX,AX
            MOV DX,COUNT-1
AGAIN:      MOV CX,DX
            MOV SI,OFFSET ARRAY
BACK:       MOV AX,[SI]
            CMP AX,[SI+2]
            JAE FORW
            XCHG AX,[SI+2]
            MOV [SI],AX
FORW:       INC SI
            INC SI
            LOOP BACK
            DEC DX
            JNZ AGAIN
            INT 03H
CODE ENDS
END START
```

Result:

INPUT:

OUTPUT:

EXPERIMENT NO-4

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to pick the median from the given array of numbers.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM SOFTWARE

Program:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        ARRAY DB 05H,03H,01H,04H,02H
        COUNT DB 05H
        MEDIAN DB ?
DATA ENDS
CODE SEGMENT
START:  MOV AX,DATA1
        MOV DS,AX
        SUB AX,AX
        MOV AL,COUNT
        MOV SI,OFFSET ARRAY
        MOV DI,OFFSET MEDIAN
        DIV AL,02H
        INC AL
        ADD SI,AL
        MOV AX,[SI]
        MOV [DI],AX
        MOV AH,4CH
        INT 03H
CODE ENDS
END START
```

Result:

INPUT:

OUTPUT:

EXPERIMENT NO-5

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to find the length of a given string which terminates with a special character.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

Program:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        STRING1 DB 'MICROPROCESSOR AND INTERFACING LAB$'
        RES DB 0
DATA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        SUB CL,CL
        MOV SI,OFFSET STRING1
BACK:   LODSB
        INC CL
        CMP AL,'$'
        JNZ BACK
        MOV RES,CL
        INT 03H
CODE ENDS
END START
```

Result:

INPUT: 'MICROPROCESSOR AND INTERFACING LAB$'

OUTPUT: 23H

EXPERIMENT NO-6

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to reverse the given string and verify whether it is a palindrome.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

Program:

```
ASSUME CS: CODE, DS: DATA, ES:EXTRA
DATA SEGMENT
        STRING1 DB 'MICROPROCESSOR'
        STRLEN EQU ($-STRING1)
DATA ENDS
EXTRA SEGMENT
        ORG 0030H
        STRING2 DB   00H
EXTRA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        MOV SI,OFFSET STRING1
        MOV DI,OFFSET STRING2
        ADD  DI,30
        CLD
        MOV CX,STRLEN
AGAIN:  DEC DI
        MOV AL,[SI]
        MOV [DI],AL
        INC  SI
        DEC CX
        JNZ  AGAIN
        INT 3H
```

CODE ENDS

END START

Result:

INPUT:

D DS:0000 000D MICROPROCESSOR

OUTPUT:

D ES:0030 003D ROSSECORPORCIM

EXPERIMENT NO-7

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to verify the password.

Software Required:

Pc with windows(95/98/XP/NT/2000)

MASM Software

Program:

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
DATA SEGMENT
        STRING1 DB 'MRCET'
        STRLEN EQU ($-STRING1)
        SANE DB 'STRINGS ARE UNEQUAL$'
        SAE DB 'STRINGS ARE EQUAL$'
DATA ENDS
EXTRA SEGMENT
        STRING2 DB 'MRCET'
EXTRA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        MOV SI,OFFSET STRING1
        MOV DI,OFFSET STRING2
        CLD
        MOV CX,STRLEN
        REP CMPSB
        JZ GO
        MOV AH,09H
        MOV DX,OFFSET SANE
        INT 21H
        JMP EXITP
```

```
GO:     MOV AH,09H
        MOV DX,OFFSET SAE
        INT 21H
EXITP:  INT 03H
CODE ENDS
END START
```

Result:

EXPERIMENT NO-8

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to insert or delete a character/ number from the given string.

