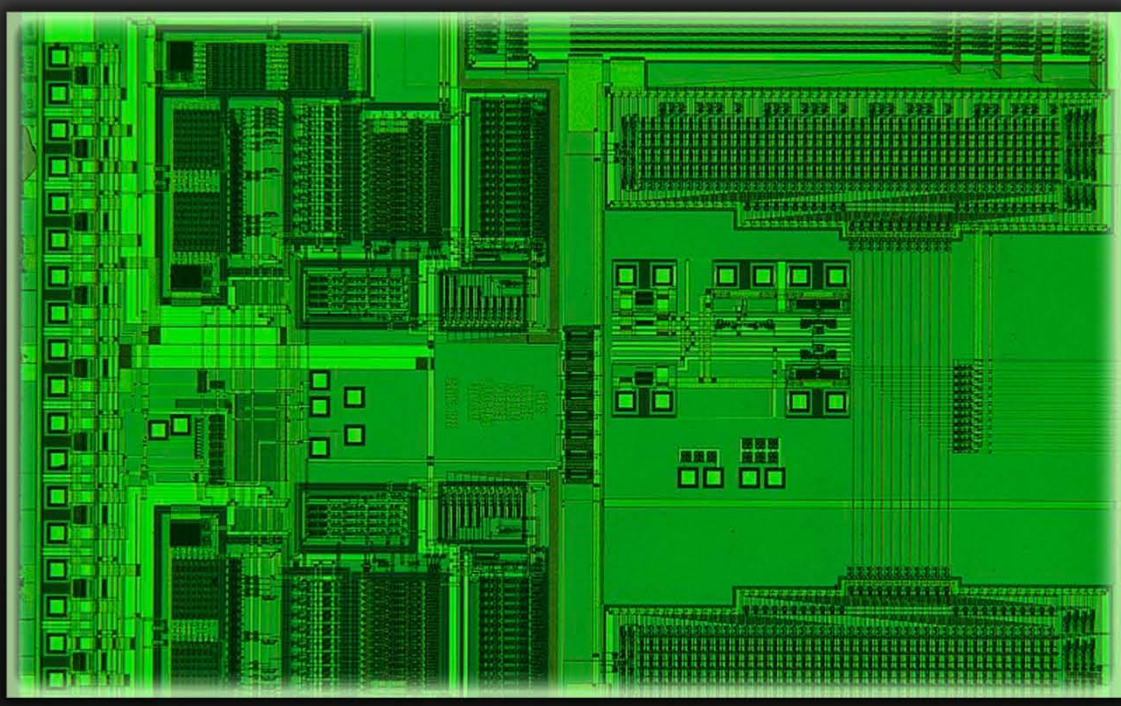
Microprocessors
Laboratory Manual



Bharath P

Microprocessors (8086)

Laboratory Manual

Contents

Execution of 8086 programs.

Run command prompt and go to Masm directory

i.e. C:\masm

Type the program by opening an editor using Edit command

i.e. C:\masm\edit filename.asm

After typing the program assemble the program using masm command.

i.e. C:\masm\masm filename.asm;

After assembling, link the file using link command

i.e. C:\masm\link filename.obj;

Finally use debug or afdebug command to execute the program.

C:\masm\debug filename.exe

- *-t* ; for single step execution
- *-g* ; for at a time execution
- *-I* ; for restarting the program execution
- *-d* ; to see the data segment
- *-q* ; to quit the execution

C:\masm\afdebug filename.exe

- *F1* ; for single step execution
- *g* ; for at a time execution
- *L filename.exe* ; to reload the program
- *Quit* ; to come out of the execute screen

1.a. Byte and word data transfer in different addressing modes

```
.MODEL SMALL
.DATA
        Array DB 5 DUP (0)
        DB 78h
        DB 20 DUP (0)
.CODE
        MOV AX,@DATA
        MOV DS,AX

        MOV BX, offset Array
        MOV SI,05h
        MOV CL,[BX+SI]
        MOV SI,11H
        MOV [BX+SI],CL

        MOV AH,4Ch
        INT 21h
END
```

