B.M.S.COLLEGE OF ENGINEERING, BANGALORE-19

(Autonomous College under VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MICROPROCESSORS AND MICROCONTROLLERS LABORATORY MANUAL

PROGRAM: BACHELOR OF ENGINEERING

SEMESTER: III

SESSION: 2019-20

COURSECODE: 19CS3ESMMC

COURSE TITLE: MICROPROCESSORS AND MICROCONTROLLERS

CREDITS: 3-0-1

FACULTY: JYOTHI S NAYAK, RAJESHWARI B S, SHYAMALA G

PREFACE

This laboratory manual is prepared by the Department of Computer Science and Engineering for Microprocessors and Microcontrollers. This lab manual can be used as instructional book for students, staff and instructors to assist in performing and understanding the experiments. In this manual, experiments are as per syllabus.

MICROPROCESSORS AND MICROCONTROLLERS

Course Outcomes

Ability to **Apply** the knowledge of Architecture, Instruction Set and Assembly Language Programming of Microprocessor and Microcontroller.
Ability to **Analyze** the attributes of Microprocessors & Microcontrollers to address the given problem
Ability to **Design** Microprocessor and Microcontroller based systems
Ability to **Conduct** experiments usingAssembly Language Programming to demonstratethe features of Microprocessor and Microcontroller.

MICROPROCESSORS AND MICROCONTROLLERS: Syllabus

Unit No.	Topics	Hrs	Text book No. from which Unit topics are being covered
1	Introduction to 8086 Microprocessor, Internal Architecture, Register Organisation, Flag register, Addressing Modes, Assembler directives, Instruction set of 8086 – Data Transfer instructions, Logical instructions, Arithmetic instructions, Example programs, Branch instructions, Loop instructions.	8 Hrs	Text Book-1: 1.1, 1.2, 2.2, 2.4, 2.3
2	Machine control instructions, Flag manipulation instructions, Shift and rotate instructions, Delay Loops, String instructions, Assembly language programming examples Instruction Templates, MOV instruction Coding Format and Examples, Special Architectural Features and related programming: Stacks, Procedures, Macros, Interrupts and the Vector Table.	7 Hrs	2.3, 3.2, 4.1-4.7
3	Pin Diagram of 8086, Maximum/ Minimum Mode of 8086, Timing Diagram, Methods of interfacing I/O devices, Programmable Peripheral interface 8255, Interfacing of Logic controller, Interfacing of Seven segment display	8 Hrs	1.8, 1.9, 5.3, 5.4, 5.5
4	Microprocessors versus Microcontrollers, 8051 Architecture: Introduction, 8051 Microcontroller Hardware, Input/ Output	9 Hrs	Text Book-2: Page No. 2 to 4,

	Pins, External Memory Interface, Addressing Modes and Instruction set. Example Demonstration using 8051 instruction set, Data transfer instructions, Arithmetic instructions.		11-28, 45-54, 71-82,
5	8051 instruction set: Logical instructions, Branching and Subroutines, Example programs. Interfacing with 8051: LCD Interfacing, Keyboard Interfacing, Seven segment display, Stepper Motor, Elevator	7 Hrs	Text Book-2: Page No. 59-68, 86-95. Text Book-3: 12.1 and 12.2, 17.2

MICROPROCESSORS AND MICROCONTROLLERS: Lab Experiments

Lab Program	Program Details
	Software Programs using 8086
1	Design and develop an assembly language program to search a key element “X” in a list of ‘n’ 16-bit numbers. Adopt Binary search algorithm in your program for searching.
2	Design and develop an assembly program to sort a given set of ‘n’ 16-bit numbers in ascending order. Adopt Bubble Sort algorithm to sort given elements.
3	Read an alphanumeric character and display its equivalent ASCII code at the center of the screen.
4	Reverse a given string and check whether it is a palindrome or not.
5	Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriate messages. Also display the length of the stored strings.
6	Develop an assembly language program to compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers.
7	Read the current time from the system and display it in the standard format on the screen.
8	Write a program to simulate a Decimal Up-counter to display 00-99.
9	Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.
10	Write a program to create a file (input file) and to delete an existing file.
	Hardware Programs Using 8051

11	Drive a Stepper Motor interface to rotate the motor in Anti- clockwise) by N steps. Introduce suitable delay between successive steps.
12	Drive a Stepper Motor interface to rotate the motor in clockwise by N steps. Introduce suitable delay between successive steps.
13	Display messages FIRE and HELP alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages.
14	Display messages BANGALORE in rolling fasion on a 7-segment display interface for a suitable period of time.
15	Program to demo the elevator interface.

