

Microprocessor laboratory

Sub Code	: 06CSL48	IA Marks	: 25
Hrs / Week	: 03	Exam Hours	: 03
Total Hrs	: 42	Exam Marks	: 50

Note:

- Develop and execute the following programs using an 8086 Assembly Language. All the programs to be executed using an assembler like MASM, TASM etc.
- 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.

- 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 even parity bits otherwise display 00. Also display number of 1's in the input data.
- a) Write ALP macros:
 - i. To read a character from the keyboard in the module (1) (in a different file)
 - ii. To display a character in module(2) (from different file)
 - iii. Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.
 - b) Perform the following functions using the Logic Controller Interface.
 - i. BCD up-down Counter
 - ii. Ring Counter
- a) Sort a given set of 'n' numbers in ascending and descending orders 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.
- 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 it is necessary for the student to compute these values).
- 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 it is necessary for the student to compute these values).
- a) Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated

messages. Also display the length of the stored strings.

- 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 in front of the message What is your name? You must clear the entire screen before display.
 - b) Drive a Stepper Motor interface to rotate the motor in clockwise direction by N steps (N is specified by the examiner). Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).
8. a) Compute the factorial of a positive integer 'n' using recursive procedure.
 - b) Drive a stepper motor interface to rotate the motor in anti-clockwise direction by N steps (N is specified by the examiner). Introduce suitable delay between successive steps (Any arbitrary value for he delay may be assumed by the student).
9. 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 by N steps left direction and N steps right direction (N is specified by the examiner). Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).
10. a) Find out whether a given sub-string is present or not in a main string of characters.
 - 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.
11. a) Generate the first 'n' Fibonacci numbers.
 - b) Scan a 8 x 3 keypad for key closure and simulate ADD and SUBTRACT operations as in a calculator.
12. 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).
13. 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).
14. 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).
15. 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 couples of seconds, and then come down to ground floor and stop. If some requests occur during going up or coming down they should be ignored.

1a. Binary search(method 1)

Title Binary search

```
.model small
```

```
.data
```

```
arr dw 1234h,2345h,3456h,4567h,5678h,6789h,789ah
```

```
len db ($-arr-1)/2
```

```
key dw 789h
```

```
suc db 13,10,"Element found at position = "
```

```
pos db ?,13,10,'$'
```

```
fai db 13,10,"Element not found!!$"
```

```
.code
```

```
start: mov ax,@data
```

```
        mov ds,ax
```

```
        mov ax,00h
```

```
        mov cx,len
```

```
        mov dx,key
```

```
lp1:    cmp cx,ax
```

```
        jb fail
```

```
        mov bx,cx
```

```
        add bx,ax
```

```
        shr bx,01h
```

```
        mov si,bx
```

```
        shl si,01h
```

```
        cmp arr[si],dx
```

```
        jb gtr
```

```
        je succ
```

```
        cmp bx,00h
```

```
        je fail
```

```
        dec bx
```

```
        mov cx,bx
```

```
        jmp lp1
```

```
gtr:    inc bx
```

```
        mov ax,bx
```

```
        jmp lp1
```

```
succ:   add bl,'1'
```

```
        mov pos,bl
```

```
        lea dx,suc
```

```
        jmp print
```

```
fail:   lea dx,fai
```

```
print:  mov ah,09h
```

```
        int 21h
```

```
        mov ah,4ch
```

```
        int 21h
```

```
end start
```

1a.Binary search(method 2)