Software Required:

Pc with windows(95/98/XP/NT/2000)

MASM Software

Program:

(A) STRING INSERTION

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
DATA SEGMENT
        STRING1 DB 'MICROPROCESSOR INTERFACING LAB$'
        STRING2 DB ‘AND ’
        STRLEN EQU ($-STRING1)
DATA ENDS
EXTRA SEGMENT
        STRING3 DB 38 DUP(0)
EXTRA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        MOV SI,OFFSET STRING1
        MOV DI,OFFSET STRING3
        CLD
        MOV CX,15
        REP MOVSB
        CLD
        MOV SI,OFFSET STRING2
        MOV CX,4
        REP MOVSB
        MOV SI,OFFSET STRING1
```

```
        MOV SI,15
        MOV CX,15
        REP MOVSB
        INT 3H
CODE ENDS
END START
```

Result:

INPUT: **STRING1:** 'MICROPROCESSOR INTERFACING LAB'
STRING2:‘AND ’

OUTPUT: **STRING3:** ‘MICROPROCESSOR AND INTERFACING LAB'

(B) STRING DELETION

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
DATA SEGMENT
    STRING1 DB 'MICROPROCESSOR AND INTERFACING LAB$'
    STRLEN EQU ($-STRING1)
DATA ENDS
EXTRA SEGMENT
    STRING2 DB STRLEN-3 DUP(0)
EXTRA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,EXTRA
        MOV ES,AX
        MOV SI,OFFSET STRING1
        MOV DI,OFFSET STRING2
        CLD
        MOV CX,15
        REP MOVSB
        CLD
```

```
        MOV SI,19
        MOV CX,15
        REP MOVSB
        INT 3H
CODE ENDS
END START
```

Result:

INPUT: **STRING1:** MICROPROCESSOR AND INTERFACING LAB'

OUTPUT: **STRING2:** 'MICROPROCESSOR INTERFACING LAB'

EXPERIMENT NO-9

Aim: To Write and execute an Assembly language Program (ALP) to 8086 processor to call a delay subroutine and display the character on the LED display.

Apparatus:

1. Pc with windows(95/98/XP/NT/2000)
2. UxAsm software
3. 8086 trainer kit
4. RS232 Cable

Program:

```
 ORG 0FFFF0H
JMPF 0F000H, 0F000H                ; the basic reset jump
ORG 0FF000H
START    MOV SP, STKPTR            ; load stack pointer
         CALL INIT8255             ; initialize 8255
         MOV AL,00H                ;
         MOV BX,NUM1               ;
         MOV [BX],AL               ; clear num1 variable
LP1      CALL DELAY                ; call delay
         MOV AL,[BX]               ; use num1 for led count value
         MOV DX,PA8255             ;
         OUT DX,AL                 ;
         MOV DX,PB8255             ;
         OUT DX,AL                 ;
         MOV DX,PC8255             ;
         OUT DX,AL                 ;
         INC AX                    ; increment counter
         MOV [BX],AL               ; save for reuse
         JNE LP1                   ; check for loop count end
         JMP START                 ; restart again
;*******  Delay module    ********
         DELAY      NOP            ;
         MOV CX, 03500H            ; load Delay count = 0x3500
         NOP                       ;
```

```
        DLY1  NOP               ;
              LOOP DLY1         ;
              RET               ; end of delay
;*******     initialize 8255   *********
              INIT8255
              MOV AL,080H       ; make all ports output
              MOV DX, CMD8255   ;
              OUT DX,AL         ; write to command register
              MOV AL,00H        ; clear all ports
              MOV DX,PA8255     ;
              OUT DX,AL         ;
              MOV DX,PB8255     ;
              OUT DX,AL         ;
              MOV DX,PC8255     ;
              OUT DX,AL         ;
              RET               ;
```

Result:

EXPERIMENT NO-10

Aim: To interface a keyboard to 8086 microprocessor and display the key number pressed on the 7-segment display which is also interfaced to 8086.