Software Programs using 8086

1. *Design and develop an assembly language program to search a key element "X" in a list of 'n' 16-bit numbers. Adopt Binary search algorithm in your program for searching.*

```
.MODEL SMALL
; MACRO TO DISPLAY THE MESSAGE....
DISPLAY MACRO MSG
    LEA DX, MSG
    MOV AH, 09H
    INT 21H
ENDM

.DATA
```

```
LIST DB 01H, 05H, 07H, 10H, 12H, 14H
NUMBER EQU ($-LIST)
KEY DB 011H
MSG1 DB 0DH, 0AH, "ELEMENT FOUND IN THE LIST...$"
MSG2 DB 0DH, 0AH, "SEARCH FAILED!! ELEMENT NOT FOUND IN THE LIST $"

.CODE
START : MOV AX, @DATA
        MOV DS, AX
        MOV CH, NUMBER-1   ; HIGH VALUE...
        MOV CL, 00H        ; LOW VALUE...
AGAIN:  MOV SI, OFFSET LIST
        XOR AX, AX
        CMP CL, CH
        JE NEXT
        JNC FAILED
NEXT:   MOV AL, CL
        ADD AL, CH
        SHR AL, 01H             ; DIVIDE BY 2
        MOV BL, AL
        XOR AH, AH              ; CLEAR AH
        MOV BP, AX
        MOV AL, DS:[BP][SI]
        CMP AL, KEY             ; COMPARE KEY AND A[I]
        JE SUCCESS              ; IF EQUAL, DISPLAY SUCCESS MESSAGE
        JC INCLOW
        MOV CH, BL              ; IF KEY>A[I]  SHIFT HIGH
        DEC CH
        JMP AGAIN
```

```
INCLOW: MOV CL, BL            ; IF KEY<A[I]  SHIFT LOW
        INC CL
        JMP AGAIN
SUCCESS:DISPLAY MSG1
        JMP FINAL
FAILED: DISPLAY MSG2          ; JOB OVER. TERMINATE....
FINAL: MOV AH, 4CH
        INT 21H
END START
```

2. *Design and develop an assembly program to sort a given set of ‘n’ 16-bit numbers in ascending order. Adopt Bubble Sort algorithm to sort given elements.*

```
.MODEL SMALL
DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H
ENDM
.DATA
LIST DB 02H, 01H, 34H, 0F4H, 09H, 05H
NUMBER EQU $-LIST
MSG1 DB 0DH, 0AH, "1 >> SORT IN ASCENDING ORDER$"
MSG2 DB 0DH, 0AH, "2 >> SORT IN DESCENDING ORDER$"
MSG3 DB 0DH, 0AH, "3 >> EXIT$"
MSG4 DB 0DH, 0AH, "ENTER YOUR CHOICE  :: $"
MSG5 DB 0DH, 0AH, "INVALID CHOICE ENTERED...$"
.CODE
START : MOV AX, @DATA
```

```
      MOV DS, AX
      LEA SI, LIST
      MOV CH, NUMBER-1        ; CL STORES THE NUMBER OF ELEMENTS IN LIST
      DISPLAY MSG1            ; DISPLAY THE MENU...
      DISPLAY MSG2
      DISPLAY MSG3
      DISPLAY MSG4
      MOV AH, 01H
      INT 21H
      SUB AL, 30H
      CMP AL, 01H             ; INPUT=1? SORT IN ASCENDING ORDER
      JE ASCSORT
      CMP AL, 02H             ; INPUT=2? SORT IN DESCENDING ORDER
      JE DESSORT
      CMP AL, 03H             ; INPUT=3? EXIT
      JE FINAL
      DISPLAY MSG5
      JMP FINAL
ASCSORT:MOV BL, 00H
AGAIN:  MOV SI, OFFSET LIST
      MOV CL, 00H             ; J VALUE
      MOV BH, CH
      SUB BH, BL              ; N-1-i
NPASS:  CMP CL, BH
      JNC NEXT
      MOV AL, [SI]
      MOV BP, 01H
      CMP AL, DS: [BP][SI]
      JC _NOPE
```

```
        XCHG AL, [SI+1]
        XCHG [SI], AL
_NOPE : INC CL
        INC SI
        JMP NPASS
NEXT:   INC BL
        CMP BL, CH
        JC AGAIN
        JMP FINAL

DESSORT:MOV BL, 00H
AGAIN1: MOV SI, OFFSET LIST
        MOV CL, 00H             ; J VALUE
        MOV BH, CH
        SUB BH, BL              ; N-1-i
NPASS1: CMP CL, BH
        JNC NEXT
        MOV AL, [SI]
        MOV BP, 01H
        CMP AL, DS: [BP][SI]
        JNC _NOPE1
        XCHG AL, [SI+1]
        XCHG [SI], AL
_NOPE1: INC CL
        INC SI
        JMP NPASS1
NEXT1:  INC BL
        CMP BL, CH
        JC AGAIN1
```

```
FINAL: MOV AH, 4CH
       INT 21H
END START
```

