## PART A

1. Search a key element in a list of 'n' 16-bit numbers using the Binary search algorithm

```
DATA SEGMENT
        A DW 1234H, 2345H, 5678H, 6347H, 7556H
        LEN DB $-A
        KEY DW 0001H
        MID DB ?
        MSG1 DB "KEY FOUND AT $"
        MSG2 DB "KEY NOT FOUND $"
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    SHR LEN, 01H
    DEC LEN
    MOV DL, 00H   ; LOW
    MOV DH, LEN   ; HIGH
    MOV BX, 0000H
    MOV AX, KEY
  UP:  CMP DL, DH
    JG NOTFOUND          ; IF LOW > HIGH; USE JG ONLY SINCE VALUE MAY GET IS -VE
    MOV BL, DL
    ADD BL, DH
    SHR BL, 01H          ;MID=(LOW+HIGH)/2
    MOV MID, BL
    MOV AL, 02H
    MUL BL
    MOV BX, AX
    MOV AX, KEY
    CMP AX, A[BX]
    JZ FOUND
    JG SECONDHALF
    DEC MID              ; IF KEY < A[MID] , HIGH = MID-1
    MOV DH, MID
    JMP UP
SECONDHALF:
    INC MID
    MOV DL, MID           ; IF KEY>A[MID], LOW=MID+1
    JMP UP
FOUND: LEA DX, MSG1
    MOV AH, 09H
    INT 21H
    INC MID
    MOV DL, MID
    AND DL, 0F0H         ; DISPLAY 1ST DIGIT
    MOV CL, 04H
    SHR DL, CL
    CMP DL, 09H
    JBE L3
    ADD DL, 07H
  L3: ADD DL, 30H
    MOV AH, 02H
    INT 21H
```

```
        MOV DL, MID
        AND DL, 0FH          ; DISPLAY 2ND DIGIT
        CMP DL, 09H
        JBE L4
        ADD DL, 07H
     L4: ADD DL, 30H
        MOV AH, 02H
        INT 21H
        JMP EXIT
NOTFOUND:
       LEA DX, MSG2
        MOV AH, 09H
        INT 21H
  EXIT: MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

2. Write ALP macros:
   i) To read a character from the keyboard in the module (1) (in a different file).
   Ii) To display a character in module(2) (from different file).
   iii) Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.

I)
```
READ MACRO
MOV AH,01H
INT 21H
ENDM
```

II)
```
WRITE MACRO
MOV DL,AL
MOV AH,02H
INT 21H
ENDM
```

III)
```
INCLUDE F1.MAC                    ;INCLUDE FILE WHERE U HAVE WRITTEN READ MACRO
INCLUDE F2.MAC                    ;INCLUDE FILE WHERE U HAVE WRITTEN WRITE MACRO
DATA SEGMENT
        MSG DB 10,13,'ENTER A STRING',10,13,'$'
        MSG2 DB 10,13,'ENTERED STRING IS:',10,13,'$'
        STR DB 50 DUP(?)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV CL,0
        LEA SI,STR
        MOV DX,OFFSET MSG
        MOV AH,09H
        INT 21H
UP:     READ
        MOV [SI],AL
        INC SI
        CMP AL,0DH
        JE DOWN
        INC CL
        CMP CL,50
        JNE UP
DOWN:   LEA SI,STR
        MOV DX,OFFSET MSG2
```

```
        MOV AH,09H
        INT 21H
UP1:    MOV AL,[SI]
        WRITE
        DEC CL
        INC SI
        CMP CL,0
        JNZ UP1
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

3. Sort a given set of 'n' numbers in ascending and descending orders using the Bubble Sort algorithm.

```
        DATA SEGMENT
            A db 12h, 00h, 12h, 56h, 34, 11h
            len db $-A
            NL db 10, 13, '$'
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START:
            MOV AX, data
            MOV DS, AX
            MOV CL, 00H
            MOV DL, len
            SUB DL, 01H         ; DL HOLDS len-1
LOOP1:      CMP CL, DL
            JZ DISPL
            MOV BX, 0000H
            MOV DH, DL
            SUB DH, CL          ; DH holds len-1-i
LOOP2:      CMP BL, DH
            JB L2
            INC CL
            JMP LOOP1
L2:         MOV AL, A[BX]
            MOV AH, A[BX+1]
            CMP AL, AH
            JBE L1
            MOV A[BX], AH
            MOV A[BX+1], AL
L1:         INC BL
            JMP LOOP2
DISPL:      MOV BX, 0000h
REPEAT:     MOV DL, A[BX]       ; display the 1st digit
            AND DL, 0F0h
            MOV CL, 04h
            SHR DL, CL
            CMP DL, 09H
            JBE L3
            ADD DL, 07H
L3:         ADD DL, 30H
            MOV AH, 02H
            INT 21H
            MOV DL, A[BX]       ; display the 2nd digit
            AND DL, 0FH
            CMP DL, 09H
            JBE L4
```

```
        ADD DL, 07H
L4:     ADD DL, 30H
        MOV AH, 02H
        INT 21H
        LEA DX, NL          ; TO DISPLAY NEW LINE
        MOV AH, 09H
        INT 21H
        INC BX
        CMP BL, LEN
        JZ EXIT
        JMP REPEAT
EXIT:   MOV AH, 4CH
        INT 21H
    CODE ENDS
    END START
```

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

```
    CLRSCR MACRO
        MOV AH, 00H
        MOV AL, 02H
        INT 10H
    ENDM
    SETCURSOR MACRO X, Y
        MOV DL, Y           ; Y COORDINATE or COLUMN
        MOV DH, X           ; X COORDINATE or ROW
        MOV BH, 00H         ;  CURRENT PAGE
        MOV AH, 02H
        INT 10H
    ENDM
    DATA SEGMENT
            msg1 db "Enter the Character", 10, 13, "$"
            n db ?
    DATA ENDS
    CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
    START:
        MOV AX, DATA
        MOV DS, AX
        LEA DX, msg1
        MOV AH, 09H    ;       DISPLAY MSG1
        INT 21H
        MOV AH, 01H         ; READ CHARACTER
        INT 21H
        MOV n, AL
        CLRSCR
        MOV AL, 02H         ; FOR 80 X 25 BW
        SETCURSOR 12, 39
        MOV DL, n
        AND DL, 0F0H        ; display 1st digit
        MOV CL, 04H
        SHR DL, CL
        CMP DL, 09H
        JBE L1
        ADD DL, 07H
L1:     ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV DL, n
        AND DL, 0FH
```

```
        CMP DL, 09H     ;display 2nd digit
        JBE L2
        ADD DL, 07H
L2:     ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV AH, 01H     ;wait until any key press just like getch in C
        INT 21H
        MOV AH, 4CH
        INT 21H
    CODE ENDS
    END START
```