1.b.i Block transfer without overlap

```
.MODEL SMALL
.DATA
        Array1 DW 1111h,2222h,3333h,4444h,5555h
        Array2 DW 5 DUP (0)
        Count DW 0005H
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA SI,Array1
        LEA DI,Array2
        MOV CX,Count

NEXT:   MOV AX,[SI]
        MOV [DI],AX
        INC SI
        INC SI
        INC DI
        INC DI
        LOOP NEXT

        MOV AH,4Ch
        INT  21h
END
```

1.b.ii Block transfer with overlap

```
.MODEL SMALL
.DATA
        Array DB 11h,22h,33h,44h,55h
        Count DW 0005h
.CODE
        MOV AX,@DATA
        MOV ES,AX
        MOV DS,AX
        LEA SI,Array

        ADD SI,Count
        MOV CX,Count
        DEC SI

        MOV DI,SI
        ADD DI,2h
        STD
        REP MOVSB

        MOV AH,4Ch
        INT  21h
END
```

1.c. Block exchange

```
.MODEL SMALL
.DATA
        Array1 DW 1111h,2222h,3333h,4444h,5555h
        Array2 DW 1010h,2020h,3030h,4040h,5050h
        Count DW 0005h
.CODE
         MOV AX,@DATA
        MOV DS,AX
        LEA SI,Array1
        LEA DI,Array2
        MOV CX,Count

NEXT:   MOV BX,[SI]
         MOV DX,[DI]
         XCHG BX,DX
        MOV [SI],BX
        MOV [DI],DX
        INC SI
        INC SI
        INC DI
        INC DI
        LOOP NEXT

        MOV AH,4Ch
        INT  21h
END
```

2.a.i 16 bit Addition/Subtraction

```
.MODEL TINY
.CODE
        MOV BX,1234h
        MOV CX,7698h
        MOV AL,BL                   ; for subtraction replace with
        ADD AL,CL                   ; SUB AL,CL
        DAA                         ; DAS
        MOV DL,AL
        MOV AL,BL
        ADC AL,CH                   ; SBB AL,CH
        DAA                         ; DAS
        MOV DH,AL

        MOV AH,4Ch
        INT 21h
END
```

2.a.ii 32 bit Addition/Subtraction

```
.MODEL SMALL
.DATA
        Data1 DW 1234h
        Data2 DW 5555h
        Data3 DW 6789h
        Data4 DW 1111h
        Ans1 DW 0000h
        Ans2 DW 0000h
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV BX,Data2            ; for subtraction replace with
        ADD BX,Data4            ; SUB BX,Data4
        MOV Ans1,BX
        MOV CX,Data1
        ADC CX,Data3            ; SBB CX,Data3
        MOV Ans2,CX

        MOV AH,4Ch
        INT 21h
END
```

2.b. ASCII adjust after Addition/Subtraction

```
.MODEL TINY
.CODE
        MOV AH,00h
        MOV AL,39h                  ; for subtraction replace with
        ADD AL,31h                  ; SUB AL,31h
        AAA                         ; AAS
        ADD AX,3030h

        MOV AH,4Ch
        INT 21h
END
```

2.c.i 16 bit Multiplication/Division

```
.MODEL TINY
.CODE
        MOV AX,1234h
        MOV BX,7698h
        ADD AL,31h                  ; for Division replace with
        MUL BX                      ; DIV BX
        MOV AH,4Ch
        INT 21h
END
```

2.c.ii 32 bit Multiplication

```
.MODEL SMALL
.DATA
        Low1 DW 5678h
        High1 DW 1234h
        Low2 DW 5678h
        High2 DW 1234h
        Ans1 DW ?
        Ans2 DW ?
        Ans3 DW ?
        Ans4 DW ?
.CODE
        MOV AX,@Data
        MOV DS,AX
        MOV AX,Low1
        MUL Low2
        MOV Ans1,AX
        MOV Ans2,DX
        MOV AX,Low1
        MUL High2
        ADD AX,Ans2
        ADC DX,00h
        MOV Ans2,AX
        MOV Ans3,DX

        MOV AX,High1
        MUL Low2
        MOV CX,AX
        MOV BX,DX
        MOV AX,High1
        MUL High2
        ADD BX, AX
        ADC DX,00H
        ADD Ans2,CX
        ADC Ans3,BX
        ADC DX,00H
        MOV Ans4,DX
        MOV AH,4Ch
        INT 21h
END
```