Title Binary search

```
.model small
```

```
.data
arr dw 1234h,2345h,3456h,4567h,5678h,6789h,789ah
len dw ($-arr-1)/2
key dw 789h
suc db 13,10,"Element found at position = "
pos db ?,13,10,'$'
fai db 13,10,"Element not found!!$"

.code
start: mov ax,@data
      mov ds,ax
      mov ax,00h
      mov cx,len
      mov dx,key
lp1:   cmp cx,ax
      jb fail
      mov bx,cx
      add bx,ax
      shr bx,01h
      mov si,bx
      shl si,01h
      cmp arr[si],dx
      jb gtr
      je succ
      dec bx
      js fail
      mov cx,bx
      jmp lp1
gtr:   inc bx
      mov ax,bx
      jmp lp1
succ:  add bl,'1'
      mov pos,bl
      lea dx,suc
      jmp print
fail:  lea dx,fai
print: mov ah,09h
      int 21h
      mov ah,4ch
      int 21h
end start
```

1b.Parity(logic controller)

Title Parity

```
.model small
```

```
.code
start: mov dx,303h
      mov al,82h
      out dx,al
      mov dx,301h
      in al,dx
      mov cx,08h
```

```
        mov bl,00h
lp1:    ror al,01h
        adc bl,00h
        loop lp1
        mov al,bl
        mov ah,00h
        mov bh,02h
        div bh
        cmp ah,00h
        je ev
        mov al,0h
        jmp disp
ev:     mov al,0ffh
disp:   mov dx,300h
        out dx,al
        mov dl,bl
        add dl,'0'
        mov ah,02h
        int 21h
        mov ah,4ch
        int 21h
end start
```

2a1.Macro to read

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

2a2.Display macro

```
disp macro
mov ah,02h
int 21h
endm
```

2a.Main program

Title String read and display using macros stored in different files

```
include 2a1.asm
include 2a2.asm

.model small

.data
loc db 100 dup(0)
st0 db 13,10,"Enter a string",13,10,'$'
st1 db 13,10,"Entered string is $"

.code
start: mov ax,@data
        mov ds,ax
        mov cl,00h
        lea bx,loc
        lea dx,st0
```

```

        mov ah,09h
        int 21h
        lea si,loc
rd:      read
        cmp al,08h
        je new
        cmp al,0dh
        je print
        mov [si],al
        inc si
        jmp rd
new:     mov dl,' '
        disp
        mov dl,08h
        disp
        cmp si,bx
        je rd
        dec si
        jmp rd
print:   mov al,'$'
        mov [si],al
        lea dx,st1
        mov ah,09h
        int 21h
        lea si,loc
pri:     mov dl,[si]
        cmp dl,'$'
        je ter
        disp
        inc si
        jmp pri
ter:     mov ah,4ch
        int 21h
end start

```

2bi.BCD updown counter(logic controller)

Title Ring counter

```
.model small
```

```
.code
```

```

start:  mov al,80h
        mov dx,303h
        out dx,al
        mov dx,300h
        mov al,80h
        mov cx,30h
lp1:    out dx,al
        ror al,01h
        call delay1
        loop lp1
        mov ah,4ch
        int 21h

        delay1 proc

```

```
        push cx
        push ax
        mov ax,0aah
lp3:    loop lp3
        dec ax
        jnz lp3
        pop ax
        pop cx
        ret
        delay1 endp
end start
```

2bii.Ring counter

Title Ring counter

```
.model small
```

```
.code
```

```
start: mov al,80h
        mov dx,303h
        out dx,al
        mov dx,300h
        mov al,80h
        mov cx,30h
lp1:    out dx,al
        ror al,01h
        call delay1
        loop lp1
        mov ah,4ch
        int 21h
```

```
        delay1 proc
        push cx
        push ax
        mov ax,0aah
lp3:    loop lp3
        dec ax
        jnz lp3
        pop ax
        pop cx
        ret
        delay1 endp
end start
```

3a.Bubble sort(Ascending order)

Title Bubble sort(ascending)

```
.model small
```

```
.data
```

```
arr db 5h,7h,6h,4h,10h,09h
len db $-arr
```

```
.code
start: mov ax,@data
      mov ds,ax
      mov cl,len
lp1:   mov bx,cx
      lea si,arr
lp2:   mov al,[si]
      inc si
      cmp [si],al
      jb lp3
      xchg [si],al
      mov [si-1],al
lp3:   dec bx
      jnz lp2
      loop lp1
      mov ah,4ch
      int 21h
end start
```