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

```
    data segment
        str1 db 20 dup(?)          ; Original String
        str2 db 20 dup(?)          ; Reversed String
        n db ?
        msg1 db 10, 13, "String is palindrome",10, 13,"$"
        msg2 db 10, 13, "String is Not palindrome",10, 13,"$"
        msg3 db 10, 13, "enter the string", 10, 13, "$"
        NL db 10, 13 , '$'
    data ends
    code segment
    assume cs:code, ds:data
    start:
        MOV AX, data
        MOV DS, AX
        LEA SI, str1
        LEA DI, str2
        MOV CL, 00H
        LEA DX, msg3
        MOV AH, 09H         ; DISPLAY MSG3
        INT 21H
    UP: MOV AH, 01H         ; to read string until enter key is pressed
        INT 21H
        CMP AL, 0dh         ; for enter key
        JZ L2
        MOV [SI], AL
        INC SI
        INC CL
        JMP UP
    L2: MOV n, CL
        MOV CL, 00H
        DEC SI              ; BRING SI TO POINT TO LAST CHAR OF STR1
    UP1: CMP CL, n
        JZ CHECK
        MOV AL, [SI]
        MOV [DI], AL        ; REVERSE
        DEC SI
        INC CL
        INC DI
        JMP UP1
    CHECK:
        LEA SI, str1
        LEA DI, str2
        MOV CL, 00H
    UP2: CMP CL, n
        JZ PAL
```

```
            MOV AL, [SI]
            CMP AL, [DI]
            JNZ NOTPAL
            INC SI
            INC DI
            INC CL
            JMP UP2
NOTPAL:     LEA DX, msg2        ; Display Not Palindrome
            MOV AH, 09H
            INT 21H
            JMP EXIT
PAL:        LEA DX, msg1        ; Display Palindrome
            MOV AH, 09H
            INT 21H
EXIT:       MOV AH,4CH
            INT 21H
        code ENDS
        END start
```

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

```
        data segment
            str1 db 20 dup(?)                   ; First string
            str2 db 20 dup(?)                   ; Second string
            n1 db ?                             ; length of 1st string
            n2 db ?                             ; length of 2nd string
            msg1 db 10, 13, "Strings are equal",10, 13,"$"
            msg2 db 10, 13, "String are not equal",10, 13,"$"
            msg3 db 10, 13, "enter string1", 10, 13, "$"
            msg4 db 10, 13, "enter string2", 10, 13, "$"
            msg5 db 10,13, “Length of string1”,10,13,”$”
            msg6 db 10,13, “Length of string2”,10,13,”$”
        data ends
        code segment
        assume cs:code, ds:data
start:  MOV AX, data
        MOV DS, AX
        LEA DX, msg3
        MOV AH, 09H         ; DISPLAY MSG3
        INT 21H
        LEA SI, str1
        CALL READSTRING
        MOV n1, CL
        LEA DX, msg4
        MOV AH, 09H         ; DISPLAY MSG4
        INT 21H
        LEA SI, str2
        CALL READSTRING
        MOV n2, CL
        COMP n1, CL         ; CMP n1, n2
        JNZ NOTEQ
        LEA SI, str1
        LEA  DI, str2
        MOV CL, 00h
UP1:    CMP CL, n1          ; CMP with n1 or n2
        JZ STREQUAL
        MOV AL, [SI]
        CMP AL, [DI]
        JNZ NOTEQ
```

```
        INC SI
        INC DI
        INC CL
        JMP UP1
NOTEQ:  LEA DX, msg2        ; Display strings are not equal
        MOV AH, 09H
        INT 21H
        JMP DISPLEN
STREQUAL: LEA DX, msg1      ; Display strings are equal
        MOV AH, 09H
        INT 21H
DISPLEN:  LEA DX, msg5      ; Display msg5
        MOV AH, 09H
        INT 21H
        MOV BL, n1          ; Display n1
        CALL STRINGLENDISP
        LEA DX, msg6        ; Display msg6
        MOV AH, 09H
        INT 21H
        MOV BL, n2          ; Display n2
        CALL STRINGLENDISP
        MOV AH, 4CH
        INT 21H
    READSTRING PROC NEAR
        MOV CL, 00H
    UP: MOV AH, 01H         ; to read string until enter key is pressed
        INT 21H
        CMP AL, 0dh         ; for enter key
        JZ L1
        MOV [SI], AL
        INC SI
        INC CL
        JMP UP
    L1: RET
        READSTRING ENDP
    STRINGLENDISP PROC NEAR
        MOV DL, BL
        AND DL, 0F0H        ; display 1st digit
        MOV CL, 04H
        SHR DL, CL
        CMP DL, 09H
        JBE L2
        ADD DL, 07H
    L2: ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV DL, BL
        AND DL, 0FH
        CMP DL, 09H     ;display 2nd digit
        JBE L3
        ADD DL, 07H
    L3: ADD DL, 30H
        MOV AH, 02H
        INT 21H
        RET
    STRINGLENDISP ENDP
    code ENDS
    END start
```

7. Read your name from the keyboard and display it at a specified location on the screen in front of the message "**What is your name?"** You must clear the entire screen before display.

```
        CLRSCR MACRO
            MOV AH, 00H
            MOV AL, 02H
            INT 10H
        ENDM
        SETCURSOR MACRO X, Y
            MOV DL, Y           ; Y COORDINATE or COLUMN
            MOV DH, X           ; X COORDINATE or ROW
            MOV BH, 00H         ; CURRENT PAGE
            MOV AH, 02H
            INT 10H
        ENDM
        DATA SEGMENT
                msg1 db "Enter Your Name", 10, 13, "$"
                str1 db 30 dup (?)
                n db ?
                str2 db "What is Your Name?$"
                x db 15
                y db 35
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START:
            MOV AX, DATA
            MOV DS, AX
            LEA DX, msg1
            MOV AH, 09H    ; DISPLAY MSG1
            INT 21H
            LEA SI, str1
            MOV CL, 00H
UP1:        CMP CL, 30
            JAE L1
            MOV AH, 01H    ; READ CHARACTER
            INT 21H
            CMP AL, 0dh
            JZ L1
            MOV [SI], AL
            INC SI
            INC CL
            JMP UP1
L1:         MOV n, CL
            CLRSCR
            MOV AL, 02H    ; FOR 80 X 25 BW
            SETCURSOR x, y
            LEA DX, str2
            MOV AH, 09H    ; DISPLAY STR2
            INT 21H
            MOV CL, 00H
            LEA SI, str1
 UP2:       CMP CL, n
            JAE EXIT
            MOV DL, [SI]        ; DISPLAY NAME
            MOV AH, 02H
            INT 21H
            INC SI
            INC CL
            JMP UP2
```

```
EXIT:       MOV AH, 01H     ; wait until any key press just like getch in C
            INT 21H
            MOV AH, 4CH
            INT 21H
        CODE ENDS
        END START
```