3. *Read an alphanumeric character and display its equivalent ASCII code at the center of the screen.*

```
.MODEL SMALL
DISPLAY MACRO MSG
       LEA DX, MSG
       MOV AH, 09H
       INT 21H
ENDM

; MACRO TO DISPLAY A CHARACTER.
DISPCHAR MACRO
       MOV AH, 02H
       INT 21H
ENDM

.DATA
MSG1 DB 0DH, 0AH, "ENTER AN ALPHANUMERIC CHARACTER: $"
MSG2 DB 0DH, 0AH, "NOT AN ALPHANUMERIC CHARACTER...$"


.CODE
START: MOV AX, @DATA
       MOV DS, AX
       DISPLAY MSG1
```

```
MOV AH, 01H
INT 21H
CALL CHECK          ; CHECK FOR ALPHANUMERIC CHARACTER...
JC ERROR
PUSH AX
; SET MODE AND CLEAR THE SCREEN
; ROW =25 AND COLUMN = 80
MOV AH, 00H
MOV AL, 03H
INT 10H
; MOVE THE CURSOR TO THE MID POINT OF SCREEN
MOV AH, 02H
MOV BH, 00H          ; PAGE NUMBER
MOV DH, 12D          ; ROW VALUE
MOV DL, 40D          ; COLUMN VALUE
INT 10H
POP AX              ; RESTORE THE CHARACTER.
AAM
PUSH AX
MOV AL, AH
XOR AH, AH
AAM
ADD AX, 3030H
MOV DL, AH
PUSH AX
DISPCHAR            ; DISPLAY THE ASCII VALUE
POP AX
MOV DL, AL
DISPCHAR
```

```
        POP AX
        ADD AL, 30H
        MOV DL, AL
        DISPCHAR
        ; WAIT FOR USER TO PRESS ANY KEY
        MOV AH, 07H
        INT 21H
        ; FINISH ...JOB OVER
        JMP FINAL
ERROR: DISPLAY MSG2
        JMP FINAL

; THIS PROCEDURE CHECKS WHETHER THE INPUT IS ALPHANUMERIC OR NOT
CHECK PROC NEAR
        CMP AL, 30H
        JE FRET
        JC ERR
        CMP AL, 39H
        JE FRET
        JNC NEXT
        JC FRET
NEXT:  CMP AL, 41H
        JE FRET
        JC ERR
        CMP AL, 5AH
        JE FRET
        JNC NEXT1
        JC FRET
NEXT1: CMP AL, 61H
```

```
        JE FRET
        JC ERR
        CMP AL, 7AH
        JE FRET
        JNC ERR
        JC FRET
ERR: STC              ; SET CARRY FOR ERROR
RET
FRET: CLC
RET
CHECK ENDP
; PROCEDURE ENDS HERE

FINAL: MOV AH, 4CH
        INT 21H
END START
```

