

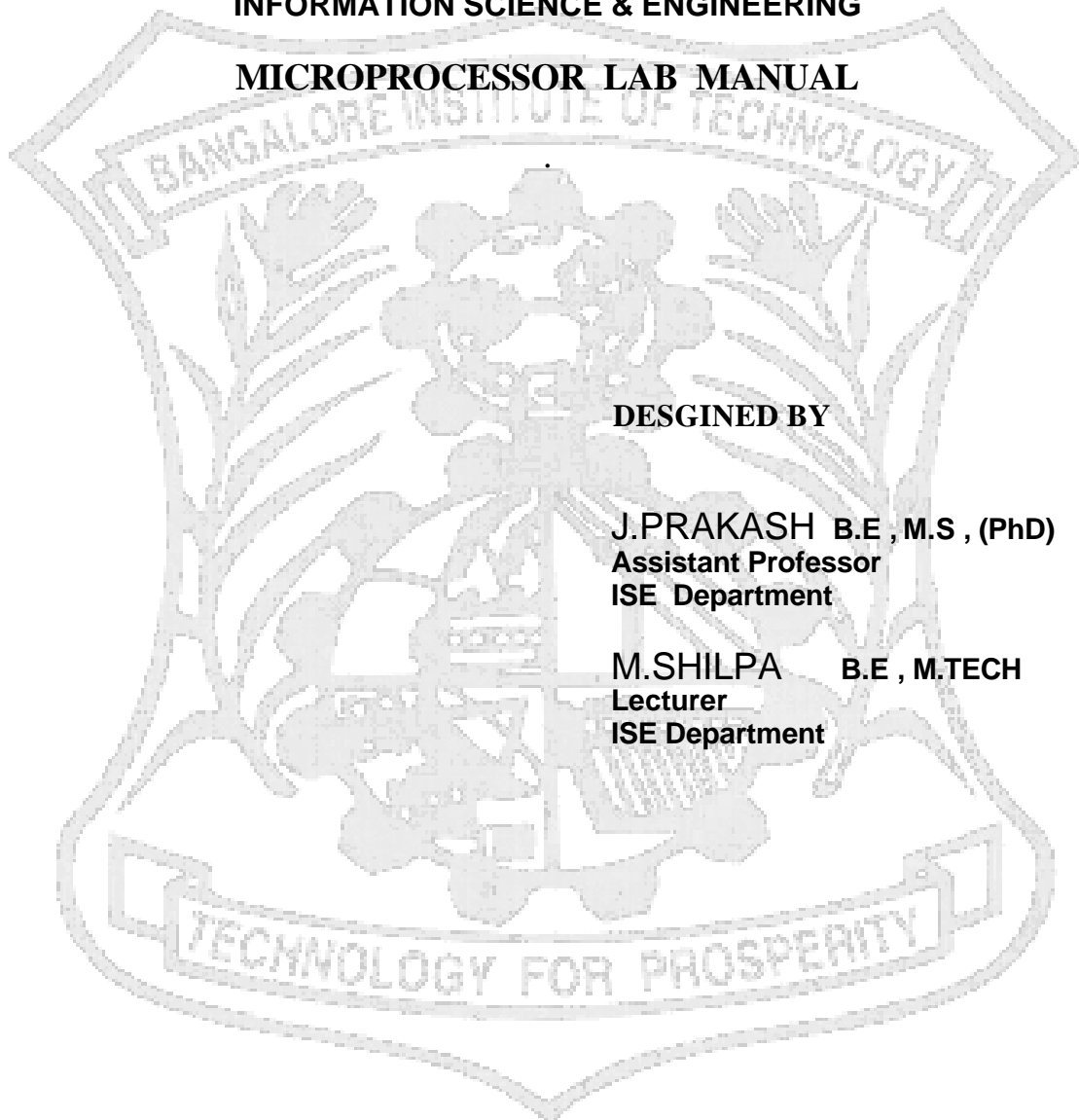


BANGALORE INSTITUTE OF TECHNOLOGY

K. R. ROAD, V. V. PURAM, BANGALORE – 560004.

**Department of
INFORMATION SCIENCE & ENGINEERING**

MICROPROCESSOR LAB MANUAL



DESIGNED BY

**J.PRAKASH B.E , M.S , (PhD)
Assistant Professor
ISE Department**

**M.SHILPA B.E , M.TECH
Lecturer
ISE Department**

PROGRAM 1A

Search a key element in a list of N numbers using binary search

```

.MODEL SMALL                ; type of model
.STACK 64H
.DATA                        ; Data segment begins
    ARRAY DW 1000H,2000H,3000H,4000H,5000H,6000H ;a sorted array

    LEN DW ($-ARRAY)/2 ; to find the length of the array
    MSG DB 0AH,0DH, ' THE KEY TO BE FOUND : 3000H $'
    KEY DW 3000H           ; key to be searched
    SUC DB 0AH,0DH, ' SUCCESSFUL SEARCH !! FOUND AT POSITION '
    POS DB ' $'
    FAIL DB 0AH,0DH, 'UNSUCCESSFUL SEARCH !! $'
.CODE                          ; code segment begins
    MOV AX,@DATA              ; move the address of the data segment to DS
    MOV DS,AX

    LEA DX,MSG                ; to o/p the key to be searched
    MOV AH,09H
    INT 21H
    MOV BX,1                   ; BX register stores the low value
    MOV DX,LEN                 ;DX has the high value
    MOV CX,KEY
AGAIN : CMP BX,DX              ; compare low and high
        JA FAILURE            ; if (low >high) then the search has failed
        MOV AX,BX              ;to find the mid
        ADD AX,DX              ;mid =(low+high)/2
        SHR AX,1               ; shifting right to divide by 2
        MOV SI,AX
        DEC SI
        ADD SI,SI
        CMP CX,ARRAY[SI]      ; compare key with array[mid]
        JAE NEXT              ; if equal search is successful
        DEC AX
        MOV DX,AX              ;else change low and high appropriately
        JMP AGAIN              ; loop back
NEXT : JE SUCCESS              ; check again if key=array[mid]
        INC AX                 ; if so find the position where it is found
        MOV BX,AX
        JMP AGAIN

SUCCESS : ADD AL,'0'           ; to o/p the success message
        MOV POS,AL
        LEA DX,SUC
        JMP DISPLAY1

FAILURE : LEA DX,FAIL          ; to o/p the failure message
DISPLAY1 : MOV AH,09H
        INT 21H                ; to return to DOS
        MOV AH,4CH
        INT 21H
        END                     ; END of program

```

PROGRAM 1B

READ THE STATUS OF A BIT INPUT FROM LOGIC CONTROLLER AND DISPLAY 'FF' IF EVEN PARITY OTHERWISE '00'. ALSO DISPLAY THE NUMBER OF ONES.

```
.MODEL SMALL
.STACK 64H
.CODE
    MOV AL,82H           ; control word
    MOV DX,0A403H       ; CWR
    OUT DX,AL           ; send to CWR

    MOV DX,0A401H       ; input from port B
    IN AL,DX

    MOV BL,AL
    AND AL,0FFH
    JP EV                ; check parity
    JMP ODD

EV:
    MOV DX,0A400H
    MOV AL,0FFH
    OUT DX,AL
    CALL DILAY
    JMP COUNT

ODD:
    MOV DX,0A400H
    MOV AL,00H
    OUT DX,AL
    CALL DILAY
    JMP COUNT

DILAY PROC
    PUSH BX
    PUSH CX
    MOV BX,4FFFH
LOOP2:
    MOV CX,0FFFFH
LOOP1:
    LOOP LOOP1
    DEC BX
    JNZ LOOP2
    POP CX
    POP BX
    RET
DILAY ENDP

COUNT:
    MOV CL,08H          ; 8 bits
    MOV BH,00H          ; to count the no. of ones
    CLC

LP:
    ROR BL,01
```

```
JNC LP2
INC BH
LP2:
LOOP LP
MOV DX,0A400H
MOV AL,BH
OUT DX,AL      ; to display the no. of ones
MOV AH,4CH
INT 21H
END
```