8. Compute the factorial of a positive integer ‘n’ using recursive procedure.

```
        DATA SEGMENT
          N DB 06H
          FACT DW ?
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START:
            MOV AX, DATA
            MOV DS, AX
            MOV AX, 1
            MOV BL, N
            MOV BH, 0
            CALL FACTORIAL
            MOV FACT, AX
            MOV AH, 4CH
            INT 21H

            FACTORIAL PROC
            CMP BX, 1
            JE L1
            PUSH BX
            DEC BX
            CALL FACTORIAL
            POP BX
            MUL BX
        L1: RET
            FACTORIAL ENDP
        CODE ENDS
        END START
```

9. Compute **nCr** using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers.

```
        data segment
                n db 10
                r db 9
                ncr db 0
        data ends
        code segment
        assume cs:code,ds:data
start:      mov ax,data
            mov ds,ax
            mov ncr,0
            mov al,n
            mov bl,r
            call encr
            call display
            mov ah,4ch
            int 21h
        encr proc
            cmp al,bl
            je ncr1
            cmp bl,0
```

```
                je ncr1
                cmp bl,1
                je ncrn
                dec al
                cmp bl,al
                je ncrn1
                push ax
                push bx
                call encr
                pop bx
                pop ax
                dec bl
                push ax
                push bx
                call encr
                pop bx
                pop ax
                ret
ncr1:           inc ncr
                ret
ncrn1:          inc al
ncrn:           add ncr,al
                ret
            encr endp

            display proc
                push cx
                mov al,ncr
                mov ch,al
                and al,0f0h
                mov cl,04
                shr al,cl
                cmp al,09h
                jbe next
                add al,07h
            next:add al,30h
                mov dl,al
                mov ah,02h
                int 21h
                mov al,ch
                and al,0fh
                cmp al,09h
                jbe next2
                add al,07h
            next2:add al,30h
                mov dl,al
                mov ah,02h
                int 21h
                pop cx
            ret
            display endp
            code ends
            end start
```

10. Find out whether a given sub-string is present or not in a main string of characters.

```
            data segment
                    T db "NMAMIT"
                    n db $-T
                    P db "MIT"
```

```
                m db $-P
                len db ?
                msg1 db "Sub String Found$"
                msg2 db "Sub String Not Found$"
                temp db ?
        data ends
        code segment
        assume cs:code, ds:data
        start:
            MOV AX, data
            MOV ds, AX
            LEA SI, T
            LEA DI, P
            MOV AL, n      ;len=n-m
            SUB AL, m
            MOV len, AL
            MOV CX, 0000H     ; i->CX
UP1:        CMP CL, len
            JA NOMATCH     ;check if i<=n-m
            MOV DX, 0000H  ; j->DX
UP2:        CMP DL, m
            JAE L1         ;check if j<m
            MOV BX, DX
            MOV AL, [DI][BX]
            MOV temp,AL
            ADD BX, CX
            MOV AL, [SI][BX]
            CMP AL, temp
            JNZ L1
            INC DX
            JMP UP2
L1:         CMP DL, m
            JZ MATCH
            INC CX
            JMP UP1
MATCH:      LEA DX, msg1
            MOV AH, 09H
            INT 21H
            JMP EXIT
NOMATCH:    LEA DX, msg2
            mov ah, 09h
            int 21h
EXIT:       MOV AH, 4CH
            INT 21H
        CODE ENDS
        END start
```

11. Generate the first ‘n’ Fibonacci numbers.

```
        data segment
                f1 db 00h
                f2 db 01h
                f3 db ?
        msg1 db "The Fibonacci series is", 10, 13, "$"
        n db 12
        data ends
        code segment
        assume cs:code, ds:data
start:      mov ax, data
            mov ds, ax
```

```
        lea dx, msg1
        mov ah, 09h
        int 21h
        mov bl, f1
        CALL DISPNUM
        mov dl, ' '
        mov ah, 02h
        int 21h
        mov bl, f2
        CALL DISPNUM
        mov dl, ' '
        mov ah, 02h
        int 21h
        mov ch, 00h
up1:    cmp ch, n
        jae exit
        mov al, f1
        add al, f2
        mov f3, al
        mov bl, f3
        CALL DISPNUM
        mov dl, ' '
        mov ah, 02h
        int 21h
        mov al, f2
        mov f1, al
        mov al, f3
        mov f2, al
        inc ch
        jmp up1
exit:   mov ah, 4ch
        int 21h
DISPNUM PROC NEAR
        MOV DL, BL
        AND DL, 0F0H        ; display 1st digit
        MOV CL, 04H
        SHR DL, CL
        CMP DL, 09H
        JBE L2
        ADD DL, 07H
 L2:    ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV DL, BL
        AND DL, 0FH
        CMP DL, 09H     ;display 2nd digit
        JBE L3
        ADD DL, 07H
 L3:    ADD DL, 30H
        MOV AH, 02H
        INT 21H
        RET
DISPNUM ENDP
code ends
end start
```

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

```
DATA SEGMENT
   MSG1 DB 'CURRENT TIME IS : $'
```

```
        HR DB ?
        MIN DB ?
        SEC DB ?
        MSEC DB ?
    DATA ENDS
    CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:  MOV AX, DATA
        MOV DS, AX
        MOV AH,2CH              ; TO GET SYSTEM TIME
        INT 21H
        MOV HR, CH              ; CH -> HOUR
        MOV MIN, CL             ; CL -> MINUTES
        MOV SEC, DH             ; DH -> SECONDS
        MOV MSEC, DL            ; DL -> 1/100TH SECOND
        LEA DX, MSG1            ; DISPLAY MSG1
        MOV AH, 09H
        INT 21H
        MOV AL, HR              ; IF AL=0D AAM WILL SPLIT THE NIBBLES INTO AH AND AL
        AAM                     ; SO AH=01 AND AL=03
        MOV BX, AX
        CALL DISPLAY            ; DISPLAY HOURS
        MOV DL, ':'             ; DISPLAY ':' AFTER DISPLAYING HOUR
        MOV AH, 02H
        INT 21H
        MOV AL, MIN
        AAM
        MOV BX, AX
        CALL DISPLAY            ; DISPLAY MINUTES
        MOV DL, ':'             ; DISPLAY ':' AFTER DISPLAYING MINUTES
        MOV AH, 02H
        INT 21H
        MOV AL, SEC
        AAM
        MOV BX, AX
        CALL DISPLAY            ; DISPLAY SECONDS
        MOV DL, '.'             ; DISPLAY '.' AFTER DISPLAYING SECONDS
        MOV AH, 02H
        INT 21H
        MOV AL, MSEC
        AAM
        MOV BX, AX
        CALL DISPLAY            ; DISPLAY 1/100TH SECONDS
        MOV AH, 4CH
        INT 21H
        DISPLAY PROC NEAR
            MOV DL, BH
            ADD DL, 30H         ; DISPLAY BH VALUE
            MOV AH, 02H
            INT 21H
            MOV DL, BL
            ADD DL, 30H         ; DISPLAY BL VALUE
            MOV AH, 02H
            INT 21H
            RET
        DISPLAY ENDP
    CODE ENDS
    END START
```

