MICROPROCESSORS LABORATORY

(Common to CSE & ISE)

Subject Code : 10CSL48	**I.A. Marks : 25**
Hours/Week : 03	**Exam Hours: 03**
Total Hours : 42	**Exam Marks: 50**

Notes:

- **Develop and execute the following programs using 8086 Assembly Language. Any suitable assembler like MASM, TASM etc may be used.**
- **Program should have suitable comments.**
- **The board layout and the circuit diagram of the interface are to be provided to the student during the examination.**

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

   b) Read the status of eight input bits from the Logic Controller Interface and display 'FF' if it is the parity of the input read is even; otherwise display 00.

2. a) Write two ALP modules stored in two different files; one module is to read a character from the keyboard and the other one is to display a character. 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.

   b) Implement a BCD Up-Down Counter on the Logic Controller Interface.

3. a) Sort a given set of 'n' numbers in ascending order using the Bubble Sort algorithm.

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

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

   b) 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 (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

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

   b) 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. (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

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

   b) 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.

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

   b) Scan a 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.

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

   b) Drive a Stepper Motor interface to rotate the motor in specified direction (clockwise or counter-clockwise) by N steps (Direction and N are specified by the examiner). Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).

9. a) Read the current time from the system and display it in the standard format on the screen.

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

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

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

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

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

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

    b) 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 (approximately), and then come down to ground floor and stop. If some requests occur during going up or coming down they should be ignored.

Note: In the examination *each* student picks one question from the lot of *all* 12 questions.

SAMPLE PROGRAMS

1) PROGRAM TO ADD 2 NUMBERS

```
code segment
        assume cs: code
                 start: mov al,3                  ;read 2 numbers
                        mov bl,2

                        add al,bl                 ;compute sum

                        mov cl,al                 ;output result
                        int 03h
code ends
                  end start
```

2) PROGRAM TO ADD TWO 8 BIT NUMBERS PRESENT IN MEMORY

```
data segment
        x db 3
        y db 2
        z db ?
data ends

code segment
        assume cs: code, ds: data
                 start: mov ax,data               ;data segment initialization
                        mov ds,ax

                        mov al,x                  ;read 2 numbers
                        mov bl,y

                        add al,bl                 ;compute sum ie al=al+bl

                        mov z,al                  ;output result
                termi: int 03h

code ends
                  end start
```

3) PROGRAM TO SUBTRACT y FROM x

```
data segment
        x db 3
        y db 2
        z db ?
data ends

code segment
        assume cs: code, ds: data
                 start: mov ax,data           ;data segment initialization
                        mov ds,ax

                        mov al,x               ;read 2 numbers
                        mov bl,y
                        sub al,bl              ;compute difference
                        mov z,al               ;output result
                termi: int 03h
code ends
                  end start
```

4) PROGRAM TO ADD 3 NUMBERS PRESENT IN MEMORY

```
data segment
        x db 3
        y db 2
        z db 1
        p db ?
data ends

code segment
        assume cs: code, ds: data
                 start: mov ax,data            ;data segment initialization
                        mov ds,ax

                        mov al,x               ;read 2 numbers
                        mov bl,y
                        mov cl,z

                        add al,bl              ;compute sum
                        add al,cl

                        mov p,al               ;output result
                 termi:int 03h
code ends
                 end start
```

5) PROGRAM TO FIND AVERAGE OF 2 NUMBERS PRESENT IN MEMORY

```
data segment
      x db 3
      y db 2
      z db ?
data ends

code segment
      assume cs:code, ds: data
             start: mov ax,data         ;data segment initialization
                    mov ds,ax

                    mov al,x             ;read 2 numbers
                    mov bl,y

                    add al,bl            ;compute sum
                    shr al,1              ;compute sum/2

                    mov z,al             ;output result
              termi:int 03h
code ends
              end start
```

6) PROGRAM TO ADD TWO 16 BIT NUMBERS PRESENT IN MEMORY

```
data segment
      x dw 3
      y dw 2
      z dw ?
data ends

code segment
      assume cs:code, ds:data
             start: mov ax,data               ;data segment initialization
                    mov ds,ax

                    mov ax,x
                    mov bx,y

                    add ax,bx

                    mov z,ax
               termi:int 03h
code ends
              end start
```

7) PROGRAM TO MULTIPLY 2 NUMBERS PRESENT IN MEMORY

```
data segment
        x dw 3
        y db 2
        z dw ?
data ends

code segment
        assume cs:code, ds:data
                start: mov ax,data          ;data segment initialization
                       mov ds,ax

                       mov ax,x              ;read 2 numbers
                       mov bl,y
                       mul bl                ;compute product(ax=ax*bl)

                       mov z,ax              ;output result
                 termi:int 03h
code ends
        end start
```

8) PROGRAM TO PERFORM DIVISION

```
data segment
        x dw 7
        y db 3
        r db ?
        q db ?
data ends

code segment
        assume cs: code, ds: data
                start: mov ax,data          ;data segment initialization
                       mov ds,ax

                       mov ax,x               ;read 2 numbers
                       mov bl,y
                       div bl                 ;perform division (ax=ax/bl)

                       mov q,al              ;after division al-> quotient ah->remainder
                       mov e,ah
                termi:int 03h
code ends
                end start
```

9) PROGRAM TO DISPLAY A MESSAGE ON THE SCREEN

```
data segment
        msg db “WELCOME TO MICROPROCESSOR LAB $”
data ends

code segment
      assume cs:code, ds: data
         start: mov ax,data              ;data segment initialization
                mov ds,ax

                lea dx,msg                ;dx contains offset address of message to be displayed
                mov ah,09h               ;fun(09h)/int 21h to display a message
                int 21h

         termi:mov ah,4ch                ;fun(4ch)/int 21h to terminate program normally
                int 21h

code ends
        end start
```

10) PROGRAM TO DISPLAY A CHARACTER ON THE SCREEN

```
data segment
        msg db “DISPLAYED CHARACTER: $”
data ends

code segment
    assume cs:code, ds:data
         start: mov ax,data                     ;data segment initialization
                mov ds,ax

                lea dx,msg
                mov ah,09h                      ;fun(09h)/int 21h to display a message
                int 21h

                mov dl,’z’
                mov ah,02h                       ;fun(02h)/int 21h to display a character
                int 21h

         termi:mov ah,4ch                ;fun(4ch)/int 21h to terminate program normally
                 int 21h

code ends
            end start
```