Apparatus:

5. Pc with windows(95/98/XP/NT/2000)
6. UxAsm software
7. 8086 trainer kit
8. RS232 Cable

Program:

```
ORG 0FFFF0H
JMPF 0F000H,0F000H              ; the basic reset jump
ORG 0FF000H
START   MOV AX,0H               ;
        MOV SS,AX               ;
        MOV AX,0F000H           ;
        MOV DS,AX               ;
        MOV SP,STKPTR           ; load stack pointer
        CALL INIT8279           ; initialize 8279
LOOP  CALL GETKEY               ; read status
        CALL RDKBD              ; we got keynumber
        CALL WRSEG              ;
        JMP LOOP                ;

;*******   READ Status Module   *********
GETKEY  MOV DX,CMD8279          ;
        IN AL,DX                ;
        AND AL,07H              ;
        JE GETKEY               ;
        RET                     ; yes there is a keypress
;*******   READ KBD Module     **************
RDKBD   MOV AL,050H             ; read FIFO sensor
        MOV DX,CMD8279          ;
        OUT DX,AL               ;
```

```
                MOV DX,DAT8279          ;
                IN  AL,DX               ; keycode read by cpu
                AND AL,01FH             ; range 00-1F
                MOV BX,KBDTBL           ;
                MOV BL,AL               ;
                MOV AL,[BX]             ; get keynumber
                RET                     ;
;*******        8279 hex string data(0-F)       ************
WRSEG           MOV BX,SEGTBL           ;
                MOV BL,AL               ;
                MOV AL,[BX]             ;
                MOV BP,AX               ; save it temp
                MOV AL,083H             ;
                MOV DX,CMD8279          ;
                OUT DX,AL               ;
                MOV AX,BP               ; reload segdata
                MOV DX,DAT8279          ;
                OUT DX,AL               ;
                WRET RET                ;
;*******        Delay module      **********
DELAY           NOP                     ;
                MOV CX,03500H           ; load Delay count = 0x3500
                NOP                     ;
                DLY1  NOP               ;
                LOOP DLY1               ;
                RET                     ; end of delay
;*******        initialize 8279 = ( cmd = 08,3F,80) -> (data =string)
; Clears all display
INIT8279
                MOV AL,08H              ; 16 character of 8 bit left entry
                MOV DX,CMD8279          ; write to command register
                OUT DX,AL               ;
                MOV AL,03FH             ; 5Mhz clock dividied by 31
```

```
                MOV DX,CMD8279      ; write to command register
                OUT DX,AL           ;
                MOV BP,080H         ;
                MOV CX,0FH          ; loop for 16 digits
IL0             MOV DX,CMD8279      ; choose the first digit out of 16
                MOV AX,BP           ; 80 = 1st , 81 = 2nd and so on
                OUT DX,AL           ;
                MOV AX,00H          ;
                MOV DX,DAT8279      ; clear display
                OUT DX,AL           ;
                INC BP              ;
                LOOP IL0            ;
                RET                 ;
ORG 0FF500H
;*******       initialize seven segment table   ***********
SEGTBL          HEX    3F,06,5B,4F,66,6D,7D,07,7F,6F,77,7C,39,5E,79,71,00
ORG 0FF600H
KBDTBL                                                                  HEX
        00,01,02,03,00,00,00,00,04,05,06,07,00,00,00,00,08,09,0A,0B,00,00,00,00,0C,0D
,0E,0F,00,00,00,00,00
```

EXPERIMENT NO-11

Aim: To interface an 8086 microprocessor trainer kit to PC and establish a communication in between them through RS232.

Apparatus:

1. Pc with windows(95/98/XP/NT/2000)
2. UxAsm software
3. 8086 trainer kit
4. RS232 Cable

Program:

```
ORG 0000H
JMP START                       ;
ORG 0038H                       ;
;********** Software interrupt service routine     **************
REGREC      SHLD REGHL          ; save regHL
            PUSH PSW            ;
            POP H               ;
            SHLD REGAF          ; save regAF
            POP H               ; get PC
            DCX H               ; take care for RST 7
            SHLD REGPC          ; save regPC
            LXI H,0000H         ;
            DAD SP              ;
            SHLD REGSP          ; save regSP
            LXI SP,MONSTK       ; load moitor stack
            PUSH B              ;
            POP H               ;
            SHLD REGBC          ; save regBC
            PUSH D              ;
            POP H               ;
            SHLD REGDE          ; save regDE
            LXI H,LFTBL         ;
            CALL SERMSG         ;
            MVI A,023H          ; print #
            CALL PUTCHAR        ;
            LHLD REGPC          ;
            SHLD BYTE16         ;
            CALL PRHEX16        ; print PC
            JMP CMDLOOP         ;

ORG 0100H
START       LXI SP,MONSTK       ; load monitor stack pointer
            CALL ZEROCMD        ;
            CALL INIT8251       ; initialize 8251
            CMDLOOP    DI       ; Disable interrupt
```

```
                LXI SP,MONSTK           ; load monitor stack pointer
                LXI H,LFTBL             ;
                CALL SERMSG             ;
                LXI H,ADMTBL            ; print signon
                CALL SERMSG             ;
                CALL GETCHAR            ; get one byte command
                CALL PUTCHAR            ; echo back
                CALL GETCMD             ;
                JMP CMDLOOP             ;
;******* Get Command Module      ***********
GETCMD          LXI H,HLPTBL            ;
                CPI 048H                ;
                JZ  SERMSG              ; help command
                CPI 044H                ;
                JZ  DUMPCMD             ; Dump command
                CPI 047H                ;
                JZ  GOCMD               ; Go command
                CPI 52H                 ;
                JZ  REGCMD              ; Register command
                CPI 53H                 ;
                JZ  SMEMCMD             ;
                CPI 05AH                ;
                JZ  ZEROCMD             ; Zero command
                RET                     ;
;******** Zero Command Module     ************
ZEROCMD         LXI H,07000H            ;
ZC1             MVI A,00H               ;
                MOV M,A                 ;
                INX H                   ;
                MOV A,L                 ;
                CPI 01FH                ;
                JNZ ZC1                 ;
                LXI H,3000H             ;
                SHLD REGPC              ; PC=3000
                SHLD DUMP               ;
                SHLD SETMEM             ;
                LXI H,5FFFH             ;
                SHLD REGSP              ;
                RET                     ;

;********** GO command         ************
GOCMD
                LHLD REGSP              ;BACK TO USER STACK
                LXI  SP,0               ;
                SPHL                    ;
                LHLD REGBC              ;
                PUSH H                  ;
                POP B                   ;
                LHLD REGDE              ;
```

```
                PUSH H                  ;
                POP D                   ;
                LHLD REGHL              ;
                LHLD REGAF              ;
                PUSH H                  ;
                POP PSW                 ;
                JMP 03000H              ;

;******* SETMEM Command Module     **************
SMEMCMD   LXI H,LFTBL;
                CALL SERMSG             ;
                LHLD SETMEM             ; get dump address
                SHLD BYTE16             ; print this address
                CALL PRHEX16            ;
                CALL PRSPC              ; print space(20h)
LHLD SETMEM                             ;
                MOV A,M                 ;
                STA BYTE8               ;
                CALL PRHEX8             ; show current data
                CALL PRSPC              ;
                CALL GETHEX8            ;
                LHLD SETMEM             ;
                LDA BYTE8               ; get user byte
                MOV M,A                 ; save the byte
                SM1    INX H            ;
                SHLD SETMEM             ;
                JMP SMEMCMD             ; loop unitl error

;******* Dump Command Module       ***************
DUMPCMD LXI H,LFTBL                     ;
                CALL SERMSG             ;
                LHLD DUMP               ; get dump address
                SHLD BYTE16             ; print this address
                CALL PRHEX16            ;
                CALL PRSPC              ; print space(20h)
LD1     LHLD DUMP                       ; get dump address
                MOV A,M                 ; get location data
                STA BYTE8               ; save data
                CALL PRHEX8             ; print the byte
                CALL PRSPC              ;
                LHLD DUMP               ;
                INX H                   ;
                SHLD DUMP               ;
                MOV A,L                 ;
                ANI 0FH                 ;
                JNZ LD1                 ;
                RET                     ;

;********** PRHEX16     *****************
```

```
PRHEX16     LHLD BYTE16                 ;
            MOV A,H             ;
            STA BYTE8           ;
            CALL PRHEX8         ;
            LHLD BYTE16         ;
            MOV A,L             ;
            STA BYTE8           ;
            CALL PRHEX8         ;
            RET                 ;

;********** PRHEX8 ***************
PRHEX8      LXI H, BYTE8                ;
            MOV A, M            ;
            RAR                 ;
            RAR                 ;
            RAR                 ;
            RAR                 ;
            ANI 0FH             ;