13. Program to simulate a Decimal Up-counter to display 00-99.

```
        CLRSCR MACRO
               MOV AH, 00H
               MOV AL, 02H
               INT 10H
        ENDM

        CODE SEGMENT
        ASSUME CS:CODE
        START:
            CLRSCR  ; TO CLEAR SCREEN
UP1:        MOV AL, 00H  ; INTIALIZE COUNTER TO 00
UP:         CALL CENTER    ; TO MOVE CURSOR TO THE CENTER
            MOV BL, AL
            CALL DISPLAY   ; TO DISPLAY THE BCD NO
            CALL DELAY     ; DELAY
            ADD AL, 01H
            DAA
            CMP AL, 99H
            JB UP
            CALL CENTER
            MOV BL, AL
            CALL DISPLAY
            CALL DELAY
            MOV AH, 01H
            INT 16H       ; IF ANY KEY PRESSED THEN STOP AND EXIT ELSE CONTINUE
            JZ UP1
            MOV AH, 4CH
            INT 21H

            CENTER PROC
                   PUSH AX
                   MOV DL, 39
                   MOV DH, 12
                   MOV BH,0
                   MOV AH,02h
                   INT 10h
                   POP AX
                   RET
               CENTER ENDP

               DISPLAY PROC
                   PUSH AX
                   MOV DL, BL
                   MOV CL,04
                   SHR DL, CL
                   CMP DL,09
                   JBE DOWN1
                   ADD DL,07h
DOWN1:             ADD DL,30h
                   MOV AH,02h
                   INT 21h
                   MOV DL, BL
                   AND DL,0Fh
                   CMP DL,09H
                   JBE DOWN2
                   ADD DL,07h
DOWN2:             ADD DL,30h
```

```
        MOV AH,02h
        INT 21h
        POP AX
        RET
    DISPLAY ENDP

    DELAY PROC
        PUSH AX
        PUSH BX
        PUSH CX
        MOV CX,07FFFh
L1:     MOV BX,0FFFh
L2:     DEC BX
        JNZ L2
        LOOP L1
        POP CX
        POP BX
        POP AX
        RET
    DELAY ENDP

CODE ENDS
END START
```

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

```
CLRSCR MACRO
    MOV AH, 00H
    MOV AL, 02H
    INT 10H
ENDM

SETCURSOR MACRO X, Y
    MOV DL, Y           ; Y COORDINATE or COLUMN
    MOV DH, X           ; X COORDINATE or ROW
    MOV BH, 00H         ; CURRENT PAGE
    MOV AH, 02H
    INT 10H
ENDM

DATA SEGMENT
        BCD_X DB ?
        BCD_Y DB ?
        BIN_X DB ?
        BIN_Y DB ?
        MSG1 DB 'ENTER X COORDINATE :$'
        MSG2 DB 'ENTER Y COORDINATE :$'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    LEA DX, MSG1
    MOV AH, 09H
    INT 21H
    CALL READBCD
    MOV BCD_X, AL
```

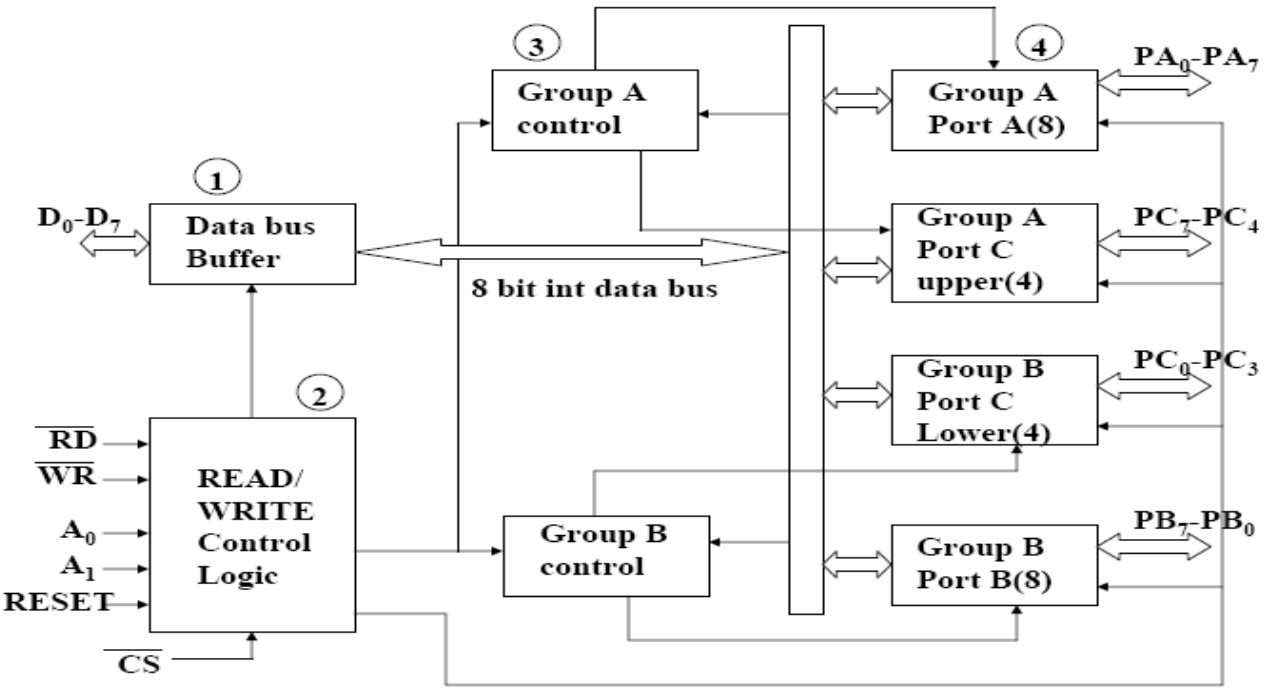
```
LEA DX, MSG2
MOV AH, 09H
INT 21H
CALL READBCD
MOV BCD_Y, AL
MOV BL, BCD_X
CALL BCD_TO_BIN
MOV BIN_X, BL
MOV BL, BCD_Y
CALL BCD_TO_BIN
MOV BIN_Y, BL
CLRSCR
SETCURSOR BIN_X, BIN_Y
MOV AH, 01H
INT 21H
MOV AH, 4CH
INT 21H

READBCD PROC NEAR
  MOV AH, 01H
  INT 21H
  AND AL,0FH
  MOV CL, 04H
  SHL AL, CL
  MOV BL, AL
  MOV AH, 01H
  INT 21H
  AND AL,0FH
  ADD AL, BL
  RET
READBCD ENDP

BCD_TO_BIN PROC NEAR
  PUSH AX
  PUSH CX
  MOV AL, BL
  AND AL, 0F0H
  MOV CL, 04H
  SHR AL, CL
  AND BL, 0FH
  MOV BH, 0AH
  MUL BH
  ADD BL, AL
  POP CX
  POP AX
  RET
BCD_TO_BIN ENDP

CODE ENDS
END START
```