2.c.iii 16-bit signed Multiplication

```
.MODEL TINY
.CODE
        MOV AX,-1234h
        MOV CX,-0ABCAh
        IMUL CX
        MOV BX, AX
        MOV AH, 4Ch
        INT 21h
END
```

2.c.iv Signed Division of word by a byte

```
.MODEL TINY
.CODE
        MOV DX,-0ABCh
        MOV AX, 1234h
        MOV BX, 2334h
        IDIV BX
        MOV CX,AX
        MOV AH, 4Ch
        INT 21h
END
```

2.d. ASCII adjust after Multiplication

```
.MODEL TINY
.CODE
        MOV AL,OFh
        MOV BL,04h
        MUL BL
        AAM
        ADD AX,3030h
        MOV AH,4Ch
        INT 21h
END
```

2.e.i Square of a word

```
.MODEL SMALL
. DATA
        Number DW OFFFFh
        Ans DW 2 DUP (?)
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV DX, OOh
        MOV AX, Number
        MUL Number
        MOV Ans, AX
        MOV Ans+2, DX
        MOV AH, 4Ch
        INT 21h
END
```

2.e.ii Cube of a byte

```
.MODEL SMALL
. DATA
        Number DB OFFh
        Ans DW 2 DUP (0)
        .CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV AX, 0000h
        MOV DX, 0000h
        MOV CX, 0000h
        MOV CL, Number
        MOV AL, CL
        MUL CL
        MUL CX
        MOV Ans, AX
        MOV Ans+2, DX
        MOV AH, 4Ch
        INT 21h
END
```

2.e.iii Cube of a word

```
.MODEL SMALL
. DATA
        NUMBER DW 05566h
        CUBE DW 3 DUP (0)
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV DX, 00h
        MOV AX, NUMBER
        MUL NUMBER
        MOV BX, DX
        MOV DX, 00h
        MUL NUMBER
        MOV CUBE, AX
        MOV CUBE+2, DX
        MOV AX, BX
        MUL NUMBER
        ADD AX, CUBE+2
        ADC DX, 00h
        MOV CUBE+2, AX
        MOV CUBE+4, DX
        MOV AH, 4Ch
        INT 21h
END
```

2.e.iv LCM of two numbers

```
.MODEL SMALL
.DATA
        Num1 DW 0005h
        Num2 DW 0002h
        Ans DW ?
.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV AX, Num1
        MOV BX, Num2
        MOV DX, 0000h
NEXT:   PUSH AX
        PUSH DX
        DIV BX
        CMP DX, 0000h
        JZ LAST
        POP DX
        POP AX
        ADD AX, Num1
        JNC NEXT
        INC DX
        JMP NEXT
LAST:   POP Ans+2
        POP Ans

        MOV AH, 4Ch
        INT 21h
END
```

2.e.v HCF of two numbers

```
.MODEL SMALL
.DATA
        Num1 DW 0005h
        Num2 DW 0002h
        Ans DW ?
.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV AX, Num1
        MOV BX, Num2
FIRST:  CMP AX, BX
        JA NEXT
        XCHG AX, BX
NEXT:   MOV DX, 0000h
        DIV BX
        CMP DX, 0000h
        JE LAST
        MOV AX, DX
        JMP FIRST
LAST:   MOV Ans, BX

        MOV AH, 4Ch
        INT 21h
END
```

2.e.vi Factorial of a number

```
.MODEL SMALL
.DATA
        Number DB 08h
        Ans DW 0000h
.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV CH, 00h
        MOV AL, Number
        MOV BL, AL
        MOV CL, AL
        SUB CL, 02h
NEXT:   DEC BL
        MUL BL
        LOOP NEXT
        MOV Ans, AX

        MOV AH, 4Ch
        INT 21h
END
```