PROGRAM 2A

MAIN MODULE: PROGRAM TO CALL TWO FAR PROCEDURES FOR READING AND PRINTING CHARACTERS INPUT FROM A KEYBOARD

```
.MODEL SMALL
.STACK 64H
.DATA
    ARR DB 50 DUP (?)
.CODE
    EXTRN READKEY : FAR
    EXTRN ECHO : FAR
START :
    MOV AX,@DATA
    MOV DS,AX
    MOV SI,OFFSET ARR
    MOV DI,OFFSET ARR
    MOV CX,00H
NEXT :
    CALL READKEY
    INC CX
    CMP AL,0DH
    JNZ NEXT
    MOV DL,10
    MOV AH,02H
    INT 21H
    MOV DL,13
    MOV AH,02H
    INT 21H
NEXT1 :
    MOV DL,[DI]
    CALL ECHO
    INC DI
    LOOP NEXT1
    MOV AH,4CH
    INT 21H
END START
END
```

MODULE2A1: PROCEDURE TO ACCEPT A KEY FROM KEYBOARD

```
.MODEL SMALL
.STACK 64H
.CODE
    PUBLIC READKEY
READKEY PROC
    MOV AH,01H
    INT 21H
    MOV [SI],AL
    INC SI
    RET
READKEY ENDP
END
```

MODULE2A2: PROGRAM TO ECHO THE INPUT CHARACTER TO OUTPUT SCREEN

```
.MODEL SMALL
.STACK 64H
.CODE
    PUBLIC ECHO
ECHO PROC
    MOV AH,02H
    INT 21H
    RET
ECHO ENDP
END
```

PROGRAM 2B

Simulating BCD up-down counter and ring counter

```

.MODEL SMALL
.DATA
MSG1 DB 10,13,"STIMULATING BCD UP_DOWN COUNTER $"
MSG2 DB 10,13,"STIMULATING RING COUNTER $"
.CODE
    MOV AX,@DATA           ; initialize the data segment
    MOV DS,AX
    MOV ES,AX             ; and also the extra segment
    MOV AH,09H           ; print MSG1
    LEA DX,MSG1          ; to tell what function its doing
    INT 21H

    MOV AL,82H           ; move the control word into CWR
    MOV DX,0A403H
    OUT DX,AL
    MOV DX,0A400H        ; port A used as o/p port
    MOV AL,00H

UP_CNTR:
    OUT DX,AL            ; o/p the count
    CMP AL,09H          ; if count >=9 start down counter
    JGE NEXT1
    INC AL               ; else increment AL and loop back
    CALL DE_LAY         ; wait after each display
    JMP UP_CNTR

NEXT1:
    MOV AL,09H           ; start counter from 9

DOWN_CNTR:
    OUT DX,AL            ; o/p the count
    CMP AL,00H          ; if count is <=0 start down counter
    JE NEXT2
    DEC AL               ; else dec al and loop back
    CALL DE_LAY         ; wait after each display
    JMP DOWN_CNTR

NEXT2:
    MOV AH,01H
    INT 21H
    MOV AH,09H
    LEA DX,MSG2          ; display MSG2
    INT 21H
    MOV AL,01H           ; pattern for ring counter
    MOV CX,10H           ; it needs to rotate this many times
    MOV DX,120H          ; port A address

RING_CNTR:
    OUT DX,AL            ; o/p the bit pattern
    CALL DE_LAY         ; wait
    ROR AL,1             ; rotate the pattern
    LOOP RING_CNTR
    MOV AH,4CH           ; return to DOS
    INT 21H

DE_LAY PROC

```

```
        PUSH CX
        MOV CX, 3FFFH           ; instructions which consume time
LP1:    MOV BP,FFFFH           ; but do not alter the contents
LP2:    DEC BP                  ; of any register
        JNZ LP2
        LOOP LP1
        POP CX
        RET
DE_LAY ENDP
        END                    ; end of program
```

PROGRAM 3A

PROGRAM TO SORT THE GIVEN SET OF ELEMENTS IN ASCENDING AND DESCENDING ORDER

```

.MODEL SMALL
.STACK 64
.DATA
    ARR DB 17H,3H,10H,5H,00H,14H,23H
    COUNT EQU 07H          ; 7 Array elements
    N DB 25 DUP(0)
    MSG1 DB 10,13,' HOW DO YOU WANT TO SORT THE ARRAY ? $'
    MSG2 DB 10,13,' 1:> ASCENDING ORDER      2:> DESCENDING ORDER $'
    MSG3 DB 10,13,' ENTER YOUR CHOICE : $'

.CODE
    MOV AX,@DATA           ; Load the address of DATA SEGEMENT into DS
    MOV DS,AX
    LEA SI,ARR
    LEA DX,MSG1           ; ask the user if he wants to in ascending
    MOV AH,09H           ; or descending order
    INT 21H
    LEA DX,MSG2
    MOV AH,09H
    INT 21H
    LEA DX,MSG3
    MOV AH,09H
    INT 21H
    MOV CX,COUNT-1
    MOV DX,CX
    MOV AH,01H
    INT 21H               ; compare your choice with 1
    CMP AL,'1'           ; if '1' sort in ascending order
    JZ ASCEND            ; else descending order
    JMP DESCEND

ASCEND :                 ; bubble-sort for ascending order
LP1 :                   ; outer loop begins
    MOV CX,DX
    LEA SI,ARR

LP2 :                   ; inner loop begins
    MOV AL,[SI]
    CMP AL,[SI+1]
    JB NEXT1
    XCHG [SI+1],AL      ; swapping elements
    XCHG [SI],AL

NEXT1 :
    ADD SI,01
    LOOP LP2            ; inner loop ends
    DEC DX
    JNZ LP1             ; outer loop ends
    JMP STOP

DESCEND :               ; bubble sort - descending order
LP3 :                   ; outer loop starts
    MOV CX,DX
    MOV SI,OFFSET ARR

LP4 :                   ;inner loop starts

```

```
        MOV AL,[SI]
        CMP AL,[SI+1]
        JGE NEXT2
        XCHG [SI+1],AL    ; swapping of elements
        XCHG [SI],AL
NEXT2:
        ADD SI,01
        LOOP LP4          ; inner loop ends
        DEC DX
        JNZ LP3          ; outer loop ends
STOP  :
        MOV AH,4CH
        INT 21H          ; return to DOS
END
```

PROGRAM 3B

READ THE STATUS OF 2 8-BIT INPUTS x AND y USING LOGIC CONTROLLER AND DISPLAY x*y

```

.model small
.stack 64h
.data
msg1 db 10,13,"entr the first operand $"
msg2 db 10,13,"enter the second operand $"
opr1 db ?
opr2 db ?                ; the two numbers
.code                    ; start the code segment
mov ax,@data             ; load address into data segment
mov ds,ax

mov al,82h
mov dx,0123h             ; address of CWR
out dx,al                ; load control word into CWR
lea dx,msg1              ; ask user to enter the first number
mov ah,09h
int 21h
mov ah,01h               ; wait until user enters the number
int 21h
mov dx,0121h
in al,dx
mov opr1,al

lea dx,msg2              ; ask the user to enter the second number
mov ah,09h
int 21h
mov ah,01h               ; wait until user enters the number
int 21h
mov dx,0121h
in al,dx
mov opr2,al

mov ah,00h
mov bl,opr1              ; multiply two operands
mul bl
mov bx,ax                ; move to register AX
mov dx,0120h
out dx,al                ; see the lower byte on the logic
controller
mov ah,01h
int 21h
mov al,bh
out dx,al
mov ah,01h               ; wait until user wants to see the result
int 21h                  ; i.e., the higher byte

mov ah,4ch
int 21h                  ; return to DOS
end