15. Program to create a file (input file) and to delete an existing file.

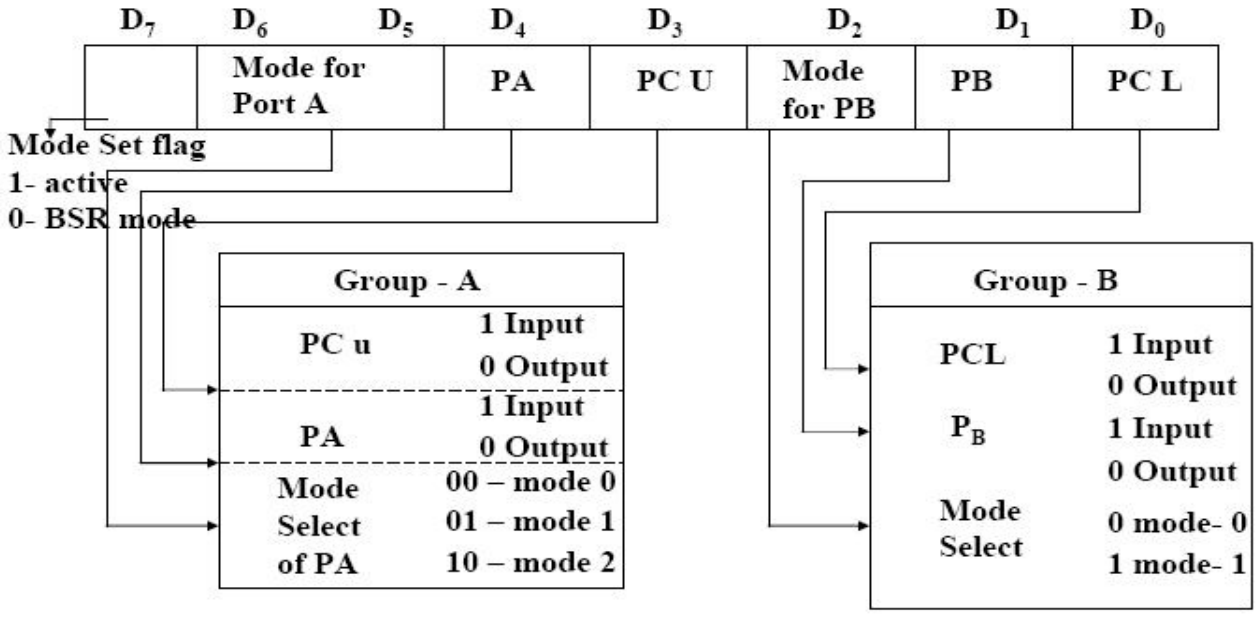
```
DATA SEGMENT
  FNAME DB 'ABCD.DAT', 0
  MSG1 DB 'DELETE Y/N$'
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START:
      MOV AX, DATA
      MOV DS, AX
      LEA DX, FNAME          ; TO CREATE FILE NAMED 'ABC.DAT'
      MOV CX, 20H
      MOV AH, 3CH
      INT 21H
      LEA DX, MSG1
      MOV AH, 09H            ; TO DISPLAY MSG1
      INT 21H
      MOV AH, 01H            ; TO READ CHAR EITHER 'Y' OR 'N'
      INT 21H
      CMP AL, 'Y'
      JNE EXIT               ; IF 'N' THEN DONT DELETE FILE
      LEA DX, FNAME          ; IF 'Y' THEN DELETE FILE
      MOV AH, 41H
      INT 21H       ; TO DELETE FILE 'ABC.DAT'
     JC ERROR

          ; WRITE CODE HERE TO DISPLAY MSG “FILE DELETED”

     JMP EXIT
ERROR:

          ; WRITE CODE HERE TO DISPLAY MSG “FILE CANNOT BE DELETED”

EXIT: MOV AH, 4CH
      INT 21H

CODE ENDS
END START
```

## PART B

### 8255A Internal Block Diagram and System Connections




Block Diagram of 8255




Control Word Format of 8255

1. Read the status of eight input bits from the Logic Controller Interface and display 'FF' if it is even parity bits otherwise display 00. Also display number of 1's in the input data.

```
        PA equ 9800h
        PB equ PA+1
        PC equ PB+1
        PCW equ PC+1
        CW equ 82h
        CODE SEGMENT
        ASSUME CS:CODE
        START:
            MOV AL,CW          ;INITIALIZE 8255
            MOV DX,PCW
            OUT DX,AL
            MOV DX,PB          ;READ THE STATUS OF PORT B AFTER SETTING LEDS
            IN AL,DX           ; AL HAVING PORTB CONTENTS
            MOV CL,0
            MOV CH,8           ;COUNTER TO ROATATE 8 TIMES
            MOV BL,AL
    UP1:    ROL AL,1
            JNC DOWN
            INC CL             ;INCREMENT COUNTER IF LED IS SET
    DOWN:   DEC CH
            JNZ UP1
            MOV CH,CL          ;CH=COUNT
            SHR CL,1           ; IF THE LAST BIT IN CL REGISTER IS 0, IT IS EVEN ELSE ODD
            JC OEVENPARITY
            MOV AL,0FFH
            JMP D1
OEVENPARITY: MOV AL,00H
    D1:     MOV DX,PA
            OUT DX,AL
            MOV AH,01H
            INT 21H
            MOV AL,CH
            MOV DX,PA
            OUT DX,AL
            MOV AH,4CH
            INT 21H
            CODE ENDS
            END START
```