3a.Bubble sort(Descending order)

Title Bubble sort(ascending)

```
.model small

.data
arr db 5h,7h,6h,4h,10h,09h
len db $-arr

.code
start: mov ax,@data
      mov ds,ax
      mov cl,len
lp1:   mov bx,cx
      lea si,arr
lp2:   mov al,[si]
      inc si
      cmp [si],al
      jb lp3
      xchg [si],al
      mov [si-1],al
lp3:   dec bx
      jnz lp2
      loop lp1
      mov ah,4ch
      int 21h
end start
```

3b.Multiplication(method 1)

Title Multiplicaion(8X8)

```
.model small
```



```

.code
start: mov dx,303h
      mov al,8bh
      out dx,al
      mov dx,301h
      in al,dx
      mov cl,al
      mov dx,302h
      in al,dx
      mov ah,00h
      mul cl
      mov dx,300h
      out dx,al
      mov bx,0aaah
lp1:   loop lp1
      dec bx
      jnz lp1
      mov al,ah
      out dx,al
      mov ah,4ch
      int 21h
end start

```

3b.Multiplication(method 2)

Title Multiplicaion(8X8)

```
.model small
```

```

.code
start: mov dx,303h
      mov al,8bh
      out dx,al
      mov dx,301h
      in al,dx
      mov cl,al
      mov dx,302h
lp:    in al,dx
      cmp al,80h
      jb lp
      mov dx,301h
      in al,dx
      mov ah,00h
      mul cl
      mov dx,300h
      out dx,al
      mov bx,0aaah
lp1:   loop lp1
      dec bx
      jnz lp1
      mov al,ah
      out dx,al
      mov ah,4ch
      int 21h

```

```
end start
```

4a.ASCII codes(method 1)

Title Alphanumeric charecter - ASCII codes

```
.model small
```

```
.data
```

```
msg db "Enter the charecter to check the ASCII value$"
```

```
no db ?
```

```
ms1 db ',' '-'
```

```
bcd db 4 dup(0)
```

```
.code
```

```
start:  mov ax,@data
```

```
        mov ds,ax
```

```
        lea dx,msg
```

```
        mov ah,09h
```

```
        int 21h
```

```
lp1:    mov ah,01h
```

```
        int 21h
```

```
        cmp al,1ah
```

```
        je ter
```

```
        mov no,al
```

```
        mov al,00h
```

```
        mov cx,00h
```

```
        mov dx,1850h
```

```
        mov ah,06h
```

```
        mov bh,07h
```

```
        int 10h
```

```
        mov dx,0c23h
```

```
        mov ah,02h
```

```
        mov bh,00h
```

```
        int 10h
```

```
        call cvt
```

```
        lea dx,no
```

```
        mov ah,09h
```

```
        int 21h
```

```
        jmp lp1
```

```
ter:    mov ah,4ch
```

```
        int 21h
```

```
        cvt proc
```

```
        push bx
```

```
        mov [bcd+3], '$'
```

```
        mov al,no
```

```
        mov cl,0ah
```

```
        mov bx,02h
```

```
lp:     mov ah,00h
```

```
        div cl
```

```
        add ah,'0'
```

```
        mov bcd[bx],ah
```

```
        dec bx
```

```
        jns lp
```

```
        pop bx
```

```

    ret
    cvt endp
end start

```

4a.ASCII codes(method 2)

Title Alphanumeric charecter - ASCII codes

```
.model small
```

```
.data
```

```
msg db "Enter 25 charecters to check the ASCII value$"

```

```
no db ?