2.f.i Binary to BCD conversion

```
.MODEL SMALL
.DATA
        Binary DB 63h
        Ans DB 00h, 00h, 00h
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV AX, 00h
        MOV AL, Binary
        MOV CL, 64h
        DIV CL
        MOV BCD, AL
        MOV AL, AH
        MOV AH, 00h
        MOV CL, 0Ah
        DIV CL

        MOV Ans+ 1, AL
        MOV Ans+2, AH
        OR Ans, 30h
        OR Ans+ l,30h
        OR Ans+2,30h

        MOV AH, 4Ch
        INT 21h
END
```

2.f.ii BCD to Binary conversion

```
.MODEL SMALL
.DATA
        BCD DB 15h
        Ans DB 00h
.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV AL, BCD
        AND AL, 0Fh
        MOV BL, AL
        MOV AL, BCD
        AND AL, 0F0h
        MOV CL, 04h
        ROR AL, CL
        MOV CL, 0Ah
        MUL CL
        ADD AL, BL
        MOV Ans, AL

        MOV AH, 4Ch
        INT 21h
END
```

3.a. Bit manupilation to check if the data is positive or negative

```
.MODEL SMALL
.DATA
        Msg1 DB 'ENTERED NUMBER IS POSITIVE. $'
        Msg2 DB 'ENTERED NUMBER IS NEGATIVE. $'
        Input DB ?
.STACK
.CODE
        MOV AX, @Data
        MOV DS, AX
        MOV AL, Input

        ROL AL, 01h
        JC NEXT

        LEA DX, Msg1
        MOV AH, 09h
        INT 21h
        JMP LAST

NEXT:   LEA DX, Msg2
        MOV AH, 09h
        INT 21h

LAST:    MOV AH, 4Ch
         INT 21h
END
```

3.b. Bit manupilation to check if the data is odd or even

```
.MODEL SMALL
.DATA
        Msg1 DB 'ENTERED NUMBER IS ODD. $'
        Msg2 DB 'ENTERED NUMBER IS EVEN. $'
        Input DB  ?
.STACK
.CODE
        MOV AX, @Data
        MOV DS, AX
        MOV AL, Input

        SAR AL, 01h
        JC NEXT

        LEA DX, Msg2
        MOV AH, 09h
        INT 21h
        JMP LAST

NEXT:   LEA DX, Msg1
        MOV AH, 09h
        INT 21h

LAST:   MOV AH, 4Ch
        INT 21h
END
```

3.c. Bit manupilation to count the number of 1's and 0's in given data

```
.MODEL SMALL
.CODE
        MOV CX, 0008h
        MOV AL, 24h
        MOV BL, 00h
        MOV DL, BL

NEXT:   SAR AL, 01h
        JC DOWN
        INC BL
        LOOP NEXT
        JMP LAST
DOWN: INC DL
        LOOP NEXT

LAST:   MOV AH, 4Ch
        INT 21h
END
```

3.d. Bit wise Palindrome

```
.MODEL SMALL
.DATA
        Msg1 DB 'GIVEN BYTE IS PALINDROME $'
        Msg2 DB 'GIVEN BYTE IS NOT PALINDROME $'
        Input DB ?
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV BL, Input
        MOV CX, 0008h
        MOV DL, BL

UP:     ROL BL, 01h
        RCL DL, 01h
        LOOP UP

        CMP BL, DL
        JZ NEXT

        LEA DX, Msg2
        MOV AH, 09h
        INT 21h
        JMP LAST

NEXT:   LEA DX, Msg1
        MOV AH, 09h
        INT 21h

LAST:   MOV AH, 4Ch
        INT 21h
END
```

3.e. Bit manupilation to check 2 out of 5 code

```
.MODEL SMALL
.DATA
        Num DB 13h
        Dis1 DB ' GIVEN BYTE IS A 2 OUT OF 5 CODE $'
        Dis2 DB 'GIVEN BYTE IS NOT A 2 OUT OF 5 CODE $'

.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV AL, Num
        TEST AL, 0E0h
        JNZ LAST
        MOV CX, 05h
        MOV AH, 00h

REPEAT: ROR AL, 01h
        JNC SKIP
        INC AH
SKIP:   LOOP REPEAT

        CMP AH, 02h
        JNE LAST
        LEA DX, Dis1
        JMP DISP
LAST:   LEA DX, Dis2
DISP:   MOV AH, 09h
        INT 21h

        MOV AH, 4Ch
        INT 21h

END
```