2. Perform the following functions using the Logic Controller Interface.
   i. BCD up-down Counter  ii. Ring Counter.

   Program for up-down counter:

```
        PA EQU 9090H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 80H             ; Control Word 80h
        N EQU 15H

        CODE SEGMENT
        ASSUME CS:CODE
START:  MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
```

```
        UP3:MOV AL,00
        UP1:MOV DX,PA
        OUT DX,AL
        CALL DELAY
        ADD AL,1
        DAA
        CMP AL,N
        JBE UP1

UP2:    MOV DX,PA
        OUT DX,AL
         CALL DELAY
         SUB AL,1
         DAS
         CMP AL,0
         JA UP2

         MOV AH,01H
         INT 16H
         JZ UP3

EXIT:   MOV AH,4CH
        INT 21H

        DELAY PROC
            PUSH CX
            PUSH BX
            MOV CX,05FFFH
THERE:      MOV BX,0FFFFH
HERE:       DEC BX
            JNZ HERE
            DEC CX
            JNZ THERE
            POP BX
            POP CX
            RET
            DELAY ENDP
CODE ENDS
END START
```

Program for up-down counter:

```
        PA EQU 9090h
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 80h              ; Control word 80h

        CODE SEGMENT
        ASSUME CS:CODE
        START:MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
            MOV AL,01H
UP1:        MOV DX,PA
            OUT DX,AL
            CALL DELAY
            ROR AL,1
            PUSH AX
```

```
        MOV AH,01H
        INT 16H
        POP AX
        JZ UP1
        MOV AH,4CH
        INT 21H
    DELAY PROC
        PUSH CX
        PUSH BX
        MOV CX,03FFFH
THERE:  MOV BX,03FFFH
HERE:   DEC BX
        JNZ HERE
        DEC CX
        JNZ THERE
        POP BX
        POP CX
        RET
        DELAY ENDP
    CODE ENDS
    END START
```

3. Read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X*Y.

```
PA EQU 9800H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 82H
DATA SEGMENT
        X DB ?
        Y DB ?
        PROD DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX, DATA
    MOV DS, AX
    MOV AL,CW
    MOV DX,PCW
    OUT DX, AL
    MOV DX, PB
    IN AL,DX
    MOV X,AL
    MOV AH,01H
    INT 21H
    MOV DX,PB
    IN AL,DX
    MOV Y,AL
    MOV AL,X
    MUL Y
    MOV DX,PA
    OUT DX, AL
    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

4. 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.

```
        PA EQU 0DD00H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 80H                  ; Control Word

        DATA SEGMENT
        FIRE DB 86H,8FH,0CFH,8EH    ; E, R, I, F
        HELP DB 8CH,0C7H,86H,89H    ; P, L, E, H
        DATA ENDS

        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START:MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
UP:         LEA SI,FIRE
            CALL DISPLAY
            CALL DELAY
            LEA SI,HELP
            CALL DISPLAY
            CALL DELAY
            JMP UP
EXIT:       MOV AH,4CH
            INT 21H

          DISPLAY PROC
          MOV CL,4
UP2:      MOV AL,[SI]
          MOV BL,8
UP1:      ROL AL,1
          MOV DX,PB
          OUT DX,AL
          PUSH AX
          MOV AL,01H
          MOV DX,PC
          OUT DX,AL
          MOV AL,00H
          MOV DX,PC
          OUT DX,AL
          POP AX
          DEC BL
          JNZ UP1
          INC SI
          DEC CL
          JNZ UP2
          RET
        DISPLAY ENDP

        DELAY PROC
            PUSH CX
            PUSH BX
            MOV CX,05FFFH
L1:         MOV BX,0FFFFH
```

```
L2:         DEC BX
            JNZ L2
            LOOP L1
            POP BX
            POP CX
            PUSH AX
            MOV AH,01H
            INT 16H
            POP AX
            JNZ EXIT
            RET
          DELAY ENDP
CODE ENDS
END START
```

5. Assume any suitable message of 12 characters length and display it in the rolling fashion 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.

```
          PA EQU 0DD00H
          PB EQU PA+1
          PC EQU PB+1
          PCW EQU PC+1
          CW EQU 80H

          DATA SEGMENT
          MSG DB 0FFH,0FFH,0FFH,0FFH          ; _ _ _ _
              DB 8EH,0CFH,8FH,86H             ; F, I , R, E
              DB 89H, 86H, 0C7H, 8CH          ; H,E,L,P
              DB 8FH,0CFH,0C8H,90H            ; R,I,N,G
              DB 0FFH,0FFH,0FFH               ; _ _ _
          DATA ENDS

           CODE SEGMENT
           ASSUME CS:CODE,DS:DATA
           START:
             MOV AX,DATA
             MOV DS,AX
             MOV AL,CW
             MOV DX,PCW
             OUT DX,AL
          UP4:
            MOV CX,16
            LEA SI,MSG
          UP3:
            CALL DISPLAY
            CALL DELAY
            INC SI
            LOOP UP3
           JMP  UP4
          EXIT:
          MOV AH,4CH
          INT 21H
          DISPLAY PROC
            PUSH CX
            MOV CX,4
            MOV DI,SI
            ADD DI,03
          UP2:
             MOV BL,8
```

```
  MOV AL,[DI]
UP1:
  ROL AL,1
  MOV DX,PB
  OUT DX,AL
  PUSH AX
  MOV AL,1
  MOV DX,PC
  OUT DX,AL
  MOV AL,0
  MOV DX,PC
  OUT DX,AL
  POP AX
  DEC BL
  JNZ UP1
  DEC DI
  LOOP UP2
  POP CX
  RET
  DISPLAY ENDP
  DELAY PROC
    PUSH CX
    PUSH BX
    MOV CX,05FFFH
L1: MOV BX,0FFFFH
L2: DEC BX
   JNZ L2
   LOOP L1
   POP BX
   POP CX
   PUSH AX
   MOV AH,01H
   INT 16H
   POP AX
   JNZ EXIT
   RET
DELAY ENDP
CODE ENDS
END START
```

6. Convert a 16-bit binary value (assumed to be an unsigned integer) to BCD and display it from left to right and right to left for specified number of times on a 7-segment display interface.

```
PA EQU 9800H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
HEXNUM DW 0018H
REMARR DB 4 DUP(?)
DISPTABLE DB 0C0H, 0CFH, 0A4H, 0B0H, 9BH
       DB 92H, 82H, 0F8H, 80H, 90H
SSCODE DB 0FFH,0FFH,0FFH,0FFH
     DB ?,?,?,?
     DB 0FFH,0FFH,0FFH
DATA ENDS