```

PROGRAM 4A

PROGRAM TO PRINT A ALPHANUMERIC CHARACTER AT THE CENTRE OF THE SCREEN

```

.MODEL SMALL
.DATA
    MSG DB 10,13," ENTER A ALPHANUMERIC CHARACTER : $"
    CHAR DB 0
.CODE
    MOV AX,@DATA           ; Load the address of DATA SEGEMENT into DS
    MOV DS,AX
    LEA DX,MSG             ; load the starting address of MSG
    MOV AH,09H             ; display the message
    INT 21H
    MOV AH,01H
    INT 21H
    MOV CHAR,AL            ; store in CHAR
    AND AL,0F0H            ; get the upper nibble
    MOV CL,04H
    ROR AL,CL              ; rotate AL 4 times i.e divide by 16
    MOV AH,02H             ; move the cursor to the centre
    MOV BH,00              ; set the page number to 00
    MOV DH,10              ; set the row to 10
    MOV DL,40              ; set the column to 40
    INT 10H
    CALL DISP              ; display the upper nibble value
    MOV AL,CHAR
    AND AL,00FH            ; get the lower nibble
    MOV DL,AL
    CALL DISP              ; display the lower nibble value
    MOV AH,4CH
    INT 21H
DISP PROC
    MOV DL,AL              ; get the character to display
    CMP DL,09H             ; check if the value is between 0-9 or A-F
    JLE NEXT              ; if 0-9 add 30H to it
    SUB DL,0AH             ; if A-F sub 0AH and add 41H
    ADD DL,41H
    JMP PRINT
NEXT : ADD DL,30H          ; if 0-9 add 30H to it
PRINT : MOV AH,02H
        INT 21H
        RET               ; return to main program
DISP ENDP
END                          ; end of the program

```

PROGRAM 4B**TO DISPLAY 'FIRE' AND 'HELP' ON A 7-SEGMENT DISPLAY**

```

.MODEL SMALL
.STACK 64h
.DATA
    A1 DB 86h,88h,0f9h,8eh
    A2 DB 8ch,0c7h,86h,89h

.CODE
    MOV AX,@DATA
    MOV DS,AX

    MOV AL,80H
    MOV DX,0A403H      ; control word
    OUT DX,AL

    MOV BH,0AH
AGAIN:
    LEA SI,A1          ; load SI which contains the code for FIRE
    CALL DISP         ; display it
    CALL DELAY
    LEA SI,A2          ; load SI which contains the code for HELP
    CALL DISP         ; display it
    CALL DELAY
    DEC BH
    CMP BH,00H
    JE EXIT
    JMP AGAIN

DISP PROC
    MOV CX,04H        ; number of characters in the message
LOOP2:
    MOV BL,08H        ; no. of bits in a byte
    MOV AL,[SI]
NEXT:
    ROL AL,01H        ; rotate left and push the value
    MOV DX,0A401H     ; bit by bit
    OUT DX,AL
    PUSH AX
    MOV AL,0FFH
    MOV DX,0A402H     ; clock pulse given
    OUT DX,AL
    MOV AL,00H
    OUT DX,AL

    DEC BL
    POP AX
    JZ NEXT1         ; run the loop 8 times
    JMP NEXT

NEXT1:
    INC SI
    LOOP LOOP2

```

```
RET
DISP ENDP

DELAY PROC
    PUSH CX                ; save the contents on stack
    PUSH AX
    MOV CX,1FFFH
N1:    MOV AX,0FFFFH
N2:    DEC AX
        JNZ N2            ; loop for delay
        LOOP N1
    POP AX
    POP CX
    RET
DELAY ENDP

EXIT:
    MOV AH,4CH
    INT 21H
    END                    ; end of program
```

PROGRAM 5A**PROGRAM TO REVERSE A GIVEN STRING AND CHECK IF IT IS A PALINDROME**

```

.MODEL SMALL
.STACK 64
.DATA
    STR1 DB 40 DUP ('$')
    REV DB 40 DUP ('$')
    MSG1 DB 10,13,' ENTER THE STRING : $'
    MSG2 DB 10,13,' THE REVERSE OF THE STRING IS : $'
    PAL DB 10,13,' THIS STRING IS A PALINDROME!! $'
    NPAL DB 10,13,' THIS STRING IS NOT A PALINDROME!! $'

.CODE
    MOV AX,@DATA
    MOV DS,AX
    MOV ES,AX
    LEA DX,MSG1
    MOV AH,09H
    INT 21H
    XOR CX,CX
    LEA SI,STR1
LP1 :   MOV AH,01H
        INT 21H
        CMP AL,0DH
        JZ NEXT
        INC CX
        MOV [SI],AL
        INC SI
        JMP LP1
NEXT :  LEA DI,REV
        MOV BP,CX
LP2 :  DEC SI
        MOV AL,[SI]
        MOV [DI],AL
        INC DI
        LOOP LP2
        LEA DX,MSG2
        MOV AH,09H
        INT 21H
        LEA DX,REV
        MOV AH,09H
        INT 21H
        LEA DI,REV
        LEA SI,STR1
        MOV CX,BP
        REPE CMPSB
        JNZ FAIL
        LEA DX,PAL
        JMP MSG
FAIL :  LEA DX,NPAL
MSG :  MOV AH,09H
        INT 21H

```

```
MOV AH, 4CH  
INT 21H  
END
```

PROGRAM 5B**DISPLAY 12-CHARACTER MESSAGE ON 7-SEGMENT DISPLAY**

```

.MODEL SMALL
.STACK 64H
.DATA
    A1 DB 86H,88H,0F9H,8EH,8CH,0C7H,86H,89H
    LEN DW ($-A1)

.CODE
    MOV AX,@DATA           ; Load the address of DATA SEGEMENT into DS
    MOV DS,AX

    MOV AL,80H
    MOV DX,0A403H         ; control word
    OUT DX,AL

    MOV BX, 0AH           ; no. of times it has to rotate
LOOP1:
    MOV CX,LEN
    LEA SI,A1
AGAIN:
    MOV AL,[SI]
    CALL DISP             ; display the message in rotating fashion
    CALL DELAY           ; delay procedure
    INC SI
    LOOP AGAIN
    DEC BX
    JNZ LOOP1
    JMP EXIT

DISP PROC
    PUSH CX
    MOV CX,08H           ; no. of bits in a byte
LOOP2:
    MOV DX,0A401H        ; port B address
    ROL AL,01H
    OUT DX,AL           ; o/p bit by bit
    PUSH AX
    MOV AL,0FFH
    MOV DX,0A402H        ; clock pulse given through port C
    OUT DX,AL
    MOV AL,00H
    OUT DX,AL

    POP AX
    LOOP LOOP2          ; run until CX=0
    POP CX
    RET
DISP ENDP

```

```

DELAY PROC

```

```
    PUSH CX
    PUSH AX                ; save the contents on stack
    MOV CX,5FFFH
N1:   MOV AX,0FFFFH
N2:   DEC AX                ; main loop for delay
      JNZ N2
      LOOP N1
      POP AX
      POP CX
      RET
      DELAY ENDP
EXIT: MOV AH,4CH
      INT 21H
      END                ; end of program
```

PROGRAM 6A

PROGRAM TO READ 2 STRINGS AND CHECK IF THEY ARE EQUAL AND DISPLAY APPROPRIATE MESSAGES

```

.MODEL SMALL
.STACK 64
.DATA
    MSG1 DB 10,13," ENTER THE FIRST STRING : $"
    MSG2 DB 10,13," ENTER THE SECOND STRING : $"
    C1 DW ?
    C2 DW ?
    STR1 DB 80 DUP ('$')
    STR2 DB 80 DUP ('$')
    MSG3 DB 10,13," THE TWO STRINGS ARE EQUAL!! $"
    MSG4 DB 10,13," THE TWO STRINGS ARE NOT EQUAL!! $"

.CODE
    MOV AX,@DATA          ; Load the address of DATA SEGEMENT into DS
    MOV DS,AX
    MOV ES,AX             ; and into ES register
    LEA DX,MSG1           ; ask the user to enter the string STR1
    MOV AH,09H
    INT 21H
    MOV BX,0000H          ; initialize the count to zero
    LEA SI,STR1           ; initialize DI with starting address of
                        ; STR1

NEXT1 : MOV AH,01H
        INT 21H
        MOV [SI],AL       ; read character by character
        INC BX
        INC SI
        CMP AL,0DH
        JNZ NEXT1
        DEC BX
        MOV C1,BX         ; store count value into memory location C1
        LEA DX,MSG2       ; ask the user to enter the second string

STR2
        MOV AH,09H
        INT 21H
        MOV BX,0000H      ; initialize the count to zero
        LEA SI,STR2       ; initialize DI with starting address of
                        ; STR2

NEXT2 : MOV AH,01H
        INT 21H
        MOV [SI],AL       ; read character by character
        INC BX
        INC SI             ; increment count and SI
        CMP AL,0DH        ; check if the character is NEWLINE
character
        JNZ NEXT2
        DEC BX
        MOV C2,BX         ; store count value into memory location C2
        CMP BX,C1         ; compare the lengths of two strings
        JE CHECK
        JMP NOT_EQU