4. *Reverse a given string and check whether it is a palindrome or not.*

```
.MODEL SMALL
DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H
ENDM

.DATA
MSG1 DB 0DH, 0AH, "ENTER STRING: $"
MSG2 DB 0DH, 0AH, "REVERSE STRING: $"
MSG3 DB 0DH, 0AH, "INPUT STRING IS PALINDROME. $"
```

```
MSG4 DB 0DH, 0AH, "INPUT STRING IS NOT A PALINDROME STRING. $"
STRING DB 80H DUP(?)
RSTRING DB 80H DUP(?)
.CODE
START: MOV AX, @DATA
       MOV DS, AX
       DISPLAY MSG1
       ; TAKE THE STRING FROM KEYBOARD CHARACTER BY CHARACTER
       MOV SI, OFFSET STRING
       XOR CL, CL
AGAIN:  MOV AH, 01H
       INT 21H
       CMP AL, 0DH
       JE NEXT
       MOV [SI], AL
       INC SI
       INC CL
       JMP AGAIN
NEXT:  MOV [SI], BYTE PTR '$'
       ; STRING INPUT OVER....
       DEC SI
       MOV CH, CL
       ; REVERSE THE STRING AND STORE IN RSTRING
       MOV DI, OFFSET RSTRING
BACK:   MOV AL, [SI]
       MOV [DI], AL
       DEC SI
       INC DI
       DEC CH
```

```
        JNZ BACK
        MOV [DI], BYTE PTR '$'
        DISPLAY MSG2
        DISPLAY RSTRING
        MOV SI,OFFSET STRING
        MOV DI, OFFSET RSTRING
AG:     MOV AL, [SI]
        CMP AL, [DI]
        JNE FAIL
        INC SI
        INC DI
        DEC CX
        JZ SUCCESS
        JMP AG
FAIL:   DISPLAY MSG4
        JMP FINAL
SUCCESS: DISPLAY MSG3
FINAL:  MOV AH, 4CH
        INT 21H
END
```

5. *Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriate messages. Also display the length of the stored strings.*

```
.MODEL SMALL

DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H
ENDM
```

```
.DATA
MSG1 DB 0DH, 0AH, "ENTER FIRST STRING     : $"
MSG2 DB 0DH, 0AH, "ENTER SECOND STRING     : $"
MSG3 DB 0DH, 0AH, "LENGTH OF FIRST STRING: $"
MSG4 DB 0DH, 0AH, "LENGTH OF SECOND STRING: $"
MSG5 DB 0DH, 0AH, "---STRINGS ARE EQUAL---$"
MSG6 DB 0DH, 0AH, "---STRINGS ARE NOT EQUAL---$"
STRING1 DB 80H DUP(?)
STRING2 DB 80H DUP(?)

.CODE
START: MOV AX, @DATA
       MOV DS, AX
       DISPLAY MSG1
       MOV SI, OFFSET STRING1
       CALL READSTR
       MOV BL, CL             ; STORE THE LENGTH OF FIRST STRING
       DISPLAY MSG2
       MOV SI, OFFSET STRING2
       CALL READSTR
       PUSH BX
       PUSH CX
       DISPLAY MSG3
       MOV AL, BL
       CALL LEN_DIS
       DISPLAY MSG4
       MOV AL, CL
       CALL LEN_DIS
```

```
        POP CX
        POP BX
        CMP CL, BL              ; COMPARE THE LENGTHS
        JNE FAIL                ; IF LENGTHS ARE EQUAL, PROCESS NEXT STATMENT
        MOV SI, OFFSET STRING1
        MOV DI, OFFSET STRING2
        CLD
CHK:    MOV AL, [SI]            ; COMPARE BOTH THE STRING
        CMP AL, [DI]
        JNE FAIL
        INC SI
        INC DI
        DEC CL
        JNZ CHK
        DISPLAY MSG5
        JMP FINAL

LEN_DIS PROC NEAR
        XOR AH, AH
        ADD AL, 00H
        AAM
        ADD AX, 3030H
        MOV BH, AL
        MOV DL, AH
        MOV AH, 02H
        INT 21H
        MOV DL, BH
        MOV AH, 02H
        INT 21H
```

```
RET
LEN_DIS ENDP
READSTR PROC NEAR
        XOR CL, CL
BACK:   MOV AH, 01H
        INT 21H
        CMP AL, 0DH
        JE FINISH
        MOV [SI], AL
        INC SI
        INC CL
        JMP BACK
FINISH: MOV [SI], BYTE PTR '$'
        RET
READSTR ENDP
FAIL:   DISPLAY MSG6
FINAL:  MOV AH, 4CH
        INT 21H
END START
```