11) PROGRAM TO READ AND DISPLAY A CHARACTER ON THE SCREEN

```
data segment
        msg db "ENTER ONE CHARACTER: $"
data ends

code segment
    assume cs:code, ds:data
         start: mov ax,data                  ;data segment initialization
                mov ds,ax

                lea dx,msg
                mov ah,09h                    ;fun(09h)/int 21h to display a message
                int 21h

                mov ah,01h              ;fun(01h)/int 21h to read a character
                int 21h

                mov dl,al
                mov ah,02h                     ;fun(02h)/int 21h to display a character
                int 21h
          termi:mov ah,4ch              ;fun(4ch)/int 21h to terminate program normally
                 int 21h
code ends
            end start
```

12) PROGRAM TO SET CURSOR POSITION

```
code segment
        assume cs:code
         start: mov ah,00h              ;fun(03h)/int 10h to clear screen
                mov al,03h
                int 10h

                mov ah,02h              ;fun(02h)/int 10h to set cursor position
                mov dh,10               ;dh-> row num(00-18h)
                mov dl,10               ;dl->col num(00-49h)
                int 10h

                mov ah,01h              ;fun(01h)/int 21h to read a character III getch()
                int 21h
         termi:mov ah,4ch              ;fun(4ch)/int 21h to terminate program normally
                int 21h
code ends
        end start
```

13) PROGRAM TO DISPLAY A MESSAGE USING MACRO

```
disp macro msg
        lea dx,msg              ;dx contains offset address of message to be displayed
        mov ah,09h             ;fun(09h)/int 21h to display a message
        int 21h
endsm

data segment
        msg db “WELCOME TO MICROPROCESSOR LAB $”
data ends

code segment
     assume cs:code, ds: data
         start: mov ax,data
                mov ds,ax

                disp msg

                mov ah,4ch                ;fun(4ch)/int 21h to terminate program normally
                int 21h
code ends
           end start
```

14) PROGRAM TO SET CURSOR POSITION USING MACRO

```
clear macro
        mov ah,00h                              ;fun(03h)/int 10h to clear screen
        mov al,03h
        int 10h
endm

setcur macro
        mov ah,02h                              ;fun(02h)/int 10h to set cursor position
        mov dh,10
        mov dl,10
        int 10h
endm

read macro
        mov ah,01h                      ;fun(01h)/int 21h to read a character III getch()
        int 21h
endm

code segment
        assume cs:code
         start: clear

                setcur

                read

                mov ah,4ch                  ;fun(4ch)/int 21h to terminate program normally
                int 21h

code ends
           end start
```

15) PROGRAM TO COPY A BLOCK OF DATA FROM ONE MEMORY LOCATION TO ANOTHER

```
data segment
        blk1  db 23h,24h,25h,26h
        blk2  db 5 dup(?)
        count db 04h
data ends

code segment
     assume cs:code, ds:data
         start: mov ax,data  ;data segment initialization
                mov ds,ax

                 mov cx, count
                lea si, blk1
                lea di, blk2

          back:mov al,[si]
                mov [di],al
                inc si
                inc di
                dec cx
                jnz back

         termi: int 03h

 code ends
         end start
```

16) PROGRAM TO FIND FACTORIAL OF NUMBER

```
DATA SEGMENT
        N DW 4
        RES DW 0
DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
              START: MOV AX,DATA                ;DATA SEGMENT INITIALIZATION
                        MOV DS,AX

                        MOV AX, N
                        MOV CX, AX
                        DEC CX
        `       BACK: MUL CX
                        DEC CX
                        JNZ  BACK               ; RESULTS STORED IN AX
                        MOV RES, AX             ; TO STORE THE RESULT IN RES

                TERMI: INT 03H
CODE ENDS
                  END START
```

SOFTWARE LAB PROGRAMS

1A) SEARCH A KEY ELEMENT IN A LIST OF 'N' 16-BIT NUMBERS USING THE BINARY SEARCH ALGORITHM.

```
data segment
        array dw 1023h,1024h,1025h,1026h,1027h
        key dw 1025h
        n dw 05h
        pos dw ?
        r dw ?
data ends

code segment
    assume cs:code, ds:data
        start:  mov ax,data  ;data segment initialization
                mov ds,ax

                lea si,array
                mov ax,key    ;initialize variables key=>ax, low=>bx ,high=>dx ,mid=>si  & tempReg=>cx
                mov bx,00h
                mov dx,n
                dec dx

         back:  cmp bx,dx      ;if(low>=high) goto unsuccess(0000h) else continue
                ja unsuccess

                mov cx,bx
                add cx,dx      ;mid=(low+high)/2
                shr cx,01h

                mov si,cx
                add si,si

                cmp ax,[si]
                jz success     ;if(key=a[mid]) then key element found goto success(0ffffh)
                ja shalf      ;if(key>a[mid]) then search second-half else search first-half

         fhalf: mov dx,cx
                dec dx         ;high=mid-1
                jmp back
         shalf: mov bx,cx
                inc bx         ;low=mid+1
                jmp back
```

```
   unsuccess: mov r,0000h
                 jmp termi

      success: mov r,0ffffh
                mov pos,si       ;move key element position into ‘pos’
                jmp termi

          termi: int 03h
code ends
       end start
```

1A) SEARCH A KEY ELEMENT IN A LIST OF 'N' 8-BIT NUMBERS USING THE BINARY SEARCH ALGORITHM (EXTRA PROGRAM).

```
data segment
        array db 23h,24h,25h,26h,27h
        key db 26h
        n db 05h
        pos dw ?
        r db ?
data ends

code segment
     assume cs:code, ds:data
        start:  mov ax,data  ;data segment initialization
                mov ds,ax

                lea si,array
                mov al,key       ;initialize variables key=>al, low=>bl, high=>dl ,mid=>si  & tempReg=>cl
                mov bl,00h
                mov dl,n
                dec dl

         back:  cmp bl,dl        ;if(low>=high) goto unsuccess(00h) else continue
                ja unsuccess

                mov cl,bl
                add cl,dl         ;mid=(low+high)/2
                shr cl,01h

                mov si,cl
                cmp al,[si]
                jz success        ;if(key=a[mid]) then key Element found goto success(0ffh)
                ja shalf          ;if(key>a[mid]) then search second-half else search first-half

         fhalf: mov dl,cl
                dec dl            ;high=mid-1
                jmp back

         shalf: mov bl,cl
                inc bl           ;low=mid+1
                jmp back
```

```
    unsuccess: mov r,00h
                jmp termi

       success: mov r,0ffh
               mov pos,si      ;move key element position into ‘pos’
               jmp termi

          termi:int 03h

code ends
         end start
```