4.i Addition/Subtraction of array of words

```
.MODEL SMALL
.DATA
        Array DW 1234h, 5678h, 9ABCh, 0DEF0h, 0AA11h
        Count DW 0005h
        Result DW 0000h, 0000h
.CODE
        MOV AX,@DATA
        MOV DS, AX

        MOV DX, 00h
        MOV CX, Count
        DEC CX
        LEA SI, Array
        MOV BX, [SI]

NEXT:   INC SI
        INC SI                          ; for subtraction replace with
        ADD BX, [SI]                    ; SUB BX,[SI]
        JC LAST
        LOOP NEXT
LAST:   INC DX
        LOOP NEXT
        MOV Result, BX
        MOV Result+2, DX

        MOV AH, 4Ch
        INT 21h
END
```

4.ii Largest/Smallest element in an array

```
.MODEL SMALL
.DATA
        Array DB 10h, 20h, 30h, 40h, 50h
        Count DW 0005h
        Result DB 00h
.CODE
        MOV AX,@DATA
        MOV DS, AX

        LEA SI, Array
        DEC Count
        MOV CX, Count
        MOV AL, [SI]

NEXT:   INC SI
        CMP AL, [SI]            ; for smallest replace with
        JC DOWN                 ; JNC DOWN
        LOOP NEXT
        JMP LAST
DOWN:   XCHG AL, [SI]
        LOOP NEXT
        MOV Result, AL

LAST:   MOV AH, 4Ch
        INT 21h
END
```

4.iii Sorting array in Ascending/Descending order

```
.MODEL SMALL
.DATA
        Array DW 2233h, 8899h, 6677h, 0011h, 4455h
        Count DW 0005h
.CODE
        MOV AX, @DATA
        MOV DS, AX

        MOV CX, Count
        LEA SI, Array

NEXT:   MOV BX, [SI]
        INC SI
        INC SI
        CMP BX, [SI]                    ; for descending order replace with
        JNC DOWN                        ; JC DOWN
        LOOP NEXT
DOWN: XCHG BX, [SI]
        DEC SI
        DEC SI
        MOV [SI], BX
        LOOP NEXT

        MOV AH, 4Ch
        INT 21h
END
```

5.i String transfer

```
.MODEL SMALL
.DATA
        String1 DB 'BMSCE DEPT OF ECE$'
        Length EQU ($-String1)
        String2 DB LEN DUP (0)
.CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

        MOV CX, Length
        CLD
        LEA SI, String1
        LEA DI, String2
        REP MOVSB

        MOV AH, 4Ch
        INT 21h
END
```

5.ii String reverse

```
.MODEL SMALL
.DATA
        String DB 'BMSCE$'
        Length EQU ($-String)
        Rvrs DB Length DUP (0)
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV ES, AX

        MOV CX, Length
        LEA SI, String+Length-1
        LEA DI, Rvrs

REPEAT: MOV AL, [SI]
        MOV [DI], AL
        DEC SI
        INC DI
        LOOP REPEAT

        MOV AH, 4Ch
        INT 21h
END
```

5.iii Character search in a string

```
.MODEL SMALL
.DATA
        String DB 'BMS COLLEGE'
        Length EQU ($-String)
        Key DB 'X'
        Dis1 DB '-IS PRESENT IN GIVEN STRING$'
        Dis2 DB  '-IS NOT PRESENT IN GIVEN STRING$'
.CODE
        MOV AX,@DATA
        MOV DS, AX
        MOV ES, AX

        MOV DL, Key
        MOV AH, 02h
        INT 21h
        LEA DI, String
        MOV AL, Key
        MOV CX, Length

        REPNE SCASB

        JE PRESENT
        LEA DX, Dis2
        CALL Display
        JMP OVER

PRESENT: LEA DX, Dis1
        CALL Display

OVER:   MOV AH, 4Ch
        INT 21h

Display PROC NEAR
        MOV AH, 09h
        INT 21h
        RET
Display ENDP

END
```