6. *Develop an assembly language program to compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers*

```
N DB 6; AIM IS TO FIND -> 6C3
 R DB 3
 ANSWER DB 0
 .CODE
 INITDS
 MOV AL,N
 MOV BL,R

 CALL NCR ; CALL NCR PROCEDURE

 MOV AL,ANSWER ; COPY THAT ANSWER TO YOUR AL
 AAM ; SPLIT AL INTO AL & AH
 ADD AX,3030H ; CONVERT INTO ASCII
 MOV BX,AX ; TAKE A COPY TO BE SAFE
 PUTCHAR BH ; DISPLAY 1ST DIGIT
 PUTCHAR BL ; DISPLAY 2ND DIGIT
 EXIT
NCR PROC
CMP BL,0 ; N
C0 = 1
 JNE GO1
 ADD ANSWER,1
 RET

 GO1: CMP BL,AL ; N
CN = 1
 JNE GO2
 ADD ANSWER,1
 RET

 GO2: CMP BL,1 ; N
C1 = N
 JNE GO3
 ADD ANSWER,AL
 RET

 GO3: DEC AL ; N
CN-1= N
 CMP BL,AL
 JNE GO4
 INC AL
 ADD ANSWER,AL
 RET

 GO4: PUSH AX
 PUSH BX ; N-1
 CALL NCR ; C
 POP BX ; R
 POP AX

 DEC BX
 PUSH AX ; N-1
 PUSH BX ; C
 CALL NCR ; R-1
 POP BX
 POP AX
 RET
NCR ENDP
END
```

7. *Read the current time from the system and display it in the standard format on the screen*

```
.MODEL SMALL
```

```
.CODE
  MOV AH,2CH
  INT 21H

  MOV AL,CH
  AAM
  MOV BX,AX
  CALL DISP
  MOV DL,20H
  MOV AH,02H
  INT 21H

 MOV AL,CL
  AAM
  MOV BX,AX
  CALL DISP
  MOV DL,20H
  MOV AH,02H
  INT 21H

  MOV AL,DH
  AAM
  MOV BX,AX
  CALL DISP

  MOV AH,4CH
  INT 21H
  DISP PROC NEAR
  MOV DL,BH
  ADD DL,30H
  MOV AH,02H
  INT 21H
  MOV DL,BL
  ADD DL,30H
  INT 21H
  RET
  DISP ENDP
  END
```

8. Write a program to simulate a Decimal Up-counter to display 00-99.

```
.MODEL SMALL
.CODE
  MOV CL,00

  MOV AH,00H
  MOV AL,03H
  INT 10H
BACK: MOV BH,00H
    MOV DH,00H
    MOV DL,00H
    MOV AH,02H
    INT 10H
    MOV AL,CL
    ADD AL,00H
    AAM
    ADD AX,3030H
    MOV CH,AL
    MOV DL,AH
    MOV AH,02H
    INT 21H
    MOV DL,CH
    MOV AH,02H
    INT 21H
    CALL DELAY
INC CL
    XOR AX,AX
    CMP CL,100D
JNE BACK
    JE LAST
    DELAY PROC NEAR
      PUSH AX
      PUSH BX
      PUSH CX
MOV CX,0FFFH
     AG:MOV BX,0FFFFH
     AG1:NOP
       DEC BX
      JNZ AG1
      DEC CX
      JNZ AG
      POP CX
POP BX
      POP AX
      RET
      DELAY ENDP
   LAST:MOV AH,4CH
   INT 21H
```

```
        END
```

9. *Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.*

```
.MODEL SMALL
DISP MACRO MSG
  LEA DX,MSG
  MOV AH,09H
  INT 21H
ENDM
.DATA
ROW DB 02 DUP(0)
COL DB 02 DUP(0)
MSG1 DB 0DH,0AH,"ENTER X-CO-ORDINATE: $"
MSG2 DB 0DH,0AH,"ENTER Y-CO-ORDINATE: $"
.CODE
    MOV AX,@DATA
    MOV DS,AX
    DISP MSG1
    MOV SI,OFFSET ROW
    CALL READ
    DISP MSG2
    MOV SI,OFFSET COL
    CALL READ
    MOV SI,OFFSET ROW
MOV AH,[SI]
    INC SI
    MOV AL,[SI]
SUB AX,3030H
    AAD
    MOV DH,AL
MOV SI,OFFSET COL
    MOV AH,[SI]
    INC SI
    MOV AL,[SI]
    SUB AX,3030H
AAD
    MOV DL,AL
    MOV AH,00
    MOV AL,03H
    INT 10H
    MOV AH,02H
    INT 10H
    JMP FINAL
 READ PROC NEAR
     MOV CX,02H
BACK:MOV AH,01H
     INT 21H
MOV [SI],AL
     INC SI
     DEC CX
JNZ BACK
RET
READ ENDP
FINAL:MOV AH,01H
INT 21H
MOV AH,4CH
INT 21H
END
```