2A) WRITE TWO ALP MODULES STORED IN TWO DIFFERENT FILES; ONE MODULE IS TO READ A CHARACTER FROM THE KEYBOARD AND THE OTHER ONE IS TO DISPLAY A CHARACTER. 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.

```
;MACRO TO READ A STRING IN FILE1.ASM
read macro
        mov ah,01h
        int 21h
endm

;MACRO TO WRITE A STRING IN FILE2.ASM
write macro
        mov dl,al
        mov ah,02h
        int 21h
endm

;MAIN PROGRAM
clear  macro
        mov ah,00h
        mov al,03h
        int 10h
endm

disp macro msg
        mov ah,09h
        lea dx,msg
        int 21h
endm

data segment
        str db 20 dup('$')
        newline db 0ah,0dh,'$'
        include c:\masm\FILE1.asm
        include c:\masm\FILE2.asm
data ends
```

```
code segment
        assume cs:code, ds:data
          start:mov ax,data
                mov ds,ax

                clear

                lea si,str
        label1:read
                mov [si],al
                inc si
                cmp al,0dh              ;if read character==carriage return then stop reading string
                jnz label1

                mov cx,si
                disp newline

                lea si,str
       label2: mov al,[si]
                write
                inc si
                dec cx
                jnz label2

        termi: mov ah,4ch
                int 21h

code ends
        end start
```

3A) SORT A GIVEN SET OF 'N' 8 BIT NUMBERS IN ASCENDING ORDER USING THE BUBBLE SORT ALGORITHM.

```
data segment
        array db 80h,60h,40h,30h,10h      ;define 5 elements in unordered list
        count db 05h                      ;number of elements
data ends

code segment
        assume cs:code, ds:data
          start: mov ax,data          ;data segment initialization
                 mov ds,ax

                 mov bl,count          ;bl->number of iterations
                 dec bl

           iter: lea si,array
                 mov cl,04h            ;cl=>number of comparisons

          cmpr:  mov al,[si]           ;al=>temp register
                 inc si
                 cmp al,[si]           ;if(a[i]>a[i+1]) then swap them else do nothing
                 jc skip               ;replace jc by jnc for sorting in descending order

         swap: xchg al,[si]           ;swap a[i] and a[i+1]
                 xchg al,[si-1]

          skip: dec cl
                 jnz cmpr
                 dec bl
                 jnz iter

          termi: int 03h

code ends
        end start
```

3A) SORT A GIVEN SET OF 'N' 16 BIT NUMBERS IN ASCENDING ORDER USING THE BUBBLE SORT ALGORITHM (EXTRA PROGRAM).

```
data segment
         array dw 1080h,1060h,1040h,1030h,1010h      ;define 5 elements in unordered list
         count db 05h                                ;number of elements
data ends

code segment
      assume cs:code, ds:data
           start:mov ax,data          ;data segment initialization
                 mov ds,ax

                 mov bl,count         ;bl->number of iterations
                 dec bl

            iter:lea si,array
                 mov cl,04h             ;cl=>number of comparisons

          cmpr:mov ax,[si]              ;al=>temp register
                 add si,02h
                 cmp ax,[si]            ;if(a[i]>a[i+1]) then swap them  else do nothing
                 jc skip                ;replace jc by jnc for sorting in ascending order

          swap:xchg ax,[si]            ;swap a[i] and a[i+1]
                 xchg ax,[si-2]

           skip: dec cl
                 jnz cmpr
                 dec bl
                 jnz iter

          termi: int 03h
code ends
         end start
```

4A) READ AN ALPHANUMERIC CHARACTER AND DISPLAY ITS EQUIVALENT ASCII CODE AT THE CENTER OF THE SCREEN.

```
clear macro
        mov ah,00h
        mov al,03h
        int 10h
endm

disp macro msg
        mov ah,09h
        lea dx,m1
        int 21h
endm

setcur macro
        mov ah,02h
        mov dx,0c29h
        int 10h
endm

read macro
        mov ah,01h
        int 21h
endm
```

```
data segment
        msg db  “enter one alphanumeric character:$”
data ends

code segment
        assume  ds:data,  cs:code
          start:mov ax,data            ; data segment initialization
                mov ds,ax

                clear

                disp msg

                read

                setcur

                mov bl,al       ;to make 2 copies of same data & manipulate 1st digit in AL & 2nd digit in BL

                and al,0f0h             ;mask lower nibble
                mov cl,04h
                shr al,cl              ;shift higher nibble => lower nibble
                call show

                mov al,bl
                and al,0fh             ;mask higher nbble
                call show

                mov ah,4ch             ;fun(4c) to terminate program normally
                int 21h

              show proc near        `        ;procedure to display a character
                        cmp al,0ah             ;if digit<0aH goto skip & add 30 else add 37(7+30)
                        jl skip
                        add al,07h
                   skip:add al,30h             ;to obtain ascii character
                        mov ah,02h             ;fun(02)/int 21h to display a char(digit's ascii equivalent char)
                        mov dl,al
                        int 21h
                        ret
             show endp

code ends
        end start
```

5A) REVERSE A GIVEN STRING AND CHECK WHETHER IT IS A PALINDROME OR NOT (EXTRA PROGRAM).

```
disp macro msg
        lea dx,msg
        mov ah,09h
        int 21h
endm

data segment
        str1db "MADAM$"
        len dw ($-str1)
        str2 db 20 dup('$')
        m1 db 10,13,"Success$"
        m2 db 10,13,"Failure$"
data ends

code segment
        assume cs:code, ds:data
          start:mov ax,data               ;data and extra segment initialization
                mov ds,ax
                mov es,ax

                mov cx,len                 ;len=>CX
                lea si,str1
                lea di,str2
                mov si,cx                  ;to point to last element
                dec si

          next:std                        ;set d=1 to process string in auto-decrement mode
                lodsb                     ;load memory content pointed by SI into AL & decrement SI
                cld                        ;clear d=0 to process string in auto-increment mode
                stosb                      ;store AL content into memory pointed by DI & increment DI
                loop next                  ;loop until CX=0

                lea si,str1
                lea di,str2
                mov cx,len
                repe cmpsb                ;repeatedly compare byte in data segment pointed by SI with byte in
                jz success                 ;extra segment pointed by DI, also, increment SI & DI
                                          ;if all bytes are equal then goto success else goto unsucces
```

```
   unsuccess:disp m2                ;display message m2
             jmp termi

     success:disp m1                 ;display message m1
             jmp termi

       termi:mov ah,04ch             ;fun(4c) to terminate program
            int 21h

code ends
     end start
```