```

```
ms1 db ',' '-' ','

```

```
bcd db 4 dup(0)

```

```
.code
```

```
start:mov ax,@data

```

```
    mov ds,ax

```

```
    lea dx,msg

```

```
    mov ah,09h

```

```
    int 21h

```

```
    mov cx,19h

```

```
lp1:  push cx

```

```
    mov ah,01h

```

```
    int 21h

```

```
    mov no,al

```

```
    mov al,00h

```

```
    mov cx,00h

```

```
    mov dx,1850h

```

```
    mov ah,06h

```

```
    mov bh,07h

```

```
    int 10h

```

```
    mov dx,0c23h

```

```
    mov ah,02h

```

```
    mov bh,00h

```

```
    int 10h

```

```
    call cvt

```

```
    lea dx,no

```

```
    mov ah,09h

```

```
    int 21h

```

```
    pop cx

```

```
    loop lp1

```

```
ter:  mov ah,4ch

```

```
    int 21h

```

```

cvt proc

```

```
    push bx

```

```
    mov [bcd+3],'$'

```

```
    mov al,no

```

```
    mov cl,0ah

```

```
    mov bx,02h

```

```
lp:   mov ah,00h

```

```
    div cl

```

```
    add ah,'0'

```

```
    mov bcd[bx],ah
    dec bx
    jns lp
    pop bx
    ret
    cvt endp
end start
```

4b.FIRE and HELP on 7 segment display

Title Display FIRE and HELP on 7 segment display

```
.model small
```

```
.data
```

```
fir db 86h,88h,0f9h,8eh
```

```
hel db 8ch,0c7h,86h,89h
```

```
.code
```

```
start: mov ax,@data
```

```
    mov ds,ax
```

```
    mov dx,303h
```

```
    mov al,80h
```

```
    out dx,al
```

```
    mov ah,0ah
```

```
lp:    mov bx,00h
```

```
    lea si,fir
```

```
lp1:   mov cx,07h
```

```
lp2:   mov dx,301h
```

```
    mov al,si[bx]
```

```
    ror al,cl
```

```
    out dx,al
```

```
    mov dx,302h
```

```
    mov al,0ffh
```

```
    out dx,al
```

```
    mov al,00h
```

```
    out dx,al
```

```
    dec cx
```

```
    jns lp2
```

```
    inc bx
```

```
    cmp bx,04h
```

```
    jb lp1
```

```
    call delay1
```

```
    mov bx,00h
```

```
    lea si,hel
```

```
lp3:   mov cx,07h
```

```
lp4:   mov dx,301h
```

```
    mov al,si[bx]
```

```
    ror al,cl
```

```
    out dx,al
```

```
    mov dx,302h
```

```
    mov al,0ffh
```

```
    out dx,al
```

```
    mov al,00h
```

```
    out dx,al
```

```
    dec cx
```

```
jns lp4
inc bx
cmp bx,04h
jb lp3
call delay1
dec ah
jns lp
mov ah,4ch
int 21h
```

```
delay1 proc
push cx
push bx
mov bx,0aaah
lp5:  loop lp5
dec bx
jnz lp5
pop bx
pop cx
ret
delay1 endp
end start
```

5a.Palindrome

Title Palindrome

.model small

.data

```
act db 99 dup(0)
rev db 99 dup(0)
sl db 13,10,"String length is="
len db ?,?','$'
pal db 13,10,"Entered string is a palindrome$"
npal db 13,10,"Entered string is not a palindrome$"
stg db "Enter a string",13,10,'$'
```