10. *Write a program to create a file (input file) and to delete an existing file*

```
.MODEL SMALL
DISP MACRO MSG
  LEA DX,MSG
  MOV AH,09H
  INT 21H
ENDM
.DATA
 MSG1 DB 0DH,0AH, "ENTER THE FILE NAME FOR CREATION:-$"
MSG2 DB 0DH,0AH,"FILE CREATED SUCCESSFULLY$"
  MSG3 DB 0DH,0AH,"CREATION FAILED$"
  MSG4 DB 0DH,0AH,"ENTER THE FILE NAME FOR DELETION:-$"
  MSG5 DB 0DH,0AH,"FILE DELETED SUCCESSFULLY$"
  MSG6 DB 0DH,0AH,"DELETION FAILED$"
  FNAME1 DB 10 DUP(0)
  FNAME2 DB 10 DUP(0)
.CODE
   MOV AX,@DATA
   MOV DS,AX
   DISP MSG1
   MOV SI,00
BACK1:MOV AH,01H
INT 21H
   CMP AL,0DH
   JE NEXT1
   MOV FNAME1[SI],AL
INC SI
   JMP BACK1
```

```
NEXT1:MOV FNAME1[SI],'$'
   LEA DX,FNAME1
   MOV CX,00
   MOV AH,3CH
   INT 21H
   JC CFAIL
   DISP MSG2
   JMP DEL
CFAIL:DISP MSG3
DEL:DISP MSG4
    MOV SI,00
 BACK2:MOV AH,01H
INT 21H
    CMP AL,0DH
    JE NEXT2
    MOV FNAME2[SI],AL
INC SI
    JMP BACK2
NEXT2:MOV FNAME2[SI],'$'
     LEA DX,FNAME2
     MOV AH,41H
     INT 21H
     JC DFAIL
     DISP MSG5
     JMP LAST
DFAIL: DISP MSG6
LAST:MOV AH,4CH
     INT 21H END
```

Hardware Programs Using 8051

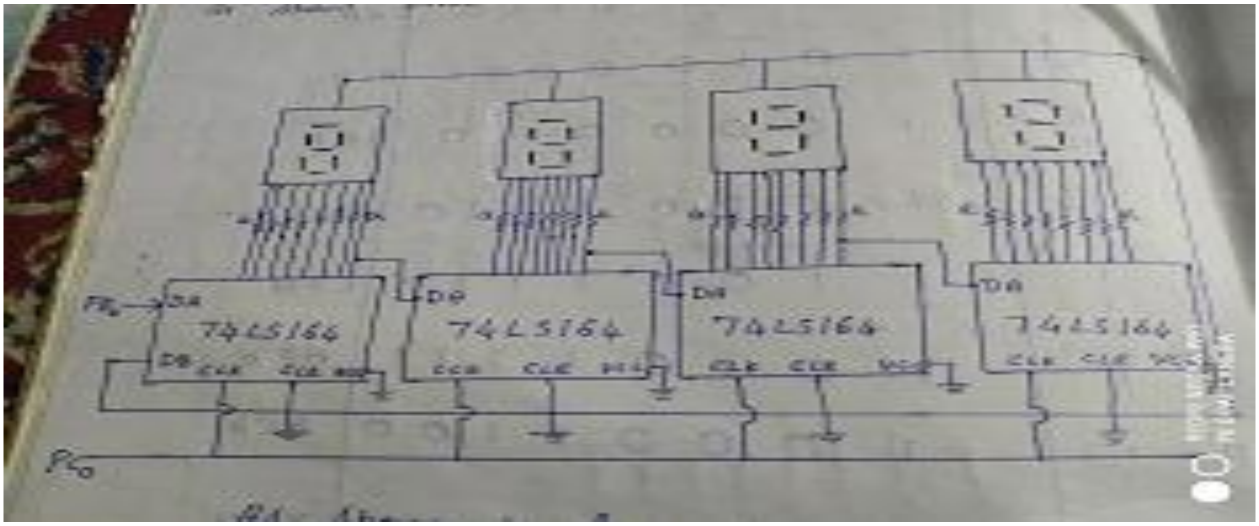
11. Drive a Stepper Motor interface to rotate the motor in Anti- clockwise) by N steps. Introduce suitable delay between successive steps

```
#include<stdio.h>
#include<reg51.h>
char xdata port _at_ 0xe803;
char xdata porta _at_ 0xe800;
charidataacc _at_ 0x30;
delay()    //DELAY BETWEEN THE ROTATION OF THE STEPPER MOTOR
{
int j;
for(j = 0;j < 800; j++)
  {}
}
void main()
{
port = 0x80;  //CONFIGURE ALL THE PORTS OF 8255 AS OUTPUT PORT
while(1)
  {
acc = 0x11;
porta = acc;
delay();
acc = 0x22;
porta = acc;
delay();
acc = 0x44;
porta = acc;
delay();
acc = 0x88
porta = acc;
delay();
  }
}
```