6A) READ TWO STRINGS, STORE THEM IN LOCATIONS STR1 AND STR2. CHECK WHETHER THEY ARE EQUAL OR NOT AND DISPLAY APPROPRIATE MESSAGES. ALSO DISPLAY THE LENGTH OF THE STORED STRINGS (EXTRA PROGRAM).

```
gets macro str
        lea dx,str
        mov ah,0ah
        int 21h
endm

disp macro msg
        lea dx,msg
        mov ah,09h
        int 21h
endm

data segment
        m1 db 10,13,"enter string 1: $"
        m2 db 10,13,"enter string 2: $"
        m5 db 10,13,"strings are equal $"
        m6 db 10,13,"strings are not equal $"
        str1 db 10 dup('$')
        str2 db 10 dup('$')
data ends
```

```
code segment
  assume cs:code, ds:data
   start:mov ax,data
         mov ds,ax
         mov es,ax

         disp m1
         gets str1
         disp m2
         gets str2

         mov cl,[str1+1]
         mov ch,[str2+1]
         cmp ch,cl
         jnz label1

         mov ch,00h
         lea si,str1+2
         lea di,str2+2
         repe cmpsb
         jnz label1

         disp m5
         jmp termi

  label1:disp m6
         jmp termi

 termi: mov ah,4ch
         int 21h
code ends
      end start
```

6A) READ TWO STRINGS, STORE THEM IN LOCATIONS STR1 AND STR2. CHECK WHETHER THEY ARE EQUAL OR NOT AND DISPLAY APPROPRIATE MESSAGES. ALSO DISPLAY THE LENGTH OF THE STORED STRINGS.

```
gets macro str
        lea dx,str
        mov ah,0ah
        int 21h
endm

disp macro msg
        lea dx,msg
        mov ah,09h
        int 21h
endm

data segment
        m1 db 10,13,"enter string 1:$"
        m2 db 10,13,"enter string 2:$"
        m3 db 10,13,"length of string 1:$"
        m4 db 10,13,"length of string 2:$"
        m5 db 10,13,"strings are equal $"
        m6 db 10,13,"string are not equal $"
        str1 db 10 dup('$')
        str2 db 10 dup('$')
data ends
```

```
code segment
    assume cs:code, ds:data ,es:data
       start: mov ax,data                  ;data & extra segment initialization
              mov ds,ax
              mov es,ax

              disp m1                       ;display message m1
              gets str1
              disp m3                      ;display message m3
              mov dl,[str1+1]              ;str1+1 contains length of string
              call len                             ;call len to get length of str1

              disp m2                       ;display message m2
              gets str2
              disp m4                       ;display message m4
              mov dl,[str2+1]              ;str2+1 contains length of string
              call len                      ;call len to get length of str1

              mov cl,[str1+1]
              mov ch,[str2+1]
              cmp ch, cl               ;if length of both strings are unequal then gotounsuccess message
              jnz label1

              mov ch,00h
              lea si,str1+2                 ;str1+2 contains offset address of str1
              lea di,str2+2                 ;str2+2 contains offset address of str2

              repe cmpsb             ;compare byte in data segment with byte in
              jnz label1             ;extra segment and increment both SI & DI
              disp m5                      ;if equal display success message
              jmp termi
      label1:disp m6
              jmp termi
      termi: mov ah,4ch               ;fun(4c) to terminate program
              int 21h

              len proc near
                     add dl,30h             ;to obtaim ASCII value
                     mov ah,02h
                     int 21h
                     ret
              len endp
code ends
      end start
```

7A) READ YOUR NAME FROM THE KEYBOARD AND DISPLAY IT AT A SPECIFIED LOCATION ON THE SCREEN AFTER THE MESSAGE "WHAT IS YOUR NAME?" YOU MUST CLEAR THE ENTIRE SCREEN BEFORE DISPLAY.

```
disp macro msg                  ;macro to display message on screen
        lea dx,msg              ;dx contains offset address of string to be displayed
        mov ah,09h              ;function(09) to display string on screen
        int 21h
endm

gets macro str
        lea dx,str          ;dx contains offset address of string to be read into buffer from keyboard
        mov ah,0ah          ;function(0ah)/int 21h to read string from keyboard into buffer
        int 21h
endm

clear macro
        mov ah,00h          ;function(00h)/int 10h to clear screen
        mov al,03h
        int 10h
endm

setcur macro
        mov ah,02h          ;function(02h)/int 10h to set cursor position to centre of screen
        mov dx,0c19h
        int 10h
endm

read macro
        mov ah,01h          ;function(01h)/int 21h to get character
        int 21h
endm

data segment
        m1 db 10,13,"enter the name:$"
        str db 20 dup('$')
        m2 db "what is your name?:$"
data ends
```

```
code segment
    assume ds:data, cs:code
          start:mov ax,data                    ;data segment initialization
                mov ds,ax

                disp m1                        ;display message m1

                gets str                       ;read string

                clear                          ;clear screen

                setcur                          ;set cursor position

                disp m2                        ;display message m2
                disp str+2                     ;display name(str+2 contains offset address of actual string)

                read                            ;similar to getch()

                clear                         ;clear screen

        termi: mov ah,4ch                     ;function(4ch)/int 21h to terminate program
                int 21h
code ends
   end start
```

8A) COMPUTE NCR USING RECURSIVE PROCEDURE. ASSUME THAT 'N' AND 'R' ARE NON-NEGATIVE INTEGERS.

```
data segment
        n db 05h
        r db 05h
        res db 00h
data ends

code segment
  assume cs:code, ds:data
        start: mov ax,data
               mov ds,ax
               mov al,n
               mov bl,r
               call ncr
        termi: mov ah,4ch
               int 21h

              ncr proc near
                      cmp bl,00                ;if r=0 then res=1
                      je exit1

                      cmp bl,01               ;if r=1 then res=n
                      je exit2

                      cmp bl,al                ;if r=n then res=1
                      je exit1

                      dec al
                      cmp bl,al                ;if r=n-1 then res=n
                      je exit3

                      push ax                 ;else $^{n}C_{r}=^{n-1}C_{r}+ ^{n-1}C_{r-1}$
                      push bx
                      call ncr
                      pop bx
                      pop ax
```

```
                              dec bl
                              push ax
                              push bx
                              call ncr
                              pop bx
                              pop ax
                              ret

                         exit1:inc res
                                  ret

                        exit3:inc res

                        exit2:add res,al
                                 ret
                  ncr endp
code ends
          end start
```