.code

```
start: mov ax,@data
       mov ds,ax
       lea dx,stg
       mov ah,09h
       lea si,act
       mov bx,00h
       int 21h
lp1:   mov ah,01h
       int 21h
       cmp al,08h
       je bck
       cmp al,0dh
       je lp2
       mov si[bx],al
       inc bx
       jmp lp1
```

```

bck:  cmp bx,00h
      je lp1
      dec bx
      mov dl,' '
      mov ah,02h
      int 21h
      mov dl,08h
      int 21h
      jmp lp1
lp2:  mov al,'$'
      mov si[bx],al
      mov ax,bx
      mov cx,bx
      mov bl,0ah
      mov ah,00h
      div bl
      add ah,'0'
      mov [len+1],ah
      add al,'0'
      mov [len],al
      lea di,rev
      mov bx,cx
      dec bx
lp3:  mov al,si[bx]
      mov [di],al
      inc di
      dec bx
      jns lp3
      mov al,'$'
      mov [di],al
      lea di,rev
lp4:  mov al,[di]
      cmp al,[si]
      jne fail
      inc si
      inc di
      loop lp4
      lea dx,pal
      jmp dsp
fail:  lea dx,npal
dsp:  mov ah,09h
      int 21h
      lea dx,sl
      int 21h
      mov ah,4ch
      int 21h
end start

```

5b.Message on a 7 segment display

Title Display any 12 charecters on 7 segment display

.model small

.data

```
codes db 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,98h,88h,80h,0c6h
```

```
.code
start: mov ax,@data
      mov ds,ax
      mov dx,303h
      mov al,80h
      out dx,al
      lea si,codes
      mov ah,0ah
lp:    mov bx,00h
lp1:   mov cx,07h
lp2:   mov dx,301h
      mov al,si[bx]
      ror al,cl
      out dx,al
      mov al,0ffh
      mov dx,302h
      out dx,al
      mov al,00h
      out dx,al
      dec cx
      jns lp2
      call delay1
      inc bx
      cmp bx,0dh
      jne lp1
      dec ah
      jnz lp
      mov ah,4ch
      int 21h

      delay1 proc
      push cx
      push bx
      mov bx,0aah
lp3:   loop lp3
      dec bx
      jnz lp3
      pop bx
      pop cx
      ret
      delay1 endp
end start
```

6a.Compare two strings

Title Cmparision of two strings

```
.model small
```

```
.data
st1 db 99 dup(0)
st2 db 99 dup(0)
sl1 db 13,10,"String length of string 1 is = "
ln1 db ?,?,'$'
```

```

sl2 db 13,10,"String length of string 2 is = "
ln2 db ?,?,'$'
ln db ?
m1 db 13,10,"Enter string 1",13,10,'$'
m2 db 13,10,"Enter string 2",13,10,'$'
suc db 13,10,"Entered strings are equal$"
fai db 13,10,"Entered strings are not equal$"

```

```
.code
```

```

start: mov ax,@data
      mov ds,ax
      lea dx,m1
      mov ah,09h
      int 21h
      lea si,st1
      call read
      mov ln,bl
      call cvt
      mov [ln+1],ah
      mov [ln1],al
      lea dx,m2
      mov ah,09h
      int 21h
      lea si,st2
      call read
      call cvt
      mov [ln2+1],ah
      mov [ln2],al
      mov ah,ln1
      mov al,ln1+1
      mov bh,ln2
      mov bl,ln2+1
      cmp bx,ax
      jne fail
      mov cl,ln
      mov ch,00h
      lea si,st1
      lea di,st2
lp3:   mov al,[si]
      cmp al,[di]
      jne fail
      inc si
      inc di
      loop lp3
      lea dx,suc
      jmp disp
fail:  lea dx,fai
disp:  mov ah,09h
      int 21h
      lea dx,sl1
      int 21h
      lea dx,sl2
      int 21h
      mov ah,4ch
      int 21h

      read proc
      mov bx,00h
lp1:   mov ah,01h

```



```

        int 21h
        cmp al,08h
        je bck
        cmp al,0dh
        je lp2
        mov si[bx],al
        inc bx
        jmp lp1
bck:    mov dl,' '
        mov ah,02h
        int 21h
        mov dl,08h
        int 21h
        cmp bx,00h
        je lp1
        dec bx
        jmp lp1
lp2:    mov al,'$'
        mov si[bx],al
        ret
        read endp

        cvt proc
        mov ax,bx
        mov bl,0ah
        mov ah,00h
        div bl
        add ah,'0'
        add al,'0'
        ret
        cvt endp
end start