            CALL HEX2ASC        ;
            CALL PUTCHAR        ;
            LXI H,BYTE8         ;
            MOV A,M             ;
            ANI 0FH             ;
            CALL HEX2ASC        ;
            CALL PUTCHAR        ;
            RET                 ;
;********** HEX2ASCII Module     ***********
HEX2ASC
            LXI H,ASCZF         ;
            MOV L,A             ;
            MOV A,M             ;
            RET                 ;
;********** ASC2HEX Module   **********
ASC2HEX
            LXI H,ASCZF         ;
            MOV B,A             ; save copy
AH2         MOV A,M             ; get first digit
            CMP B               ; compare
            JZ AH1              ;
            INX H               ; get next digit
            MOV A,M             ;
            CPI 00H             ;
            JZ ERRHEX           ;
            JMP AH2             ;
AH1         MOV A,L             ; L contains digit number(0-F)
            RET                 ;
;******* Serial Messaging Module   **********
SERMSG      MOV A,M             ;
            CALL PUTCHAR        ; output data to serial port
```

```
                INX H                   ;
                MOV A,M                 ;
                CPI 00H                 ;
                JNZ SERMSG              ; signon over now get command
                RET                     ;
;******** Print Space Module   ********
PRSPC           LXI H,SPCTBL            ;
                CALL SERMSG             ;
                RET                     ;
;******** Print Space Module   *******
ERRHEX          LXI H,ERHTBL            ;
                CALL SERMSG             ;
                JMP CMDLOOP             ;
;******* Get Hex byte from serial port     ********
GETHEX8         CALL GETCHAR            ; get hexvalue
                CALL PUTCHAR            ;
                CPI 020H                ; check for space
                RZ                      ;
                CALL ASC2HEX            ; we get digit in A
                RAL                     ;
                RAL                     ;
                RAL                     ;
                RAL                     ;
                ANI 0F0H                ;
                STA BYTE8               ;
                CALL GETCHAR            ;
                CALL PUTCHAR            ;
                CALL ASC2HEX            ; we get next digit in A
                ANI 0FH                 ;
                MOV B,A                 ;
                LDA BYTE8               ;
                ORA B                   ;
                STA BYTE8               ;
                RET                     ;
;*******        Get Character from Serial Port  **********
GETCHAR MVI A,00H                       ;
GC1     IN CMD8251                      ;
                ANI 02H                 ;
                JZ GC1                  ; no character
                IN DAT8251              ;
                RET                     ;
;*******        Put Character to Serial Port  88888888888  *********
PUTCHAR         MOV B,A                 ;
PC1             IN CMD8251              ;
                ANI 04H                 ;
                JZ PC1                  ;
                MOV A,B                 ;
                OUT DAT8251             ;
                RET                     ;
```

```
;*******     Delay module   ********

DELAY       NOP                     ;
            MVI A,0FFH              ; load lsb of delay=0x34FF
            MVI B,080H              ;
DLY1  DCR A                         ; decrement lsb count
            JNZ DLY1                ;
            DCR B                   ;
            JNZ DLY1                ;
            RET                     ; end of delay

;*******     initialize 8251   **********
INIT8251
            MVI A,00FH              ;
            OUT CMD8251             ;
            OUT CMD8251             ;
            OUT CMD8251             ;
            MVI A,040H              ;
            OUT CMD8251             ;
            MVI A,0CEH              ;
            OUT CMD8251             ;
            MVI A,015H              ;
            OUT CMD8251             ;
            RET                     ;
;******** Dump Registers   *********
REGCMD      LXI H,LFTBL             ;
            CALL SERMSG             ;
            LXI H,REGTBL            ;
            CALL SERMSG             ;
            LXI H,LFTBL             ;
            CALL SERMSG             ;
            LXI H,REGAF             ; show regA
            INX H                   ;
            MOV A,M                 ;
            STA BYTE8               ;
            CALL PRHEX8             ;
            CALL PRSPC              ;
            LHLD REGBC              ; show regBC
            SHLD BYTE16             ;
            CALL PRHEX16            ;
            CALL PRSPC              ;
            LHLD REGDE              ; show regDE
            SHLD BYTE16             ;
            CALL PRHEX16            ;
            CALL PRSPC              ;
            LHLD REGHL              ; show regHL
            SHLD BYTE16             ;
            CALL PRHEX16            ;
```

```
                CALL PRSPC              ;
                LHLD REGSP              ; show regSP
                SHLD BYTE16             ;
                CALL PRHEX16            ;
                CALL PRSPC              ;
                LHLD REGPC              ; show regPC
                SHLD BYTE16             ;
                CALL PRHEX16            ;
                CALL PRSPC              ;
                LXI H,REGAF             ; show regF
                MOV A,M                 ;
                STA BYTE8               ;
                CALL PRHEX8             ;
                CALL PRSPC              ;
                RET
ORG 0400H
;*******       initialize seven segment table  ***********
ASCZF           HEX   30,31,32,33,34,35,36,37,38,39,41,42,43,44,45,46,00
ADMTBL          ASCII "ADM>"
ENDTBL          HEX   00
SPCTBL          HEX   20,00
LFTBL           HEX   0D,0A,00,00
TEND01          HEX   00
REGTBL          ASCII "A  BC   DE   HL   SP   PC   F "
TEND02          HEX   00
ERHTBL          ASCII "Error: Only Hex values"
TEND03  HEX           00
HLPTBL          ASCII " Dump SetMem Reg Zeroreg Go Brk(ffh)"
TEND04          HEX   00
```