ASSUME CS:CODE, DS:DATA
```

```
        CODE SEGMENT
        START:
            MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
            CALL CONVERT
UP3:        MOV CX,08
            LEA SI,SSCODE
UP1:        MOV DI,SI
            ADD DI,03
            CALL SSDISP
            CALL DELAY
             INC SI
             DEC CX
             JNZ UP1
             MOV CX,07
             LEA SI, SSCODE
             ADD SI, 10
UP2:         MOV DI,SI
            CALL SSDISP
            CALL DELAY
            DEC SI
            DEC CX
            JNZ UP2
            PUSH AX
            MOV AH,01
            INT 16H
            JZ UP3
            POP AX
LAST:       MOV AH,4CH
            INT 21H
           CONVERT PROC NEAR
            LEA SI,REMARR
            MOV CX,04
            MOV AX,HEXNUM
UP4:        MOV DX,0
            MOV BX,10
            DIV BX
            MOV [SI],DL
            INC SI
            DEC CX
            JNZ UP4
            LEA SI,REMARR   ; OR ADD SI,04
            LEA BX, DISPTABLE
            MOV CX,04
            LEA DI,SSCODE
            ADD DI,07
UP5:        MOV AL, [SI]
            XLAT
            MOV [DI],AL
            INC SI
            DEC DI
            DEC CX
            JNZ UP5
            RET
            CONVERT ENDP
```

```
        SSDISP PROC
            PUSH CX
            MOV CX,04
UP7:        MOV BL,08
            MOV AL,[DI]
UP6:        ROL AL,01
            MOV DX,PB
            OUT DX,AL
            PUSH AX
            MOV DX,PC
            MOV AL,01
            MOV DX,PC
            OUT DX,AL
            MOV AL,00
            MOV DX,PC
            OUT DX,AL
            POP AX
            DEC BL
            JNZ UP6
            DEC DI
            DEC CX
            JNZ UP7
            POP CX
            RET
        SSDISP ENDP
        DELAY PROC
            PUSH CX
            PUSH BX
            PUSH AX
            MOV CX, 0FFFFH
THERE:      MOV BX, 4FFFH
HERE:       DEC BX
            JNZ HERE
            DEC CX
            JNZ THERE
            MOV AH,01
            INT 16H
            JNZ LAST
            POP AX
            POP BX
            POP CX
            RET
            DELAY ENDP
        CODE ENDS
        END START
```

7. Drive a Stepper Motor interface to rotate the motor in clockwise direction by N steps (N is specified by the examiner). Introduce suitable delay between successive steps.

```
PA EQU 0DD00H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
N DB 50
DATA ENDS
```

```
        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START:MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
            MOV BL,N
            MOV AL,00H
            MOV DX,PA
            OUT DX,AL
            MOV AL,01H
            MOV DX,PA
            OUT DX,AL
            MOV AX,9911H   (9988H) CLKWISE
UP:         MOV DX,PA
            OUT DX,AL
            CALL DELAY
            ROR AL,1
            XCHG AL,AH
            OUT DX,AL
            CALL DELAY
            ROR AL,1
            XCHG AL,AH
            DEC BL
            JNZ UP
EXIT:       MOV AH,4CH
            INT 21H

            DELAY PROC
            PUSH BX
            MOV CX,03FFFH
THERE:      MOV BX,0FFFH
HERE:       DEC BX
            JNZ HERE
            LOOP THERE
            POP BX
            RET
            DELAY ENDP
            CODE ENDS
            END START
```

8. Drive a stepper motor interface to rotate the motor in anticlockwise direction by N steps (N is specified by the examiner). Introduce suitable delay between successive steps.

```
        PA EQU 0DD00H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 80H
        DATA SEGMENT
        N DB 50
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:      MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
```

```
        MOV BL,N
        MOV AL,00H
        MOV DX,PA
        OUT DX,AL
        MOV AL,01H
        MOV DX,PA
        OUT DX,AL
        MOV AX,3311H
UP:     MOV DX,PA
        OUT DX,AL
        CALL DELAY
        ROL AL,1
        XCHG AL,AH
        OUT DX,AL
        CALL DELAY
        ROL AL,1
        XCHG AL,AH
        DEC BL
        JNZ UP
EXIT:   MOV AH, 4CH
        INT 21H

        DELAY PROC
        PUSH BX
        MOV CX,03FFFH
THERE:  MOV BX,0FFFH
HERE:   DEC BX
        JNZ HERE
        LOOP THERE
        POP BX
        RET
        DELAY ENDP
        CODE ENDS
        END START
```

9. Drive a stepper motor interface to rotate the motor by N steps left direction and N steps right direction (N is specified by the examiner). Introduce suitable delay between successive steps.

Solution: Similar to problem no 7 and 8. Just combine the code to rotate in clockwise and anticlockwise direction.

10. Scan an 8 x 3 keypad for key closure and to store the code of the key pressed in a memory location or display on screen. Also display row and column numbers of the key pressed.

```
        PA EQU 0DD00H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 90H
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
            ROW DB ?
            COL DB ?
            VAL DB ?
        DATA ENDS
        CODE SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
```

```
UP:         MOV AL,01          ; PASS CURRENT TO 1ST ROW
            MOV DX,PC
            OUT DX,AL
            MOV DX,PA
            IN AL,DX
            CMP AL,0
            JNE FIRSTROW       ; IF ANY BIT SET IN AL THEN SOME CHAR PRESSED IN 1ST ROW
            MOV AL,02          ; PASS CURRENT TO 2ND ROW
            MOV DX,PC
            OUT DX,AL
            MOV DX,PA
            IN AL,DX
            CMP AL,0
            JNE SECONDROW
            MOV AL,04          ; PASS CURRENT TO 3RD ROW
            MOV DX,PC
            OUT DX,AL
            MOV DX,PA
            IN AL,DX
            CMP AL,0
            JNE THIRDROW
            JMP UP
FIRSTROW:   CALL DELAY
            MOV BL,1  ;ROW
            MOV BH,1 ;COL
            MOV CL,0  ;VALUE
UP1:        ROR AL,1
            JC DISPLAY
            INC BH
            INC CL
            JMP UP1
SECONDROW:  CALL DELAY
            MOV BL,2
            MOV BH,1
            MOV CL,8
UP2:        ROR AL,1
            JC DISPLAY
            INC BH
            INC CL
            JMP UP2
THIRDROW:   CALL DELAY
            MOV BL,3
            MOV BH,1
            MOV CL,10H
UP3:        ROR AL,1
            JC DISPLAY
            INC BH
            INC CL
            JMP UP3
DISPLAY:    MOV ROW,BL
            MOV COL,BH
            MOV VAL,CL
            MOV BL,ROW
            CALL DISP8B
            MOV BL,COL
            CALL DISP8B
            MOV BL,VAL
            CALL DISP8B
```

```
                MOV AH,4CH
                INT 21H

                DISP8B PROC
                PUSH AX
                MOV CL,4
                MOV AL,BL
                AND AL,0F0H
                CMP AL,09H
                JBE D1
                ADD AL,07H
D1:             ADD AL,30H
                MOV DL,AL
                MOV AH,02H
                INT 21H
                MOV AL,BL
                AND AL,0FH
                CMP AL,09H
                JBE D2
                ADD AL,7
D2:             ADD AL,30H
                MOV DL,AL
                MOV AH,02H
                INT 21H

                POP AX
                RET
                DISP8B ENDP

                DELAY PROC
                PUSH CX
                PUSH BX
                MOV CX,0FFFFH
THERE:          MOV BX,0FFFFH
HERE:           DEC BX
                JNZ HERE
                DEC CX
                JNZ THERE
                POP BX
                POP CX
                RET
                DELAY ENDP
                CODE ENDS
                END START
```

11. Scan an 8 x 3 keypad for key closure and simulate ADD and SUBTRACT operations as in a calculator.

    Solution:

    The problem is similar to question no 10. Only difference is to read two operands, an operator and to display the result in the monitor. Write a procedure to read a character from the keypad which is given in the previous program.

12. Generate the Sine Wave using DAC interface (The output of the DAC is to be displayed on the CRO).

    The $V_L$ (Voltage Level) is calculated as follows:

    $V_L = V_{REF} / 2 + \sin\Theta$. The values are calculated for each interval of an angle $\Theta$.