5.iv Palindrome in string

```
.MODEL SMALL
.DATA
                String DB 'BMSCE$'
                Length EQU ($-String)
                Rvrs DB 30 DUP(0)
                Dis1 DB '-IS NOT A PALINDROME$'
                Dis2 DB '-IS A PALINDROME$'

.CODE
                MOV AX,@DATA
                MOV DS, AX
                MOV ES, AX

                MOV CX, Length
                LEA SI, String+Length-1
                LEA DI, Rvrs
REPEAT:         MOV AL, [SI]
                MOV [DI], AL
                DEC SI
                INC DI
                LOOP REPEAT

                LEA DX, String
                CALL Display

                LEA SI, String
                LEA DI, Rvrs
                MOV CX, Length
                REPE CMPSB
                JNZ NO

                LEA DX, Dis2
                CALL Display
                JMP OVER

NO:             LEA DX, Dis1
                CALL Display

OVER:           MOV AH, 4Ch
                INT 21h

Display PROC NEAR
                MOV AH, 09h
                INT 21h
                RET
Display ENDP

END
```

6. Display string on console using DOS interrupts

```
.MODEL SMALL

DATA SEGMENT
        MSG DB 'BMSCE ECE DEPT$'
DATA ENDS

CODE SEGMENT

        MAIN    PROC    FAR
        ASSUME  CS: CODE, DS: DATA

                START:  PUSH DS
                        XOR AX, AX
                        PUSH AX
                        MOV AX, DATA
                        MOV DS, AX
                        MOV AH, 09h
                        MOV DX, OFFSET MSG
                        INT 21h
                        RET
                END START

        MAIN ENDP

CODE ENDS
```

Interfacing the 8086 processor.

Run command prompt and go to Masm directory

i.e. C:\masm

Type the program by opening an editor using Edit command

i.e. C:*masm\edit filename.asm*

After typing the program assemble the program using masm command.

i.e. C:\masm\masm filename.asm;

After assembling, link the file using link command

i.e. C:\masm\ link filename.obj;

Convert the executable file to binary program.

i.e. C:\masm\ exe2bin filename.exe

Convert the binary file to hex program.

i.e. C:\masm\bin2hex filename.exe

Open the hex uploader program.

i.e. C:\masm\mmeterm

Set the Baud rate to 9600bits/second.

i.e. 5. *Configuration>* 1. *Baud Rate>5. 9600*

Press reset on the interface kit.

Message will appear on the LCD as uP 8086.

Press *Download* on the interface kit to prepare the processor to receive file.

Message will appear on the LCD as Reading RS 232.

Send file to the processor via serial port [COM1].

i.e. 3. *Sendfile>Which* File? *filename.hex*

Press Enter

Message will appear on the LCD as Data received.

1. Matrix Keyboard Interfacing

```
CODE SEGMENT
        ASSUME CS:code,DS:code,ES:code,SS:code
CWR EQU 46h
PORTA EQU 40h
PORTB EQU 42h
PORTC EQU 44h

                ORG 0400h
                MOV AL, 88h             ; port a and port c high as output
                OUT CWR,AL              ; port b and port c low as output

READKEY:
                MOV DL,0                ; clear e/dl register
                MOV AL,0F0h             ; output all one's to pc high
                OUT PORTC,AL
LP:
                IN AL,PORTC
                AND AL,0F0h
                CMP AL,0F0h
                JNZ LP
                CALL FAR PTR ONEMS
KEYCHK:
                IN AL,PORTC
                AND AL,0F0h
                CMP AL,0F0h
                JZ KEYCHK               ;wait for key press
                CALL FAR PTR ONEMS
                MOV AL,7Fh
                MOV BH,04h
NXTCOLM:
                ROL AL, 01h             ; scan each column
                MOV DH,AL               ; and wait for the data
                OUT PORTC,AL            ; in any of the four
                IN AL,PORTC             ; rows
                AND AL,0F0h
                MOV CL,04h
NXTROW:
                ROL AL,01h              ; scan each column
                JNC CODEN               ; scan each column
                INC DL                  ; in any of the four
                DEC CL                  ; rows
                JNZ NXTROW
                MOV AL,DH
                DEC BH
                JNZ NXTCOLM
                JMP KEYCHK
```

```
CODEN:
                MOV AL,DL
                MOV DH,0h
                MOV BX,OFFSET LOOKUP+8000h
                ADD BX,DX
                MOV AL,BYTE PTR[BX]
                OUT PORTB,AL
                JMP READKEY

ONEMS:                                          ; delay routine
                PUSH AX
                MOV AL,0FFh
LOP:
                DEC AL
                JNZ LOP
                POP AX
                RETF

LOOKUP:
        DB      00h,04h,08h,0Ch,01h,05h,09h,0Dh
        DB      02h,06h,0Ah,0Eh,03h,07h,0Bh,0Fh
CODE ENDS

END
```