9A) READ THE CURRENT TIME FROM THE SYSTEM AND DISPLAY IT IN THE STANDARD FORMAT ON THE SCREEN.

```
disp macro msg                  ;macro to display message on screen
     lea dx,msg                 ;dx contains offset address of string to be displayed
     mov ah,09h                 ;function(09)/int 21h  to display string on screen
     int 21h
endm

data segment
      msg db “current time is:$”
data ends

code segment
   assume ds:data, cs:code
      start:mov ax,data         ;data segment initialization
            mov ds,ax

            disp msg

            mov ah,2ch     ;function(2c)/int 21h to get system time CH=hr, CL=min & DH=sec
            int 21h

            mov al,ch
            aam          ;to convert packed-BCD into unpacked-BCD & to display hr
            mov bx,ax
            call show

            mov dl,':'          ;dl contains a character
            mov ah,02h          ;function(02)/int 21h  to display a character on
            screen
            int 21h

            mov al,cl
            aam          ;to convert packed-BCD into unpacked-BCD & to display
            min
            mov bx,ax
            call show
            mov dl,':'              ;dl contains a character
            mov ah,02h              ;function(02)/int 21h  to display a character on screen
            int 21h

            mov al,dh
            aam                     ;to convert packed-BCD into unpacked-BCD & to
            display sec
            mov bx,ax
            call show

        termi: mov ah,4ch            ;function(4c)/int 21h  to terminate program
            int 21h

        show proc near              ;subroutine to display a character on screen
            mov dl,bh
            add dl,30h              ;obtain ASCII code of digit in BH
            mov ah,02h             ;function(02)/int 21h  to display a character on screen
            int 21h
            mov dl,bl
            add dl,30h              ;obtain ASCII code of digit in BL
            mov ah,02h             ;function(02)/int 21h  to display a character on screen
            int 21h
            ret
        show endp

code ends
        end start
```

10A) WRITE A PROGRAM TO SIMULATE A DECIMAL UP-COUNTER TO DISPLAY 00-99.

```
clear macro                      ;macro to clear screen
        mov ah,00h               ;fun(00)/int 19h to clear screen
        mov al,03h               ;set video mode=3
        int 10h
endm

dispc macro                     ;macro to display a character
        push ax                ;push & pop to save & restore ax content
        mov ah,02h             ;fun(02)/int 21h to display a charcter
        int 21h
        pop ax
endm

place macro pos                 ;macro to set cursor position
        mov ah,02h              ;fun(02)/int 10h to set cursor position
        mov dx,pos              ;dx contains positions, dh=y coordinate & dl=x coorinate
        int 10h
endm

data segment
        count dw 100
        n db  0ah
data ends
```

```
code segment
    assume cs:code, ds:data
         start: mov ax,data
                mov ds,ax

                clear

                mov bx,00h
        label1:mov ax,bx
                div n                  ;al=ax/n if number>09(0a) then obtain units place digit

                add ax,3030h            ;to obtain ascii value
                mov cx,ax              ;ah=tenth place digit    al=unit place digit

                place 0c28h            ;to set cursor position to tenth place
                mov dl,cl
                dispc

                place 0c29h             ;to set cursor position of unit place
                mov dl,ch
                dispc

                call delay
                inc bx
                dec count
                jnz label1
        termi: mov ah,4ch
                int 21h

        delay proc  near
                push ax
                push cx
                mov cx,1000h
          label2:mov ax,0000h
          label3:dec ax
                jnz label3
                dec cx
                jnz label2
                pop cx
                pop ax
               ret
        delay endp
code ends
        end start
```

11A) READ A PAIR OF INPUT CO-ORDINATES IN BCD AND MOVE THE CURSOR TO THE SPECIFIED LOCATION ON THE SCREEN.

```
disp macro msg              ;macro to display string
        mov ah,09h
        lea dx,msg
        int 21h
endm

clear macro                  ;macro to clear screen
        mov ah,00h
        mov al,03h
        int 10h
endm

read macro                  ;macro to read a character
        mov ah,01h
        int 21h
endm

setcur macro rw,cl           ;macro to set cursor position
        mov ah,02h
        mov dh,rw
        mov dl,cl
        int 10h
endm

data segment
        msg1 db 10,13,"enter the row value: $"
        msg2 db 10,13,"enter the column value: $"
        r db ?
        c db ?
data ends
```

```
code segment
   assume cs:code, ds:data
        start: mov ax,data
               mov ds,ax

               clear

               disp msg1
               call rdrowcol          ;read row# (00-18)
               mov r,al

               disp msg2
               call rdrowcol         ;read col# (00-49)
               mov c,al

               setcur r,c             ;set cursor position

               read                   ;similar to getch

               mov ah,4ch
               int 21h

       rdrowcol  proc  near          ;subroutine to read row and column
               read                   ;to read 1st digit of row/col
               sub al,30h            ;convert upper nibble from ASCII=>BCD
               mov cl,04h
               shl al,cl
               mov bl,al
               read                  ;to read 2nd digit of row/col
               sub al,30h            ;convert lower nibble from ASCII=>BCD
               add al,bl             ;byte=(upper nibble)+(lower nibble)
                ret
       rdrowcol endp

code ends
       end start
```

12A) WRITE A PROGRAM TO CREATE A FILE (INPUT FILE) AND TO DELETE AN EXISTING FILE.

```
disp macro msg
        mov ah,09h
        lea dx,msg
        int 21h
endm

read macro
        mov ah,01h
        int 21h
endm

data segment
        m1 db 10,13,"enter path name to create a file: $"
        m2 db 10,13,"enter path name to delete a file: $"
        m3 db 10,13,"file  is successfully created : $"
        m4 db 10,13,"file is not created : $"
        m5 db 10,13,"file  is successfully deleted : $"
        m6 db 10,13,"file is not deleted : $"
        str1 db 20 dup(0)
        str2 db 20 dup(0)
data ends
```

```
code segment
        assume cs:code, ds: data
         start: mov ax,data
                mov ds,ax
                disp m1
                lea si,str1
                call readfile

                lea dx,str1             ;dx contains offset address of file pathname
                mov cx,20h
                mov ah,3ch              ;fun(3c)/int21h used to create a new file
                int 21h
                jc next1                ;if error occurs while creating a file, goto next1
                disp m3
                jmp del
         next1:disp m4

           del: disp m2
                lea si,str2
                call readfile

                lea dx,str2             ;dx contains offset address of file pathname
                mov ah,41h              ;fun(41)/int21h used to delete an existing file
                int 21h
                jc next2                ;if error occurs while deleting a file, goto next1
                disp m5
                jmp termi

        next2: disp m6
         termi: mov ah,4ch
                 int 21h

        readfile proc near                  ;subroutine to read a file pathname
           back:read
                mov [si],al
                incsi
                cmp al,0dh
                jnz back
                mov bh,00h
                mov [si-1],bh
                ret
        readfile endp
code ends
        end star
```