```
PA EQU 0DD00H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H
```

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        TABLE DB 128,136,144,153,160,166,172,175,178,178,178,175,172,166,160,153,144
              DB 136,128,119,110,102,95,88,83,80,77,76,77,80,83,88,95,102,110,119, 128
        DATA ENDS
        CODE SEGMENT
        START:MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
UP2:        MOV BL,37           ; COUNT =TOTAL NO OF VALUES
            LEA SI,TABLE
UP1:        MOV DX,PA
            MOV AL,[SI]
            INC SI
            OUT DX,AL
            CALL DELAY
            DEC BL
            JNZ UP1
            JMP UP2
            MOV AH,4CH
            INT 21H
         DELAY PROC
            PUSH CX
            PUSH BX
            MOV CX,05FFH
THERE:      MOV BX,0FFH
HERE:       DEC BX
            JNZ HERE
            DEC CX
            JNZ THERE
            POP BX
            POP CX
            RET
         DELAY ENDP
         CODE ENDS
         END START
```

13. Generate a Half Rectified Sine wave form using the DAC interface. (The output of the DAC is to be displayed on the CRO).

```
        PA EQU 0DD00H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 80H
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        TABLE DB 128,136,144,153,160,166,172,176,179,179,179,176,172,166,160,153,144,136,128
              DB 128  128,128,128,128,128,128,128,128,128,128,128, 128,128,128,128,128
        DATA ENDS
        CODE SEGMENT
START:      MOV AX,DATA
            MOV DS,AX
            MOV AL,CW
            MOV DX,PCW
            OUT DX,AL
UP2:        MOV BL,37           ; COUNT =TOTAL NO OF VALUES
```

```
        LEA SI,TABLE
UP1:    MOV DX,PA
        MOV AL,[SI]
        INC SI
        OUT DX,AL
        CALL DELAY
        DEC BL
        JNZ UP1
        JMP UP2
        MOV AH,4CH
        INT 21H
    DELAY PROC
        PUSH CX
        PUSH BX
        MOV CX,02FFH
THERE:  MOV BX,0FFH
HERE:   DEC BX
        JNZ HERE
        DEC CX
        JNZ THERE
        POP BX
        POP CX
        RET
    DELAY ENDP
    CODE ENDS
    END START
```

14. Generate a Fully Rectified Sine waveform using the DAC interface. (The output of the DAC is to be displayed on the CRO).

```
    PA EQU 0DD00H
    PB EQU PA+1
    PC EQU PB+1
    PCW EQU PC+1
    CW EQU 80H
    ASSUME CS:CODE,DS:DATA
    DATA SEGMENT
    TABLE DB 128, 136,144,153,160,166,172,176,179,179,179,176,172,166,160,153,144,136,128
          DB 128,136,144,153,160,166,172,176,179,179,179,176,172,166,160,153,144,136,128
    DATA ENDS
    CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
UP2:    MOV BL,38
        LEA SI,TABLE
UP1:    MOV DX,PA
        MOV AL,[SI]
        INC SI
        OUT DX,AL
        CALL DELAY
        DEC BL
        JNZ UP1
        JMP UP2
        MOV AH, 4CH
        INT 21H
```

```
        DELAY PROC
            PUSH CX
            PUSH BX
            MOV CX,09FFH
THERE:      MOV BX,0FFH
HERE:       DEC BX
            JNZ HERE
            DEC CX
            JNZ THERE
            POP BX
            POP CX
            RET
        DELAY ENDP
        CODE ENDS
        END START
```

15. Drive an elevator interface in the following way:

    i. Initially the elevator should be in the ground floor, with all requests in OFF state.

    ii. When a request is made from a floor, the elevator should move to that floor, wait there for a couple of seconds, and then come down to ground floor and stop. If some requests occur during going up or coming down they should be ignored.

```
        PA EQU 0DD00H
        PB EQU PA+1
        PC EQU PB+1
        PCW EQU PC+1
        CW EQU 82H
        DATA SEGMENT
            GLOW DB 00H,03H,06H,09H
            CLEAR DB 0E0H,0D3H,0B6H,79H
        FLR DB 00H
        DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV DX,PA
        MOV AL,0F0H          ;  TO GLOW LED 14 CORRESPONDING TO GROUND FLOOR
        OUT DX,AL
AGAIN:  MOV CL,00
        MOV CH,0F0H
        MOV DX,PB
        IN AL,DX
        AND AL,0FH
        CMP AL,0FH           ; IF ANY KEY PRESSED OR NOT
        JE AGAIN             ; IF NOT REPEAT UNTIL A KEY IS PRESSED
        MOV CL,FLR
        MOV AH,00H
BACK:   ROR AL,01
        JNC NEXT
        INC AH
        JMP BACK
NEXT:   MOV FLR,AH
        MOV AL,AH
        LEA BX,GLOW
        XLAT
```

```
        MOV AH, AL
        ADD AH,0F0H
        MOV DX,PA
UP:     MOV AL,CH
        OUT DX,AL
        CALL DELAY
        CMP AH,CH
        JE EXIT
        ADD CH,01
        JMP UP

EXIT:   MOV AL,FLR
        LEA BX,CLEAR
        XLAT
        MOV DX,PA
        OUT DX,AL
        CALL DELAY
        CALL DELAY
        CALL DELAY
        MOV AL, CH
D10:    DEC AL
        MOV DX, PA
        OUT DX, AL
        CALL DELAY
        CMP AL, 0F0H
        JNE D10
        MOV AH,01H
        INT 16H
        JZ AGAIN
        MOV AH,4CH
        INT 21H

        DELAY PROC
        PUSH BX
        PUSH CX
        MOV CX,0FFFFH
THERE:  MOV BX,0FFFH
HERE:   DEC BX
        JNZ HERE
        DEC CX
        JNZ THERE
        POP CX
        POP BX
        RET
        DELAY ENDP
        CODE ENDS
        END START
```