```

6b.Convert from BCD to binary and display on 7 segment display

Title BCD to binary and display on 7 segment display

```

.model small

.data
bin dw 0ffffh
bcd db 5 dup(0)
cod db 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,90h

.code
start: mov ax,@data
        mov ds,ax
        mov al,80h
        mov dx,303h
        out dx,al
        mov ax,bin
        mov dx,00h
        mov bx,04h
        mov cx,0ah

```

```
lp:    div cx
        mov bcd[bx],dl
        mov dl,00h
        dec bx
        cmp ax,09h
        jnb lp
        mov bcd[bx],al
        mov ah,03h
lp1:    push ax
        mov bx,00h
lp2:    lea si,bcd
        mov al,si[bx]
        push bx
        mov bl,al
        lea si,cod
        mov cx,07h
        call disp
        pop bx
        cmp bx,03h
        jne incr
        call delay1
incr:   inc bx
        cmp bx,05h
        jne lp2
        call delay1
        mov bx,03h
lp4:    lea si,bcd
        mov al,si[bx]
        push bx
        mov bl,al
        lea si,cod
        mov cx,07h
        call disp
        pop bx
        dec bx
        jns lp4
        call delay1
        mov bx,04h
lp8:    lea si,bcd
        mov al,si[bx]
        push bx
        mov bl,al
        lea si,cod
        mov cx,07h
        call disp
        pop bx
        dec bx
        jnz lp8
        call delay1
        pop ax
        dec ah
        cmp ah,00h
        jnz lp1
        mov ah,4ch
        int 21h

        disp proc
lp3:    mov al,si[bx]
        mov dx,301h
```

```
    ror al,cl
    out dx,al
    mov al,0ffh
    mov dx,302h
    out dx,al
    mov al,00h
    out dx,al
    dec cx
    jns lp3
    ret
disp endp

delay1 proc
push cx
push bx
mov bx,05aah
lp5:  loop lp5
      dec bx
      jnz lp5
      pop bx
      pop cx
      ret
delay1 endp
end start
```

7a.Read name from some location of the screen

Title Read name from some location on the screen

.model small

.data

ms1 db "What is your name? \$"

ms2 db "My name is: "

nam db 99 dup(0)

.code

```
start: mov ax,@data
      mov ds,ax
      call clr
      mov dx,0c23h
      call pos
      lea dx,ms1
      mov ah,09h
      int 21h
      lea si,nam
      call read
      mov dx,0d23h
      call pos
      lea dx,ms2
      mov ah,09h
      int 21h
      mov ah,4ch
      int 21h

      clr proc
```

```
        mov ah,06h
        mov al,00h
        mov bh,07h
        mov cx,00h
        mov dx,1850h
        int 10h
        ret
    clr endp

    pos proc
        mov ah,02h
        mov bh,00h
        int 10h
        ret
    pos endp

    read proc
        lea di,nam
lp1:    mov ah,01h
        int 21h
        cmp al,08h
        je bck
        cmp al,0dh
        je dol
        mov [si],al
        inc si
        jmp lp1
bck:    mov dl,' '
        mov ah,02h
        int 21h
        mov dl,08h
        int 21h
        cmp si,di
        je lp1
        dec si
        jmp lp1
dol:    mov al,'$'
        mov [si],al
        ret
    read endp
end start
```

7b. Stepper motor(clockwise direction)

Title Motor clock wise

.model small

.data

msg db "Motor is rotating in clockwise direction\$"

.code

```
start: mov ax,@data
        mov ds,ax
        lea dx,msg
        mov ah,09h
```

```
int 21h
mov al,80h
mov dx,303h
out dx,al
mov cx,0c8h
mov al,077h
mov dx,302h
lp1: out dx,al
     call delay1
     ror al,01h
     loop lp1
     mov ah,4ch
     int 21h

     delay1 proc
     push cx
     push bx
     mov bx,00aah
lp2:  loop lp2
     dec bx
     jnz lp2
     pop bx
     pop cx
     ret
     delay1 endp
end start
```