1b) Read the status of eight input bits from the Logic Controller Interface and display ‘FF’ if it is the parity of the input read is even; otherwise display 00.

```
data segment
        portA dw 0b0coh
        portB dw 0b0c1h
        portC dw 0b0c2h
        cr dw 0b0c3h
data ends

code segment
 assume cs:code,ds:data
   start:  mov ax,data                    ;data segment initialization
           mov ds,ax

           mov al,82h                    ;control register initialization
           mov dx,cr                      ;portB=input    portA=output
           out dx,al

           mov dx, portB                  ;read data via portB
           in al,dx

           mov cx,8                       ;cx=counter
           mov bx,0                       ;bx used to hold no. of 1s
   back:ror al,1                         ;check if the bit is 0 or 1
           jnc down                      ;if no carry, goto label ‘down’ else increment bx
           inc bx
  down:dec cx
           jnz back

           shr bx,1                      ;if carry occurs then odd parity else even parity
           jc oddlab

evenlab:mov al,0ffh                       ;if even parity, display all lights
           mov dx, portA
           out dx,al
           jmp termi

 oddlab:mov al,00h                        ;if odd parity, switch off  all lights
           mov dx, portA
           out dx,al

  termi:  mov ah,4ch
           int 21h

code ends
  end start
```

2 b) Implement a BCD Up-Down Counter on the Logic Controller Interface

```
disp macro msg
        mov ah,09h
        lea dx,msg
        int 21h
endm

data segment
    portA dw 0b0coh
    portB dw 0b0c1h
    portC dw 0b0c2h
    cr  dw 0b0c3h
data ends

code segment
  assume  cs:code, ds:data
    start: mov ax,data                  ;data segment initialization
          mov  ds,ax

          mov al,82h                    ;control register initialization
          mov dx,cr                     ;porta=output   portb=input
          out dx,al

          mov cx,8                      ;cx=counter
          mov al,1                      ;load first value
   back1:mov dx, portA                  ;display the value via portA
          out dx,al
          call delay                    ;cause delay
          inc al                        ;increment to get next value
          dec cx
          jnz back1

          mov cx,8                       ;cx=counter
          mov al,8                      ;load first value
   back2:mov dx, portA                  ;display the value via portA
          out dx,al
          call delay                    ;cause delay
          dec al                        ;increment to get next value
          dec cx
          jnz back2

   termi:mov ah,4ch
          int 21h

          delay proc near
                    push ax
                    push bx
                    push cx
                    push dx
                    mov ax,5000h
            back2:mov cx,0ffffh
            back1:dec cx
                    jnz back1
                    dec ax
                    jnz  back2
                    pop  dx
                    pop  cx
                    pop  bx
                    pop  ax
                    ret
          delay  endp
  code ends
    end start
```

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

```
disp macro msg
          mov ah,09h
          lea dx,msg
          int 21h
endm

read macro
          mov ah,01h
          int 21h
endm

data segment
          portA dw 0b0c0h
          portB  dw 0b0c1h
          portC  dw 0b0c2h
          cr dw 0b0c3h
          m1 db 10,13, “enter first operand $”
          m2 db 10,13, ”enter second operand $”
          op1 db 00h
          op2 db 00h
data ends

code segment
 assume cs:code,ds:data
   start: mov ax,data                    ;data segment initialization
          mov ds,ax

          mov al,82h                     ;control register initialization
          mov dx,cr                      ;portA=output  portB=input
          out dx,al

          disp m1
          read
          mov dx, portB
          in al,dx                        ;read 1st  data via portB
          mov op1,al

          disp m2
          read
          mov dx, portB
          in al,dx                         ;read 2nd  data via portB
          mov op2,al

          mov al,op1
          mov bl,op2                       ;al=al*bl
          mul bl

          mov dx, portA                   ;display product via portA
          out dx,al

    termi:mov ah,4ch
          int 21h
code ends
  end start
```

4b) 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 (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

```
data segment
        array1 db 079h,0f7h,030h,071h,00h,00h           hexadecimal value of FIRE in reverse order
        array2 db 073h,038h,079h,076h,00h,00h   ;       hexadecimal value of HELP in reverse order
        pa dw 0b0c0h
        pb dw 0b0c1h
        pc dw 0b0c2h
        pr dw ob0c3h
data ends

code segment
 assume cs:code,ds:data
    start:mov ax,data           ;data segment initialization
        mov ds,ax

        mov dx,cr               ;control register initialization
        mov al,80h              ;portA=output    portC=output     portB=not used
        out dx,al

        mov di,04h              ;di=no 0f times to display FIRE/HELP
 again:mov ch,50d
 line1:mov cl,00h               ;select first position
        lea bx,array1
  chr1:mov dx,pc               ;select position of character (0th  t0 5th) via portc
        mov al,cl
        out dx,al
        xlat                   ;obtain 7-segment value of character from lookup table
        mov dx,pa              ;send 7 segment value via porta
        out dx,al
        call delay
        inc cl                 ;increment to select position of next character to be displayed
        cmp cl,05h             ;check if 5th position is reached
        jne chr1
        dec ch
        jnz line1

        mov ch,50d
 line2:mov cl,00h              ;select first position
        lea bx,array2
  chr2:mov dx,pc              ;select position of character (0th  t0 5th) via portc
        mov al,cl
        out dx,al
        xlat                  ;obtain 7-segment value of character from lookup table
        mov dx,pa             ;send 7 segment value via porta
        out dx,al
        call delay
        inc cl                 ;increment to select position of next character to be displayed
        cmp cl,05h             ;check if 5th position is reached
        jne chr2
        dec ch
        jnz line2

        dec di
        jnz again

  stop:mov ah,4ch
        int 21h
```

```
     delay proc near
                          push ax
                          push bx
                          push cx
                          push dx
                          mov bx,05ffh
                     b2:mov cx,0fffh
                     b1:dec cx
                          Jnz b1
                          dec bx
                          Jnz b2
                          pop dx
                          pop cx
                          pop bx
                          pop ax
                          ret
     delay endp
code ends
    end start
```

5b) 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. (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

```
data segment
  array1 db 00h ,00h,00h,00h,00h,00h
                db 079h,0f7h,030h,071h,00h,00h         ;hexadecimal value of FIRE in reverse order
                db 073h,038h,079h,076h,00h,00h         ;hexadecimal value of HELP in reverse order
                db 079h,0f7h,030h,071h,00h,00h         ;hexadecimal value of FIRE in reverse order
                db 00h,00h,00h,00h,00h,00h
       pa dw 0b0c0h
       pb dw 0b0c1h
       pc dw 0b0c2h
       cr dw 0b0c3h
data ends

code segment
  assume cs:code,ds:data
   start:mov ax,data                ;data segment initialization
               mov ds,ax

               mov dx,cr              ;control register initialization
               mov al,80h             ;portA=output    portC=output     portB=not used
               out dx,al

   again:mov di,18                  ;bh➔ no of characters to display message in rolling fashion
               lea bx,array1          ;bx➔contains offset address of table used with xlat

          nxt:mov ch,               ;ch=counter
       line1:mov cl,00h             ; select first position
       chr1:mov dx,pc                  ;select position of character (0th  t0 5th) via portc
               mov al,cl
               out dx,al
               xlat                   ;obtain 7-segment value of character from lookup table
               mov dx,pa              ;send 7 segment value via porta
               out dx,al
               call delay
               inc cl                 ;increment to select position of next character to be displayed
               cmp cl,05h             ;check if 5th position is reached
               jne chr1
               dec ch
               jnz line1
               Inc bx
               dec di
               jnz nxt

               push ax
               mov ah,0bh
               int 21h
               or al,al
               pop ax
               jz again

    stop:mov ah,4ch
               int 21h
```

```
delay proc near
              push cx
              push bx
              mov bx,05ffh
        b2:mov cx,0fffh
        b1:dec cx
              jnz b1
              dec bx
              jnz b2
              pop bx
              pop cx
              ret
  delay endp

code ends
     end start
```