```

```
CHECK : MOV SI,OFFSET STR1 ; load SI and DI with the starting address
        MOV DI,OFFSET STR2 ; of STR1 and STR2
        MOV CX,BX          ; move count to CX register
        CLC
        CLD
        REPE CMPSB
        JNZ NOT_EQUAL
EQUAL : LEA DX,MSG3        ; if equal display the message ,as strings
                                ; are same
        JMP STOP
NOT_EQUAL : LEA DX,MSG4    ; display that two strings are not equal
STOP : MOV AH,09H
       INT 21H
       MOV AH,4CH
       INT 21H
       END
```

PROGRAM 6B**CONVERT FROM BINARY TO BCD AND DISPLAY ON SEVEN SEGMENT DISPLAY**

```

.model small
.stack 64h
.data
bin equ 0064h
table1 db 0c0h,0f9h,0a4h,0b0h,099h,92h,82h,0f8h,80h,98h
table2 db 4 dup(?)
table3 db 4 dup(?)
result dw ?
.code
    mov ax,@data
    mov ds,ax
    mov bx,bin
    mov ax,0000h
    mov cx,0000h
contin: cmp bx,0000h
        jz endprog
        dec bx
        mov al,cl
        add al,01h
        daa
        mov cl,al
        mov al,ch
        adc al,00h
        daa
        mov ch,al
        jmp contin

endprog:

        lea bx,table1
        lea si,table2
        lea di,table3
        mov cx,result
        and cl,0fh
        mov al,cl
        xlat
        mov [si],al
        mov [di+3],al
        inc si

        mov dx,result
        mov cl,04h
        shr dl,cl
        mov al,dl
        xlat
        mov [si],al
        mov [di+2],al
        inc si

        mov cx,result

```

```
    and ch,0fh
    mov al,ch
    xlat
    mov [si],al
    mov [di+1],al
    inc si

    mov dx,result
    mov cl,04h
    shr dh,cl
    mov al,dh
    xlat
    mov [si],al
    mov [di],al
    mov al,80h
    mov dx,0a403h
    out dx,al

    lea si,table2
    call disp
    call delay
    lea di,table3
    call disp
    call delay
    mov ah,4ch
    int 21h

disp proc
mov bh,04h
lp2: mov cl,08h
    mov bl,[si]
    lp3: rol bl,01h
    mov al,bl
    mov dx,0a401h
    out dx,al
    push bx
    mov al,0ffh
    mov dx,0a402h
    out dx,al
    mov al,00h
    out dx,al
    pop bx
    dec cl
    jnz lp3
    inc si
    dec bh
    jnz lp2
    ret
disp endp

delay proc
push ax
push cx
    mov ax,4ffffh
lp5: mov cx,0ffffh
lp6: dec cx
```

```
jnz lp6  
dec ax  
jnz lp5  
pop cx  
pop ax  
ret  
delay endp  
  
end
```

PROGRAM 7A

PROGRAM TO FIND THE FACTORIAL OF A GIVEN NUMBER RECURSIVELY

```

.MODEL SMALL
.STACK 64H
.DATA
    NUM DW 04H
    FACT DW ?
    N DW 25 DUP(0)
.CODE
START:
    MOV AX,@DATA    ; initialize DATA SEGMENT
    MOV DS,AX

    MOV AX,NUM      ; get the number
    MOV BX,01H     ; BX is used while multiplying
    MOV FACT,00H
    CALL FACTORIAL ; function call
    MOV FACT,BX    ; store the result in memory location
    MOV AH,4CH
    INT 21H       ; return to dos

FACTORIAL PROC
    CMP AX,01H    ; compare with 1 to end process
    JE NEXT      ; if equal return
    MOV CX,AX
    MUL BX       ; DO [AX]=[AX]*[BX]
    MOV BX,AX   ; [BX]=[AX]
    MOV AX,CX   ; get the initial value of AX from CX
    DEC AX
    CALL FACTORIAL ; recursive call
NEXT : RET     ; if AX==1 then return
FACTORIAL ENDP ; end of factorial
END START    ; end of program
END

```

PROGRAM 7B**STEPPER MOTOR-CLOCKWISE DIRECTION**

```
.MODEL SMALL
.STACK 64
.CODE
    MOV AL,080H           ; control word
    MOV DX,0A403H        ; address of control word
    OUT DX,AL
    MOV AL,088H          ; bit pattern is to be rotated
    MOV CX,100           ; no. of times it needs to rotate
LP1:
    MOV DX,0A402H        ; address of o/p port
    OUT DX,AL           ; o/p the bit pattern
    ROR AL,1            ; rotate left
    PUSH CX              ; push contents on stack
    PUSH AX
    CALL DELAY           ; before calling the subroutine for delay
    POP AX
    POP CX               ; pop back the count
    DEC CX
    JNZ LP1              ; jump if not zero
    MOV AH,4CH
    INT 21H

DELAY PROC
    MOV CX,0FFFH

LOOP1:
    MOV AX,01FFFH

LOOP2:
    DEC AX
    JNZ LOOP2
    DEC CX
    JNZ LOOP1
    RET

DELAY ENDP
END
```

PROGRAM 8A

PROGRAM TO READ THE NAME FROM KEYBOARD, CLEAR THE SCREEN AND DISPLAY THE NAME AT THE CENTRE OF THE SCREEN

```

.MODEL SMALL
.STACK 64
.DATA
    MSG DB " WHAT IS YOUR NAME ? $"
    NAM DB 40 DUP ('$')
.CODE
    MOV AX,@DATA
    MOV DS,AX           ; Load the address of DATA SEGEMENT into DS
    LEA DX,MSG         ; ask the user to enter the name
    MOV AH,09H
    INT 21H
    LEA SI,NAM         ; get the starting address of NAM
    MOV CX,00H        ; set the count to zero
LP1 :   MOV AH,01H     ; read the name character by character
    INT 21H
    CMP AL,0DH        ; compare if character is NEWLINE character
    JZ NEXT
    MOV [SI],AL       ; store in location NAM
    INC SI            ; increment SI and COUNT
    INC CX
    JMP LP1
NEXT :   MOV AH,0FH    ; to clear the screen
    INT 10H
    MOV AH,00H
    INT 10H
    MOV AH,02H        ; to move the cursor position
    MOV BH,00H        ; page number is 00
    MOV DX,0A1CH     ; ROW NO. in DH and COL NO. in DL
    INT 10H
    LEA DX,MSG        ; print both MSG and NAM
    MOV AH,09H
    INT 21H
    LEA DX,NAM
    MOV AH,09H
    INT 21H          ; return to DOS
    MOV AH,4CH
    INT 21H
END      ; end of program