Result:

EXPERIMENT NO-12

Aim: To interface a stepper motor to 8086and operate it in clockwise and anti-clockwise by choosing variable step size.

Apparatus:

1. Pc with windows(95/98/XP/NT/2000)
2. UxAsm software
3. 8086 trainer kit
4. RS232 Cable
5. Stepper motor
6. Connectors

Programs:

STEPPER MOTOR INTERFACING PROGRAM

.8086

```
        .MODEL TINY
        .STACK
        .CODE
        PORTA   EQU 0FFE0H
        PORTB   EQU PORTA+2
        PORTC   EQU PORTA+4
        PCR     EQU PORTA+6
        PCW     EQU 80H
        ORG 2000H
        START: MOV AX,CS
               MOV DS,AX
               MOV DX,PCR
               MOV AL,PCW
               OUT DX,AL
        L1:    MOV CX,200
               MOV AL,88H
               CALL BACK1
               MOV CX,200
               MOV AL,88H
               CALL BACK2
```

```
        JMP L1
BACK1: MOV DX,PORTA
        OUT DX,AL
        RCR AL,1
        CALL DELAY
        LOOP BACK1
        RET
BACK2: MOV DX,PORTA
        OUT DX,AL
        RCL AL,1
        CALL DELAY
        LOOP BACK2
        RET
DELAY:  PUSH CX
        MOV CX,0AFFH
D:      LOOP D
        POP CX
        RET
END START
```

Result:

EXPERIMENT NO-13

Aim: To write an assembly language program for performing the multiplication and division on 16-bit signed data.

Software Required:

1. Pc with windows(95/98/XP/NT/2000)
2. MASM Software

(A) SIGNED 16-BIT MULTIPLICATION

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        N1 DW -1000H
        N2 DW 1324H
DATA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,0000H
        MOV DX,0000H
        MOV AX,N1
        MOV BX,N2
        IMUL BX
        INT 3H
CODE ENDS
END START
```

RESULT:

INPUT :	N1	-1000H
	N2	1324H
OUTPUT:	AX:	C000H
	DX:	FECDH

(B).SIGNED 16- BIT DIVISION

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
        N1 DW 1000H
        N2 DW -0025H
DATA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        XOR AX,AX
        MOV DX,0000H
        MOV AX,N1
        MOV BX,N2
        IDIV BX
        INT 3H
CODE ENDS
END START
```

RESULT:

INPUT:	N1	1000H
	N2	-0025H
OUTPUT:	AX:	FF92H
	DX:	001AH