12. *Drive a Stepper Motor interface to rotate the motor in clockwise by N steps. Introduce suitable delay between successive steps*

```
#include<stdio.h>
#include<reg51.h>
charxdata port _at_ 0xe803;
charxdata porta _at_ 0xe800;
charidataacc _at_ 0x30;
delay()    //DELAY BETWEEN THE ROTATION OF THE STEPPER MOTOR
{
int j;
for(j = 0;j < 800; j++)
  {}
}
void main()
{
port = 0x80;  //CONFIGURE ALL THE PORTS OF 8255 AS OUTPUT PORT
while(1)
  {
acc = 0x88;
porta = acc;
delay();
acc = 0x44;
porta = acc;
delay();
acc = 0x22;
porta = acc;
delay();
acc = 0x11
porta = acc;
delay();
  }
}
```

13. *Display messages FIRE and HELP alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages*



```
#include<stdio.h>
#include<reg51.h>
```

```
charxdataCommW _at_ 0xe803;
charxdataportB _at_ 0xe801;
charxdataportC _at_ 0xe802;
char port[20] = {0x8e,0xf9,0xde,0x86,0xff,0xff,0xff,0xff,0x89,0x86,0xc7,0x8c},i;
delay()
{
long u;
        for(u=0;u<8000;u++);
}
void main()
{
        intd,b,j,m;
        unsigned char k;
        CommW = 0x80;
        do
        {
        i=0;
        for(d=0;d<3;d++)
        {
                for(b=0;b<4;b++)
                {
                k = port[i++];
                for(j=0;j<8;j++)
                {
                        m=k;
                        k=k&0x80;
                        {
                        if(k==00)
```

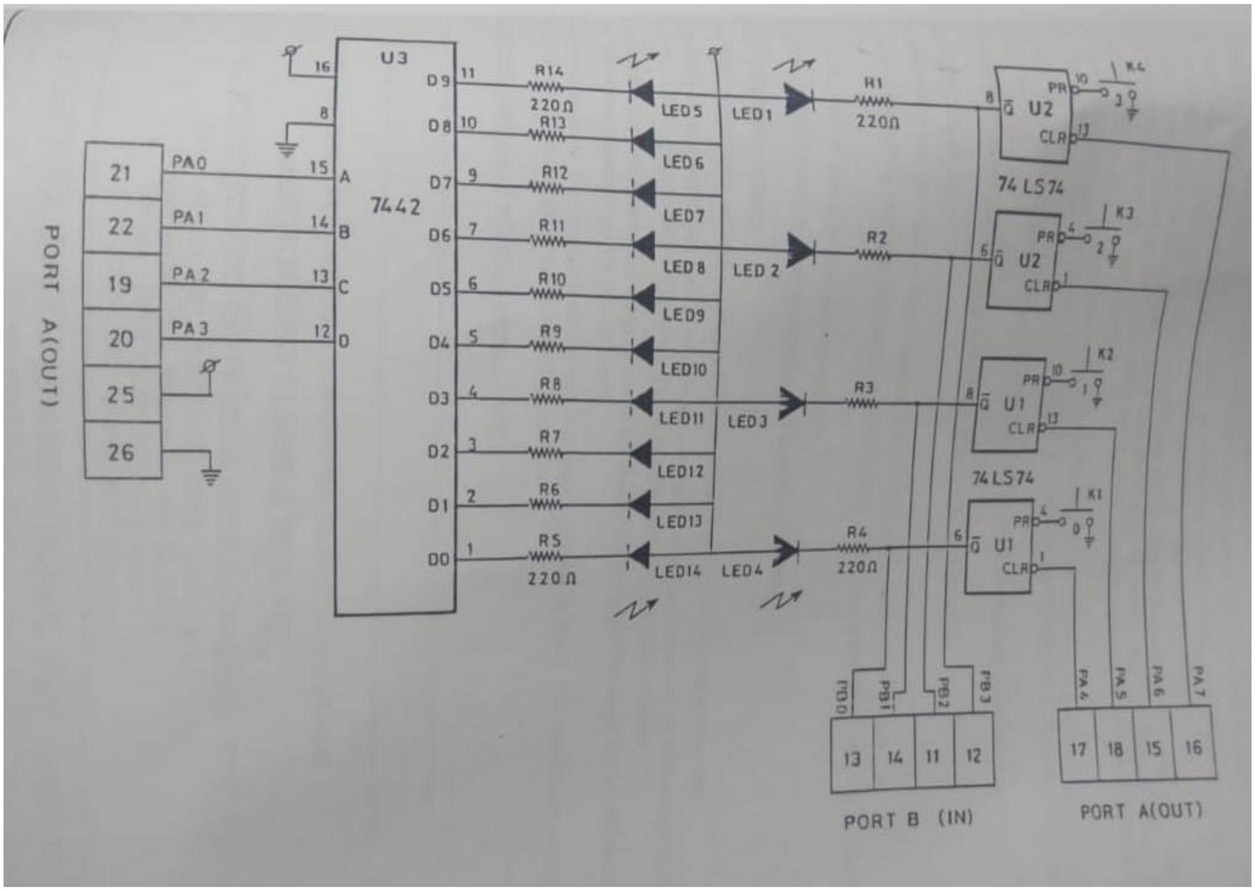
```
            portB=0x00;
            else
            portB=0x01;
            }
            portC = 0x01;
            portC = 0x00;
            k=m;
            k<<=1;
        }
    }
delay();
}
}
while(1);
}
```