```

PROGRAM 8B**STEPPER MOTOR-ANTICLOCKWISE DIRECTION**

```
.MODEL SMALL
.STACK 64
.CODE
    MOV AL,080H           ; control word
    MOV DX,0A403H        ; address of control word
    OUT DX,AL
    MOV AL,088H          ; bit pattern is to be rotated
    MOV CX,100           ; no. of times it needs to rotate
LP1:
    MOV DX,0A402H        ; address of o/p port
    OUT DX,AL            ; o/p the bit pattern
    ROL AL,1             ; rotate left
    PUSH CX              ; push contents on stack
    PUSH AX
    CALL DELAY           ; before calling the subroutine for delay
    POP AX
    POP CX               ; pop back the count
    DEC CX
    JNZ LP1             ; jump if not zero
    MOV AH,4CH
    INT 21H

DELAY PROC
    MOV CX,0FFFH

LOOP1:
    MOV AX,01FFFH

LOOP2:
    DEC AX
    JNZ LOOP2
    DEC CX
    JNZ LOOP1
    RET

DELAY ENDP
END
```

PROGRAM 9A

PROGRAM TO FIND THE VALUE OF NCR RECURSIVELY

```

.MODEL SMALL
.STACK 100H
.DATA
    N DW 7H
    R DW 5H
    NCR DW ?
    M DB 25 DUP(0)
    MSG1 DB 10,13," INVALID VALUES OF N AND R!! $"
.CODE
START :
    MOV AX,@DATA           ; initialize the data segment
    MOV DS,AX
    MOV AX,N
    MOV BX,R
    MOV NCR,00H
    MOV AX,R               ; get the value of R
    PUSH AX                ; push it on stack
    CMP AX,N               ; check if N>R
    JLE NEXT               ; if so continue
    LEA DX,MSG1             ; else print the values are invalid
    MOV AH,09H
    INT 21H
    JMP STOP                ; and return
NEXT : MOV AX,N            ; get the value of N
    PUSH AX                ; save it on stack
    CALL BINOMIAL           ; call the function to compute nCr
    MOV NCR,AX              ; store it in memory location
STOP :  MOV AH,4CH          ; return to DOS
    INT 21H
BINOMIAL PROC              ; start of function definition
    PUSH BP                ; get the frame pointer in BP
    MOV BP,SP
    MOV AX,[BP+6]           ; get r
    CMP AX,[BP+4]           ; compare with n
    JE NEXT1                ; if equal then return 1 as answer
    CMP AX,00H              ; do the same if r=0
    JG NEXT2                ; else continue c(n-1,r)+c(n-1,r-1)
NEXT1 : MOV AX,01H          ; returning 1
    JMP RETURN
NEXT2 : PUSH [BP+6]         ; save R on stack
    MOV CX,[BP+4]           ; get n

    DEC CX
    PUSH CX                 ; save on stack
    CALL BINOMIAL           ; call the function recursively
    PUSH AX                 ; store the result on stack
    MOV CX,[BP+6]           ; get r
    DEC CX
    PUSH CX                 ; save r-1 on stack
    MOV CX,[BP+4]           ; get N
    DEC CX                  ; find N-1
    PUSH CX                 ; save N-1

```

```
CALL BINOMIAL      ; call function recursively
POP BX             ; get the result into BX
ADD AX,BX          ; ADD  $C(n-1,r)$  and  $C(n-1,r-1)$ 
RETURN : POP BP    ; pop all the contents that are pushed
RET 4              ; before returning
BINOMIAL ENDP     ; end of function definition
END START         ; end of program
END
```

PROGRAM 9B**ROTATE THE STEPPER MOTOR IN LEFT AND RIGHT DIRECTIONS FOR N-
STEPS**

```

.MODEL SMALL
.STACK 64
.CODE
    MOV AL,080H           ; control word
    MOV DX,0A403H        ; address of control word
    OUT DX,AL
    MOV AL,088H          ; bit pattern is to be rotated
    MOV CX,100           ; no. of times it needs to rotate
LP1:
    MOV DX,0A402H        ; address of o/p port
    OUT DX,AL            ; o/p the bit pattern
    ROL AL,1             ; rotate left
    PUSH CX              ; push contents on stack
    PUSH AX
    CALL DELAY           ; before calling the subroutine for delay
    POP AX
    POP CX               ; pop back the count
    DEC CX
    JNZ LP1             ; jump if not zero
    MOV AH,4CH
    INT 21H

    MOV AL,088H          ; bit pattern is to be rotated
    MOV CX,100           ; no. of times it needs to rotate
LP2:
    MOV DX,0A402H        ; address of o/p port
    OUT DX,AL            ; o/p the bit pattern
    ROR AL,1             ; rotate left
    PUSH CX              ; push contents on stack
    PUSH AX
    CALL DELAY           ; before calling the subroutine for delay
    POP AX
    POP CX               ; pop back the count
    DEC CX
    JNZ LP2             ; jump if not zero
    MOV AH,4CH
    INT 21H

DELAY PROC
    MOV CX,0FFFH

LOOP1:
    MOV AX,01FFFH

LOOP2:
    DEC AX
    JNZ LOOP2
    DEC CX
    JNZ LOOP1

```

```
    RET  
DELAY ENDP  
END
```

PROGRAM 10A

PROGRAM TO FIND A GIVEN STRING IN THE TEXT

```

.MODEL SMALL
.STACK 64
.DATA
    STR1 DB 50 DUP ('$')
    SUBSTR1 DB 20 DUP ('$')
    LEN2 DW ?
    LEN1 DW ?
    MSG1 DB 10,13," ENTER THE TEXT : $"
    MSG2 DB 10,13," ENTER THE STRING FOR SEARCH : $"
    MSG3 DB 10,13," THE STRING SUCCESSFULLY FOUND !! $"
    MSG4 DB 10,13," STRING NOT FOUND !! $"

.CODE
    MOV AX,@DATA           ; initialize the data segment
    MOV DS,AX             ; ask the user to input the text
    MOV AH,09H
    LEA DX,MSG1
    INT 21H
    MOV CX,00H           ; initialize the count to zero
    LEA SI,STR1
    CALL READ             ; read the text and store it in
                        ; memory location STR

    MOV LEN1,CX           ; store the length of STR
    MOV AH,09H           ; ask the user for the substring
    LEA DX,MSG2
    INT 21H
    MOV CX,00H           ; read the SUBSTR and also store its
                        ; length

    LEA SI,SUBSTR1
    CALL READ
    MOV LEN2,CX
    CMP CX,LEN1          ; if len of SUBSTR > len STR then "NOT
                        ; FOUND"

    JG FAIL              ; else check char by char by Brute force
                        ; method

    LEA SI,STR1           ; set the starting of STR and SUBSTR into
    LEA DI,SUBSTR1        ; SI and DI respectively
    MOV CX,LEN2
    MOV DX,LEN1

LOOP1 : CMP CX,00H        ; while(STR[i]!='\0' || SUBSTR[i]!='\0')
    JE BREAK             ; if do the following else break
    CMP DX,00H
    JE BREAK
    MOV AL,[SI]          ; get the contents of SI into DI
    INC SI
    DEC DX
    CMP AL,[DI]
    JE NEXT              ; if(STR[i]!=SUBSTR[i])

    LEA DI,SUBSTR1       ; move DI point to start address of SUBSTR
    MOV CX,LEN2          ; initialise the count back to LEN2
    JMP LOOP1

NEXT : INC DI            ; increment the counter DI

```

```
        DEC CX                ; decrement the count tells how many chars
        JNZ LOOP1            ; are yet to be matched
BREAK :  CMP CX,00H          ; if all the characters in SUBSTR is
        JE SUCC              ; matched then CX=0 else CX!=0
FAIL  :  LEA DX,MSG4         ; print messages appropriately
        JMP STOP             ; failure message
SUCC  :  LEA DX,MSG3         ; success message
STOP  :  MOV AH,09H          ; function to print message on screen
        INT 21H
        MOV AH,4CH          ; return to DOS
        INT 21H
READ PROC                ; procedure to read the string
        PUSH AX              ; save AX on stack
LOOP2 :  MOV AH,01H          ; read char by char until '\n' char is
        ; read
        INT 21H
        CMP AL,0DH
        JE RETURN
        MOV [SI],AL
        INC SI
        INC CX                ; increment the count
        JMP LOOP2
RETURN :  POP AX              ; get back the contents from stack into AX
        RET
        READ ENDP
        END                  ; end of program
```