6 b) 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.

```
data segment
        portA dw 0b090h
        portB dw 0b091h
        portC dw 0b092h
        cr dw 0b093h
        array1 db 0c0h,0f9h,0a4h,0b0h,099h, 092h,082h,0f8h,080h,90h          ;hexadecimal value of 0 to 9 in order
        array2 db 20 dup(00h)                                                ;00h means switch off all LEDs
        num dw 10000000b                                                     ;binary value of 128
data ends

code segment
assume cs:code, ds:data
   start:mov ax,data          ;data segment initialization
        movds,ax

        mov al,80h             ;control register initialization
        mov dx,cr              ;portA=output    portC=output     portB=not used
        out dx,al

        lea bx,array1          ;bx contains address of lookup-table used with xlat
        lea si,array2
        add si,07             ;set si to point to 7th element in array2

        mov ax,num
        mov dx,0
        mov cx,100             ;(dx:ax)/cx    128/100
        div cx                 ;after division, remainder=>dx(28), quotient=>ax(01)

        aam                    ;convert packed BCD to unpacked BCD(01h to 0001h)
        push ax
        mov al,ah
        xlat                   ;obtain 7 segment value of digit(ie 0) from lookup table and load into array2
        mov [si],al
        dec si                 ;decrement si to point to 6th element in array2
        pop ax
        xlat                   ;obtain 7 segment value of digit(ie 1) from lookup table and load into array2
        mov [si],al
        dec si                 ;decrement si to point to 5th element in array2

        mov ax,dx
        aam                    ;convert packed BCD to unpacked BCD(28h to 0208h)
        push ax
        mov al,ah
        xlat                   ;obtain 7 segment value of digit(ie 2) from lookup table and load into array2
        mov [si],al
        dec si                 ;decrement si to point to 4th element in array2
        pop ax
        xlat                   ;obtain 7 segment value of digit(ie 8) from lookup table and load into array2
        mov [si],al
       dec si

   again:mov di,18            ;bh→ no of characters to display message in rolling fashion
        lea bx,array1          ;bx→contains offset address of table used with xlat

    nxt:mov ch,50d            ;ch=counter
   line1:mov cl,00h           ; select first position
   chr1:mov dx,pc             ;select position of character (0th  t0 5th) via portc
        mov al,cl
        out dx,al
```

```
            xlat                    ;obtain 7-segment value of character from lookup table
            mov dx,pa               ;send 7 segment value via porta
            out dx,al
            call delay
            inc cl                  ;increment to select position of next character to be displayed
            cmp cl,05h              ;check if 5th position is reached
            jne chr1
            dec ch
            jnz line1
            Inc bx
            dec di
            jnz nxt

            push ax
            mov ah,0bh
            int 21h
            or al,al
            pop ax
            jz again

   stop:mov ah,4ch
            int 21h

delay proc near
            push cx
            push bx
            mov bx,05ffh
        b2:mov cx,0fffh
        b1:dec cx
            jnz b1
            dec bx
            jnz b2
            pop bx
            pop cx
            ret
  delay endp

code ends
    end start
```

7(b) Scan a 8*3 keypad for key closure and store the code of the key pressed in a memory location. Also, display row and column numbers of the key pressed.

```
data segment
            key db ?                 :used to hold key pressed
            pos db ?                 ;used to hold position of key pressed
            table db 00h,01h,02h,03h,04h,05h,06h,07h          ;hexa value of positions of characters of row0
                   db 10h,11h,12h,13h,14h,15h,16h,17h          ;hexa value of positions  of characters of row1
                   db 20h,21h,22h,23h,24h,25h,26h,27h          ;hexa value of positions of characters of row2
            portA dw 0b090h
            portB dw 0b091h
            portC dw 0b092h
            cr dw 0b093h
data segment

code segment
 assume cs:code, ds:data
      start: mov ax,data           ;data segment initialization
            mov ds,ax

            movdx,cr                 ;control register initialization
            mov al,90h               ;portA=input      portC=output
            out dx,al

    again: mov ch,00h                ;initially, assume ‘ch’ contains 1st number of 1st row
            mov al,1                  ;activate 1st row via portC
            mov dx,portC
            out dx,al
            call scan                  ;read if any key is pressed
            cmp al,0                  ;check if no key is pressed?
            jnz keypres               ;if key pressed then goto label ‘keypres’

            mov ch,08h                ;assume ‘ch’ contains 1st number of 2nd row
            mov al,2                  ;activate 2nd  row via portC
            mov dx,portC
            out dx,al
            call scan                  ;read if any key is pressed
            cmp al,0                   ;check if no key is pressed?
            jnz keypres               ;if key pressed then goto label ‘keypres’

            mov ch,10h                ;assume ‘ch’ contains 1st number of 3rd  row
            mov al,4                  ;activate 3rd  row via portC
            mov dx,portC
            out dx,al
            call scan                  ;read if any key is pressed
            cmp al,0                   ;check if no key is pressed?
            jnz keypres                ;if key pressed then goto label ‘keypres’
            jmp again

   keypres: mov key,ch               ;move key present in ‘ch’ to ‘val’
            mov al,ch
            lea bx,table
            xlat                     ;obtain position of key value pressed
            mov pos,al              ;move position into ‘pos’

        stop: int 03h                  ;terminate program
        scan proc near              ;procedure to read key pressed on the keypad
                mov dx,portA
                in al,dx            ;read key pressed via portA
                mov bh,08h          ;bh=counter

             rept: ror al,1         ;rotate to determine which bit is set(=1) i.e. to find key
                jc yes              ;if carry(1) generated then key found so return to main program
                inc ch              ;else search ‘1’(key) in remaining bits
                dec bh
                jnz rept
              yes: ret               ;return to main program
        scan endp

code ends
  end start
```