2. LCD Interface

```
CODE SEGMENT
        ASSUME CS:CODE,DS:CODE,ES:CODE
DISINT EQU 21h
DSPBUF EQU 9E00h

        ORG 400h
MES1 DB 'BMSCE'                         ;maximum size of message can be 16 bytes
MES2 DB 'OF E AND C'
MES3 DB ' '

L1:     MOV SI,OFFSET MES3+8000h        ; move result format message
        MOV DI,DSPBUF                   ; to display buffer
        MOV CX,08h                      ; counter for movs instruction
        REP MOVSW                       ; counter for movs instruction
        INT DISINT
        CALL DELAY

        MOV SI,OFFSET MES1+8000h        ; move result format message
        MOV DI,DSPBUF                   ; to display buffer
        MOV CX,08h                      ; counter for movs instruction
        REP MOVSW                       ; move 8 words to display buffer
        INT DISINT
        CALL DELAY

        MOV SI,OFFSET MES2+8000h        ; move result format message
        MOV DI,DSPBUF                   ; to display buffer
        MOV CX,08h                      ; counter for movs instruction
        REP MOVSW                       ; move 8 words to display buffer
        INT DISINT
        CALL DELAY
        JMP L1

DELAY PROC NEAR
        MOV AX,0FF00h
AGAIN: DEC AX
        JNZ AGAIN
        RET
DELAY ENDP
CODE ENDS
END
```

3. Logical Controller

```
CODE SEGMENT
        ASSUME CS:CODE,DS:CODE,SS:CODE,ES:CODE

        ORG 0400h
        MOV AL, 89h
        OUT 46h,AL

AGAIN: IN AL, 44h
        MOV BL, AL
        AND AL, 0F0h
        MOV CL, 04h
        SHR AL,CL
                                        ; replace with
        AND AL,BL                       ; XOR AL,BL    for XOR operation
                                        ; OR AL,BL     for OR operation
        AND AL, 0Fh
        OUT 42h,AL
        JMP AGAIN

CODE ENDS
END
```

4. Stepper Motor Interface

```
CODE SEGMENT
        ASSUME CS:CODE,DS:CODE
        CWR EQU 46h
        PORTA EQU 40h

        ORG 400h
                MOV AL,80h
                OUTCWR,AL
                                        ; send 09h,0Ah,06h,05h for
                                        ; anti-clockwise rotation

                                        ; send 05h,06h,0Ah,09h for
                                        ; clockwise rotation

        AGAIN:  MOV AL,05h
                OUT PORTA,AL
                CALL DELAY

                MOV AL,06h
                OUT PORTA,AL
                CALL DELAY

                MOV AL,0Ah
                OUT PORTA,AL
                CALL DELAY

                MOV AL,09h
                OUT PORTA,AL
                CALL DELAY

                JMPAGAIN

        DELAY PROC NEAR
                MOV CX, 0800h
        BACK:   DEC CX
                JNZ BACK
                RET
        DELAY ENDP

CODE ENDS
END
```

DsynFLO
dsynflo@in.com
India