PROGRAM 10B

8x3 KEY PAD INTERFACE GET THE CHARACTER AND PRINT ITS ROW AND COLUMN

```

.MODEL SMALL
.STACK 64H
.DATA
    MSG1 DB 10,13,"PRESS ANY KEY FROM KEYPAD $"
    MSG2 DB 10,13,"THE ENTERED CHARACTER IS: $"
    KEY DB ?
    MSG3 DB 10,13,"THE ROW IS : $"
    MSG4 DB 10,13,"THE COLUMN IS : $"
    ROW DB 1
    COL DB 1
    ARR DB '0','1','2','3','4','5','6','7','8','9','+','-
', 'x', '/', '=', '%', 'C', 'E'

.CODE
    MOV AX,@DATA          ; initializes the data and extra segments
    MOV DS,AX

    MOV ES,AX
    MOV AL,90H
    MOV DX,0A403H        ; control word
    OUT DX,AL
    LEA DX,MSG1          ; ask the user to press a key
    MOV AH,09H
    INT 21H

LP1:  MOV BH,03H          ; NO. Of rows in the keypad
      MOV ROW,01H        ; initially search the first row
      MOV CL,00H        ; the key no. is initialized to 00
      MOV AL,01H        ; bit pattern to activate the first row
      MOV BL,AL

LOOP1: MOV DX,0A402H     ; activate a row
      OUT DX,AL          ; to see if the key from the row is
                        ; pressed
      IN AL,DX           ; get the contents of the row
      CMP AL,00H        ; if AL!=0 then key is pressed
      JNZ FOUND
      ADD CL,08H        ; else try to find in the next row
      MOV AL,BL         ; so, key no. will be greater than 8
      ROL AL,01         ; bit pattern to activate next row
      MOV BL,AL
      INC ROW
      DEC BH            ; no. of still to be searched
      JNZ LOOP1
      JMP LP1           ; loop untill a key is pressed

FOUND: ROR AL,01        ; if found search the column
      JC DISP
      INC CL
      INC COL

```

```

        JMP FOUND

DISP:   LEA DX,MSG2           ; print the key pressed
        MOV AH,09H
        INT 21H
        MOV CH,00H
        MOV SI,CX
        MOV DL,ARR[SI]      ; display the character
        MOV KEY,DL
        MOV AH,02H
        INT 21H
        MOV CL,ROW
        LEA DX,MSG3         ; display the row
        MOV AH,09H
        INT 21H
        CALL DIS_PLAY
        LEA DX,MSG4
        MOV AH,09H
        INT 21H
        MOV CL,COL         ; display the column
        CALL DIS_PLAY
        MOV AH,4CH
        INT 21H

DIS_PLAY PROC
        MOV DL,CL
        AND DL,0F0H        ; get the upper nibble
        MOV AL,DL
        ROR AL,01
        ROR AL,01
        ROR AL,01
        ROR AL,01
        ADD AL,30H         ; add 30H to get its ASCII value
        MOV DL,AL
        MOV AH,02H        ; display the upper nibble
        INT 21H
        MOV DL,CL
        AND DL,0FH        ; get the lower nibble
        ADD DL,30H        ; get its ASCII value
        MOV AH,02H
        INT 21H
        RET
DIS_PLAY ENDP
END

```

PROGRAM 11A**PROGRAM TO GENERATE N FIBONACCI NUMBERS GIVEN N**

```

.MODEL SMALL
.STACK 64H
.DATA
    NUM DW 10                ; 10 fibonacci numbers are to be
                            ; generated
    FIB DB 10 DUP (?)       ; FIB is the location where the numbers
                            ; are stored

.CODE
START :
    MOV AX,@DATA
    MOV DS,AX                ; initialize the data and extra segment
    MOV ES,AX
    LEA SI,FIB                ; get the address of FIB
    MOV AX,00H                ; FIB[0]=0,FIB[1]=1
    MOV BX,01H
    MOV CX,NUM
    SUB CX,02H                ; CX is decremented by 2
    CALL FIBO                 ; function call
    MOV AH,4CH                ; return to DOS
    INT 21H

FIBO PROC
    MOV [SI],AX                ; store the first two NOS.
    MOV [SI+1],BX              ; AX=FIB[n-2] and BX=FIB[N-1]
    ADD SI,02H                 ; increment SI by 2
LOOP1 : MOV DX,BX                ; DX=BX
        ADD BX,AX                ; BX=BX+AX
        MOV [SI],BX              ; FIB[n]=BX
        MOV AX,DX                ; AX=DX
        INC SI                    ; make SI point to next element
        LOOP LOOP1                ; decrement CX until n fibonacci numbers
                                ; are generated
    RET                        ; return to main
FIBO ENDP
END START                    ; end of program
END

```

PROGRAM 11B

ADD OR SUBTRACT TWO NUMBERS TAKEN AS INPUT FROM KEYPAD

```

.MODEL SMALL
.STACK 64H
.DATA
    ROW DB 01H
    COL DB 01
    MSG1 DB 10,13,"ENTER THE FIRST NUMBER : $"
    MSG3 DB 10,13,"ENTER THE SECOND NUMBER : $"
    MSG2 DB 10,13,"ENTER THE OPERATOR : $"
    ARR DB '0','1','2','3','4','5','6','7','8','9','+', '-
', '*', '/', '=', '%', 0,0,0, '='
    OPT DB ?
    RES DB ?
    MSG4 DB 10,13,"THE RESULT IS : $"
    OP1 DB ?
    OP2 DB ?

.CODE
    MOV AX,@DATA
    MOV DS,AX                ; initialize the data and extra
                             ; segment

    MOV ES,AX
    MOV OP1,0
    MOV OP2,0
    MOV OPT,0

    MOV AL,90H
    MOV DX,0A403H           ; control word
    OUT DX,AL
    LEA DX,MSG1             ; ask for operand 1
    MOV AH,09H
    INT 21H

    CALL READ               ; read the operand
    MOV CH,00               ; CX will have the index for the
                             ; element in array declared
    MOV SI,CX               ; arr[8]=8 & arr[11]='+'
    MOV DL,ARR[SI]
    MOV AH,02H
    INT 21H                 ; display the operand
    MOV OP1,CL
    MOV CX,00H
    MOV AH,01H
    INT 21H                 ; wait
    LEA DX,MSG2
    MOV AH,09H             ; ask for the operator
    INT 21H
    CALL READ               ; CX will have the index for the
                             ; element in array declared
    MOV DL,ARR[SI]         ; arr[8]=8 & arr[11]='+'
    MOV SI,CX
    MOV DL,ARR[SI]
    MOV OPT,DL
    MOV AH,02H             ; display the operand

```

```

INT 21H
MOV AH,01
INT 21H
MOV CX,00H
LEA DX,MSG3           ; ask for operand 2
MOV AH,09H
INT 21H
CALL READ             ; read the operand
MOV CH,00             ; CX will have the index for the
MOV SI,CX             ; element in array declared
MOV DL,ARR[SI]       ; arr[8]=8 & arr[11]='+'

MOV SI,CX
MOV DL,ARR[SI]
MOV OP2,CL
MOV AH,02H
INT 21H
MOV AH,OP1           ; get the 2 operands
MOV AL,OP2
CMP OPT,'+'         ; see if operator is '+' or '-'
JNE SUBT
ADD AH,AL
MOV RES,AH
JMP NEXT

SUBT:  SUB AH,AL
      MOV RES,AH

NEXT:  MOV CL,AH
      LEA DX,MSG4           ; display the result
      MOV AH,09H
      INT 21H
      CALL DISP             ; result may be any integer
      MOV AH,4CH           ; so convert and display it in 2 bytes
      INT 21H

READ PROC
      PUSH AX
      PUSH BX
      PUSH DX
LP1:   MOV BH,03H
      MOV ROW,01
      MOV CL,00H
      MOV AL,01H
      MOV BL,AL

LOOP1: MOV DX,0A402H
      OUT DX,AL
      MOV DX,0A400H
      IN AL,DX
      CMP AL,00H
      JNZ FOUND
      ADD CL,08H
      MOV AL,BL
      ROL AL,01
      MOV BL,AL
      INC ROW