8(b) Drive a stepper motor interface to rotate the motor by N steps right direction and N steps left direction

```
data segment
            portA dw 0b090h
            portB dw 0b091h
            portC dw 0b092h
            cr dw 0b093h
data segment

code segment
  assume cs:code, ds:data
     start: mov ax,data             ;data segment initialization
            mov ds,ax

            mov al,80h              ;control register initialization
            mov dx,cr               ;portC=output
            outdx,al

            mov cx,08h              ;cx=counter
            mov al,88h              ;energize A as first coil
    back1:mov dx,portC
            out dx,al               ;energize each coil via portA
            call delay
            ror al,1                ;rotate right to energize each coil(A=>B=>C=>D=>A=>B=>C=>D)
            dec cx
            jnz back1

            mov cx,08h              ;cx=counter
            mov al,11h              ;energize D as first coil
    back2:mov dx,portC
            out dx,al               ;energize each coil via portA
            call delay
            rol al,1                ;rotate left to energize each coil(D=>C=>B=>A=>D=>C=>B=>A)
            dec cx
            jnz back2

     stop: mov ah,4ch
            int 21h


            delay proc near                     ;procedure to cause delay
                        push ax
                        pushbx
                        push cx
                        push dx
                        mov cx,06000h
                    z2: mov ax,0ffffh
                    z1: dec ax
                        jnz z1
                        dec cx
                        jnz z2
                        pop dx
                        pop cx
                        popbx
                        pop ax
                     ret
            delay endp

code ends
   end start
```

9b) Generate the sine wave using DAC interface

```
data segment
              porta dw 0b090h
              cr dw 0b093h
              table db 80h,96h,0abh,0c1h,0d2,0e2h,0ech,0f8h,0feh
                       db 0ffh,0feh,0f8h,0ech,0e2h,0d2h,0c1h,0abh,096h
                       db 80h, 69h,54h,40h,2dh,1dh,11h,07h,01h
                       db 00h,01h,07h,11h,1dh,2dh,40h,54h,69h
data ends

code segment
   assume  cs:code, ds:data
      start: mov ax,data
              movds,ax

              mov al,80h
              movdx,cr
              out dx,al

     again: lea si,table
               mov cx,36

     back: mov al,[si]
              mov dx,porta
              out dx,al
              call delay
              inc si
              dec cx
              jnz back

              mov ah,01h
              int 16h
              jz again

     stop: mov ah,4ch
              int 21h


              delay proc
                            mov bx,0ffffh
                   back1:dec bx
                            jnz back1
                            ret
               delay endp

code ends
     end start
```

10b) Generate the half rectified sine wave using DAC interface

```
data segment
        porta dw 0b0c0h
        cr dw 0b0c3h
        table db 80h,96h,0abh,0c1h,0d2,0e2h,0ech,0f8h,0feh
             db 0ffh,0feh,0f8h,0ech,0e2h,0d2h,0c1h,0abh,096h
             db 80h,80h,80h,80h,80h,80h,80h,80h,80h
             db 80h,80h,80h,80h,80h,80h,80h,80h,80h
data ends

code segment
  assume  cs:code, ds:data
    start: mov ax,data
        movds,ax

        mov al,80h
        movdx,cr
        out dx,al

   again: lea si,table
         mov cx,36

   back: mov al,[si]
        mov dx,porta
        out dx,al
        call delay
        inc si
        dec cx
        jnz back

        mov ah,01h
        int 16h
        jz again

   stop: mov ah,4ch
        int 21h


        delay proc
                mov bx,0ffffh
           back1:dec bx
                jnz back1
                ret
        delay endp

code ends
   end start
```

11b) Generate the fully rectified sine wave using DAC interface

```
data segment
        porta dw 0b0c0h
        cr dw 0b0c3h
        table db 80h,96h,0abh,0c1h,0d2,0e2h,0ech,0f8h,0feh
             db 0ffh,0feh,0f8h,0ech,0e2h,0d2h,0c1h,0abh,096h
             db 80h,96h,0abh,0c1h,0d2,0e2h,0ech,0f8h,0feh
             db 0ffh,0feh,0f8h,0ech,0e2h,0d2h,0c1h,0abh,096h
data ends

code segment
  assume  cs:code, ds:data
    start: mov ax,data
        movds,ax

        mov al,80h
        movdx,cr
        out dx,al

   again: lea si,table
         mov cx,36

   back: mov al,[si]
        mov dx,porta
        out dx,al
        call delay
        inc si
        dec cx
        jnz back

        mov ah,01h
        int 16h
        jz again

   stop: mov ah,4ch
        int 21h


        delay proc
                mov bx,0ffffh
           back1:dec bx
                jnz back1
                ret
         delay endp

code ends
   end start
```

12 b) 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 (approximately), and then come down to ground floor and stop. If some requests occur during going up or coming down they should be ignored.

```
data segment
            portA dw 0b0c0h
            portB dw 0b0c1h
            portC dw 0b0c2h
            cr dw 0b0c3h
data ends

read macro
            mov ah,01h
            int 21h
endm

code segment
  assume cs:code,ds:data
     start: mov ax,data                 ;data segment initialization
            mov ds,ax

            mov al,82h                   ;control register initialization
            mov dx,cr                   ;portA=output      portB=input
            out dx,al

            mov dx,portA
            mov al,0f0h
            out dx,al

            read                        ;similar to getch()

            mov dx, portB               ;read floor# requested(GND,1,2 or 3)
            in al,dx

            and al,0fh                   ;mask higher nibble of portB
            cmp al,0eh                   ;check if GND floor is selected
            jz gnd

            cmp al,0dh                  ;check if 1st floor is selected
            jz first

            cmp al,0bh                   ;check if 2nd  floor is selected
            jz sec

            jmp third                   ;default 3rd floor is selected

      gnd:mov cl,0
            call execute
            jmp stop

     first: mov cl,3
            call execute
            jmp stop
      sec:mov cl,6
            call execute
            jmp stop

     third: mov cl,9
            call execute
            jmp stop

      stop:mov ah,4ch
            int 21h


            execute proc near             ;procedure to move elevator to floor# requested, then unload & finally move to GND
                      mov al,00h
                      mov dx,portA
                   up:out dx,al           ;display elevator’s position at each step(i.e. intermediate steps from GND to requested floor#)
                      call delay
                      cmp al,cl
                      jz down
                      inc al              ;increment al to move elevator to floor# requested
                      jmp up
               down:out dx,al             ;display elevator’s position at each step(i.e. intermediate steps from requested floor# to GND)
                      cmp al,00h
                      jz last
                      call delay
                      dec al              ;decrement al to move elevator to GND
                      jmp  down
                  last:ret
            execute endp

            delay proc near
                      push ax
                      push bx
                      push cx
                      push dx
                      mov ax,5000h
                  z2:mov cx,0ffffh
                  z1:dec cx
                      jnz z1
                      dec ax
                      jnz z2
                      pop dx
                      pop cx
                      pop bx
                      pop ax
                      ret
            delay endp
code ends
  end start
```