14. Display messages BANGALORE in rolling fasion on a 7-segment display interface for a suitable period of time.

```
#include<stdio.h>
#include<reg51.h>
charxdataCommW _at_ 0xe803;
charxdataportB _at_ 0xe801;
charxdataportC _at_ 0xe802;
char port[20] = { 0xff, 0xff, 0xff, 0xff ,0x83,0x88,0xC8,0x82, 0x88,0xC7,0xC0,0xAF,0x86},i;
delay()
{
long u;
for(u=0;u<4000;u++);
}
void main()
```

```
{
intd,b,j,m;
unsigned char k;
CommW = 0x80;
do
{
i=0;
for(d=0;d<1;d++)
{
for(b=13;b>0;b--)
{
delay();
k = port[i++];
for(j=0;j<8;j++)
{
m=k;
k=k&0x80;
{
if(k==00)
portB=0x00;
else
portB=0x01;
}
portC = 0x01;
portC = 0x00;
k=m;
k<<=1;
}
}
delay();
}
}
while(1);
}
```

15. Program to demo the elevator interface



```
#include<stdio.h>
#include<reg51.h>

unsigned char xdataCommandWord  _at_  0xe803;
unsigned char xdataPortA  _at_  0xe800;
unsigned char xdataPortB  _at_  0xe801;
unsigned char xdataPresentFloor,RequestedFloor,Step = 0xf0;
unsigned long xdataCount,i;

Delay()
{
for(Count = 0; Count <= 4500; Count++);
}

Reset()
{
        Step = Step & 0x0f;
        PortA = Step;
        Step = Step | 0xf0;
        PortA = Step;
}
GoUp()
{
switch(RequestedFloor)
 {
        case 0x0d:      while(Step < 0xf3)
                                {
                                  Step++;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;

        case 0x0b:  while(Step < 0xf6)
```

```
                                {
                                  Step++;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;



        case 0x07:  while(Step < 0xf9)
                                {
                                  Step++;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;
 }
}

GoDown()
{
switch(RequestedFloor)
 {
        case 0x0d:      while(Step > 0xf3)
                                {
                                  Step--;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;

        case 0x0b:  while(Step > 0xf6)
                                {
                                  Step--;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;



        case 0x0e:  while(Step > 0xf0)
                                {
                                  Step--;
                                PortA = Step;
                                Delay();
                                }
                                Reset();
                                break;
  }
}



void main()
{
CommandWord = 0x82;
```

```
PortA = 0xf0;
PresentFloor = 0x0e;
while(1){
RequestedFloor = PortB;
RequestedFloor = RequestedFloor& 0x0f;

if(RequestedFloor != 0x0f &&RequestedFloor != PresentFloor){
        if(RequestedFloor<PresentFloor)
                GoUp();
        else
                GoDown();
        PresentFloor = RequestedFloor;
        }
        RequestedFloor = PortB;
  }
```