```

```
        DEC BH
        JNZ LOOP1
        JMP LP1
FOUND:  ROR AL,01
        JC RETURN
        INC CL
        INC COL
        JMP FOUND
RETURN: POP DX
        POP BX
        POP AX
        RET
        READ ENDP
```

DISP PROC

```
        MOV DL,RES           ; save contents of AL
        MOV AH,02H          ; functional value
        AND DL,0F0H         ; get the value of upper nibble
        ROR DL,01
        ROR DL,01
        ROR DL,01
        ROR DL,01
        CMP DL,09
        JBE NEXT2
        ADD DL,7
NEXT2:  ADD DL,30H           ; display the upper nibble
        INT 21H

        MOV DL,RES
        AND DL,0FH          ; get the lower nibble
        MOV AH,02H
        CMP DL,9            ; get the ASCII value
        JBE NEXT1
        ADD DL,7H
NEXT1:  ADD DL,30H           ; print the lower nibble
        INT 21H
        RET
        DISP ENDP          ; end of procedure
END                                           ; end of program
```

PROGRAM 12A

PROGRAM TO READ THE CURRENT TIME FROM THE SYSTEM AND DISPLAY IT IN STANDARD FORMAT ON THE SCREEN

```
.MODEL SMALL
.STACK 64H
.DATA
    MSG DB 10,13,' THE CURRENT TIME IS : $'
    TIME DB '00:00:00$'
.CODE
    MOV AX,@DATA           ; initializing the data segment
    MOV DS,AX

    LEA BX,TIME           ; to get the values HRS,MINS,SECS into TIME
    LEA DX,MSG            ; print the message "the current time is"
    MOV AH,09H
    INT 21H
    CALL GET_TIME         ; call GET_TIME procedure
    LEA DX,TIME           ; print the time returned
    MOV AH,09H
    INT 21H
    MOV AH,4CH            ; return to DOS
    INT 21H

GET_TIME PROC            ; GET_TIME procedure definition
    MOV AH,2CH
    INT 21H
    MOV AL,CH             ; these registers will have the ASCII value
    CALL CONVERT
    MOV [BX],AX           ; move these to TIME array
    MOV AL,CL             ; CL :minutes(00-59)
    CALL CONVERT          ; convert minutes
    MOV [BX+3],AX        ; store this

    MOV AL,DH            ; DH seconds (00-59)
    CALL CONVERT          ; convert the seconds
    MOV [BX+6],AL        ; store seconds
    RET
GET_TIME ENDP           ; end of procedure

CONVERT PROC            ; function definition for conversion
    MOV AH,0             ; make higher byte 00H
    MOV DL,10            ; divide AX by 10
    DIV DL               ; after division quotient in AL and
                        ; remainder in AH

    OR AX,3030H         ; OR it with ASCII of '0' i.e. 30H
    RET
CONVERT ENDP

END                      ; end of program
```

PROGRAM 12B

DAC INTERFACE :: SINE WAVE

```

.MODEL SMALL
.STACK 64
.DATA
    ARRAY DB 80H,96H,0ABH,0C1H,0D2H,0E2H,0EEH,0F8H,0FEH
           DB 0FFH,0FEH,0F8H,0EEH,0E2H,0D2H,0C1H,0ABH,96H
           DB 80H,69H,54H,40H,2DH,1DH,11H,07H,01H
           DB 00H,01H,07H,11H,1DH,2DH,40H,54H,69H
    COUNT DW 36
.CODE
    MOV AX,@DATA           ; initialize data segment
    MOV DS,AX

    MOV DX,0A403H         ; move the control word into CWR
    MOV AL,80H
    OUT DX,AL

    BACK: MOV CX,COUNT     ; initialize the count
           LEA SI,ARRAY    ; SI points to the base address of array
LOOP1:  MOV DX,0A400H
           MOV AL,[SI]
           OUT DX,AL
           CALL DELAY
           INC SI           ; make SI point to next element in the array
           DEC CX          ; decrement count
           JZ BACK         ; CX!=0 continue
           MOV AH,01H      ; check for the key board interrupt
           INT 16H         ; if no key is pressed looping is continued
           JNZ STOP
           JMP LOOP1
STOP:   MOV AH,4CH         ; return to DOS
           INT 21H

    DELAY PROC
    PUSH AX
    MOV AX,0FFFFFFH       ; to bring in delay
LP1:    DEC AX
           JNZ LP1
    POP AX
    RET
    DELAY ENDP

END

```

PROGRAM 13A

PROGRAM TO SIMULATE A DECIMAL UP COUNTER FROM 00 TO 99

```

.MODEL SMALL
.STACK 64H
.DATA
    ROW DB 10
    COL DB 20
.CODE
    MOV AX,@DATA
    MOV DS,AX
    MOV SI,2           ; no. of times to count 00-99
    MOV BP,00         ; counter value
    MOV AH,0FH        ; clear screen
    INT 10H
    MOV AH,00H        ; clear screen
    INT 10H
LP1 :   MOV AH,02H     ; placing the cursor at the
        MOV BH,00H     ; coordinates given
        MOV DH,ROW
        MOV DL,COL
        INT 10H
        MOV AX,BP      ; get the counter value into AX
        CALL DISP      ; display the counter value
        CALL DELAY
        INC BP         ; increment counter
        CMP BP,100    ; if its <=99 continue
        JL CONT
        MOV BP,00     ; else make it 0 again
        DEC SI
CONT :
        JNZ LP1      ; go to next counter value
        MOV AH,4CH   ; return to DOS
        INT 21H

DISP PROC
    CALL CONVERT      ; to convert HEX to BCD
    MOV DL,AL
    MOV BL,AL
    AND DL,0F0H      ; get higher significant digit
    MOV CL,04
    ROR DL,CL        ; move it to lower significant digit position
    ADD DL,30H       ; get ASCII equivalent
    MOV AH,02H       ; print
    INT 21H
    MOV DL,BL
    AND DL,0FH       ; get lower significant digit
    ADD DL,30H       ; get ASCII value
    MOV AH,02
    INT 21H
    RET
DISP ENDP

```

```
CONVERT PROC
    CMP AL,00
    JNE LP2
    RET
LP2 :   MOV CL,AL           ; get no. into CL-counter
        MOV AL,00         ; will have the final decimal equivalent
LP3 :   INC AL             ; increment no.
        MOV BL,AL
        AND BL,0FH        ; get lower significant digit of AL
        CMP BL,09        ; see if its >9
        JLE CONTC
        AND AL,0F0H      ; if yes, clear it and
        ADD AL,10H       ; add 1 to higher significant digit
CONTC : DEC CL
        JNZ LP3
        RET
CONVERT ENDP

DELAY PROC
    MOV DX,0FFFFH
LP4 :   MOV BX,0FFCH
LP5 :   DEC BX
        JNZ LP5
        DEC DX
        JNZ LP4
        RET
DELAY ENDP

END
```

PROGRAM 13B

DAC INTERFACE :: HALF-RECTIFIED SINE WAVE

```

.MODEL SMALL
.STACK 64
.DATA
    ARRAY DB 80H,96H,0ABH,0C1H,0D2H,0E2H,0EEH,0F8H,0FEH
           DB 0FFH,0FEH,0F8H,0EEH,0E2H,0D2H,0C1H,0ABH,96H
           DB 80H,80H,80H,80H,80H,80H,80H,80H,80H
           DB 80H,80H,80H,80H,80H,80H,80H,80H,80H
    COUNT DW 36
.CODE
    MOV AX,@DATA           ; initialize data segment
    MOV DS,AX

    MOV DX,0A403H         ; move the control word into CWR
    MOV AL,80H
    OUT DX,AL

BACK: MOV CX,COUNT        ; initialize the count
      LEA SI,ARRAY        ; SI points to the base address of array
LOOP1: MOV DX,0A400H
      MOV AL,[SI]
      OUT DX,AL
      CALL DELAY
      INC SI              ; make SI point to next element in the array
      DEC CX              ; decrement count
      JZ BACK            ; CX!=0 continue
      MOV AH,01H         ; check for the key board interrupt
      INT 16H           ; if no key is pressed looping is continued
      JNZ STOP
      JMP LOOP1
STOP: MOV AH,4CH         ; return to DOS
      INT 21H

DELAY PROC
    PUSH AX
    MOV AX,0FFFFFFH      ; to bring in delay
LP1: DEC AX
      JNZ LP1
    POP AX
      RET
DELAY ENDP
END

```

PROGRAM 14A**PROGRAM TO PLACE THE CURSOR IN THE SPECIFIED POSITION**

```

.MODEL SMALL
.DATA
    ROW DB 10,13," ENTER THE ROW : $"
    COL DB 10,13," ENTER THE COLUMN : $"
    R DB ?
    C DB ?
    CHAR DB "_$"

.CODE
    MOV AX,@DATA           ; initialize the data segment
    MOV DS,AX
    MOV AH,0FH             ; clear the screen before
    INT 10H                ; placing the cursor
    MOV AH,00H
    INT 10H
    MOV AH,09H             ; ask user to input the row
    LEA DX,ROW
    INT 21H
    XOR AX,AX
    CALL READ              ; read the row
    MOV R,AL
    MOV AH,09H             ; read the column
    LEA DX,COL
    INT 21H
    XOR AX,AX
    CALL READ
    MOV C,AL
    MOV AH,02H             ; place the cursor at the
    MOV BH,00H             ; coordinates given
    MOV DH,R
    MOV DL,C
    INT 10H
    LEA DX,CHAR            ; to see the cursor position display
    MOV AH,09H            ; a '_' at that position
    INT 21H
    MOV AH,01H             ; wait for the user to
    INT 21H                ; see the cursor position
    MOV AH,4CH             ; return to DOS
    INT 21H

READ PROC
    MOV AH,01H             ; read a 2-digit no.
    INT 21H                ; first read a number
    SUB AL,30H             ; get its integer value
    MOV BL,10              ; it should be in the 10's place of the
    ; digit
    MUL BL                 ; so multiply by 10
    MOV DL,AL
    MOV AH,01H             ; read the digit in its units place
    INT 21H
    SUB AL,30H             ; get its integer value
    ADD AL,DL
    RET

```

```
READ ENDP  
END
```

PROGRAM 14B**DAC INTERFACE :: FULL-RECTIFIED SINE WAVE**

```

.MODEL SMALL
.STACK 64
.DATA
    ARRAY DB 80H,96H,0ABH,0C1H,0D2H,0E2H,0EEH,0F8H,0FEH
           DB 0FFH,0FEH,0F8H,0EEH,0E2H,0D2H,0C1H,0ABH,96H

    COUNT DW 18
.CODE
    MOV AX,@DATA           ; initialise data segment
    MOV DS,AX

    MOV DX,0A403H         ; move the control word into CWR
    MOV AL,80H
    OUT DX,AL

BACK: MOV CX,COUNT        ; initialize the count
      LEA SI,ARRAY        ; SI points to the base address of array
LOOP1: MOV DX,0A400H
      MOV AL,[SI]
      OUT DX,AL
      CALL DELAY
      INC SI               ; make SI point to next element in the array
      DEC CX               ; decrement count
      JZ BACK             ; CX!=0 continue
      MOV AH,01H          ; check for the key board interrupt
      INT 16H             ; if no key is pressed looping is continued
      JNZ STOP
      JMP LOOP1

STOP: MOV AH,4CH          ; return to DOS
      INT 21H

DELAY PROC
    PUSH AX
    MOV AX,0FFFFH        ; to bring in delay
LP1:  DEC AX
      JNZ LP1
    POP AX
    RET
DELAY ENDP
END

```

PROGRAM 15A

PROGRAM TO CREATE AND DELETE A FILE

```
.MODEL SMALL
.DATA
    F1 DB 40 DUP ('$')
    F2 DB 40 DUP ('$')
    FPTR DW ?
    MSG1 DB 10,13," ENTER THE FILENAME FOR INPUT : $"
    MSG2 DB 10,13," ENTER THE FILENAME FOR DELETION : $"
    MSG3 DB 10,13," FILE OPENED SUCCESSFULLY !! $"
    MSG4 DB 10,13," FILE OPERATION FAILED !! $"
    MSG5 DB 10,13," FILE DELETED SUCCESSFULLY !! $"
    MSG6 DB 10,13," FILE DELETION FAILED !! $"

.CODE
    MOV AX,@DATA
    MOV DS,AX
    MOV DX,OFFSET MSG1
    MOV AH,09H
    INT 21H
    MOV SI,OFFSET F1
    CALL READ
    MOV AH,3CH
    LEA DX,F1
    MOV CL,00H
    INT 21H
    MOV FPTR,AX
    ;CMP AX,00H
    JC FAIL
    MOV AH,09H
    MOV DX,OFFSET MSG3
    INT 21H
    MOV AH,3EH
    MOV BX,FPTR
    INT 21H
    JMP NEXT
FAIL : MOV AH,09H
    LEA DX,MSG4
    INT 21H
NEXT : MOV AH,09H
    LEA DX,MSG2
    INT 21H
    LEA SI,F2
    CALL READ
    MOV AL,00H
    MOV AH,41H
    LEA DX,F2
    INT 21H
    JNC SUCC
    MOV AH,09H
    LEA DX,MSG6
    INT 21H
    JMP STOP
SUCC : LEA DX,MSG5
    MOV AH,09H
```

```
        INT 21H
STOP   : MOV AH,4CH
        INT 21H

READ PROC
        PUSH AX
LOOP2  : MOV AH,01H
        INT 21H
        CMP AL,0DH
        JZ RETURN
        MOV [SI],AL
        INC SI
        JMP LOOP2
RETURN : POP AX
        RET
        READ ENDP
END
```

PROGRAM 15B**ELEVATOR PROGRAM**

```

.MODEL SMALL
.STACK 64
.DATA
.CODE
    MOV AX,@DATA           ; initialise DS
    MOV DS,AX
    MOV DX,A403H          ; move the control word into CWR
    MOV AL,82H
    OUT DX,AL
    MOV DX,A400H          ; clear the contents of port A
    MOV AL,0F0H
    OUT DX,AL
    XOR CX,CX             ; clear CX
LOOP1:  MOV DX,A401H
        IN AL,DX          ; read the port B contents
        AND AL,0FH        ; if a key is not pressed loop back
        CMP AL,0FH
        JZ LOOP1          ; else GOTO next
NEXT:   SHR AL,01
        JNC GFLR          ; find which floor key is pressed
        SHR AL,01
        JNC FFLR
        SHR AL,01
        JNC SFLR
        SHR AL,01
        JNC TFLR
GFLR:   MOV CL,01H        ; if ground floor move only once
        JMP MOVE
FFLR:   MOV CL,4
        JMP MOVE          ; if the first floor 4 LEDs SHD
                          ; light
SFLR:   MOV CL,7
        JMP MOVE          ; if second floor 7 LEDs
TFLR:   MOV CL,10
        JMP MOVE          ; if the third floor 10 LEDs SHD
                          ; light
        JMP MOVE
MOVE:   MOV AL,00
        MOV BL,CL         ; save count
LP1:    MOV DX,A400H
        OUT DX,AL        ; moving up
        CALL DELAY
        INC AL
        DEC CL
        JNZ LP1
        MOV CL,BL
        DEC AL
LP2:    MOV DX,A400H      ; moving down
        OUT DX,AL
        CALL DELAY
        DEC AL
        DEC CL
        JNZ LP2

```

```
        MOV AH,4CH                ; return to DOS
        INT 21H
DELAY PROC
        PUSH AX
        PUSH CX
        MOV AX,0CFFH
LP3:    MOV CX,0FFFH
LP4:    DEC CX
        JNZ LP4
        DEC AX
        JNZ LP3
        POP CX
        POP AX
        RET
DELAY ENDP
END                ; end of program
```