8a.Factorial

Title Factorial

```
.model small
```

```
.data
loc db 08h
fct dw ?
```

```
.code
start: mov ax,@data
       mov ds,ax
       mov bl,loc
       mov ax,01h
       call fact
       mov fct,ax
       mov ah,4ch
       int 21h
```

```
fact proc
  cmp bx,00h
  je rtn
  mul bx
  dec bx
  call fact
rtn:  ret
fact endp
end start
```

8b. Stepper motor(anti clockwise direction)

Title Motor anti clock wise

.model small

.data

msg db "Motor is rotating in anti clockwise direction\$"

.code

```
start: mov ax,@data
      mov ds,ax
      lea dx,msg
      mov ah,09h
      int 21h
      mov al,80h
      mov dx,303h
      out dx,al
      mov cx,0c8h
      mov al,0eeh
      mov dx,302h
lp1:   out dx,al
      call delay1
      rol al,01h
      loop lp1
      mov ah,4ch
      int 21h
```

```
      delay1 proc
      push cx
      push bx
      mov bx,00aah
lp2:   loop lp2
      dec bx
      jnz lp2
      pop bx
      pop cx
      ret
      delay1 endp
end start
```

9a.nCr

Title ncr

.model small

.data

n db 05h

r db 02h

ncr dw ?

.code

```
start: mov ax,@data
```

```
mov ds,ax
mov ax,00h
mov al,n
mov bl,r
mov ncr,00h
call ncrp
mov ah,4ch
int 21h
```

```
ncrp proc
cmp ax,bx
je pls1
cmp bx,00h
je pls1
cmp bx,01h
je plsn
dec ax
cmp ax,bx
je pls
push ax
push bx
call ncrp
pop bx
pop ax
dec bx
push ax
push bx
call ncrp
pop bx
pop ax
ret
pls1: inc ncr
ret
plsn: add ncr,ax
ret
pls:  add ncr,ax
inc ncr
ret
ncrp endp
end start
```

9b. Stepper motor in both directions

Title Motor clock wise and anti clock wise

```
.model small
```

```
.data
```

```
msg db "Motor is rotating in clockwise direction$"
```

```
ms1 db "Motor is rotating in anti-clockwise direction$"
```

```
.code
```

```
start: mov ax,@data
mov ds,ax
lea dx,msg
mov ah,09h
```

```

        int 21h
        mov al,80h
        mov dx,303h
        out dx,al
        mov cx,064h
        mov dx,302h
        mov al,077h
lp:      out dx,al
        ror al,01h
        call delay1
        loop lp
        lea dx,ms1
        mov ah,09h
        int 21h
        mov cx,064h
        mov al,0eeh
lp1:     out dx,al
        call delay1
        rol al,01h
        loop lp1
        mov ah,4ch
        int 21h

        delay1 proc
        push cx
        push bx
        mov bx,00aah
lp2:     loop lp2
        dec bx
        jnz lp2
        pop bx
        pop cx
        ret
        delay1 endp
end start

```

10a.Substring

Title Sub string

```
.model small
```

```

.data
st0 db 99 dup(0)
st1 db 99 dup(0)
str0 db 13,10,"Enter main string",13,10,'$'
str1 db 13,10,"Enter sub string",13,10,'$'
ln1 db 13,10,"Length of main string is = "
len1 db ?,?,'$'
ln2 db 13,10,"Length of sub string is = "
len2 db ?,?,'$'
succ db 13,10,"Substring found in string$"
fail db 13,10,"Substring not found in string$"

```

```

.code
start: mov ax,@data

```



```

mov ds,ax
lea dx,str0
mov ah,09h
int 21h
lea si,st0
call read
mov len1,al
mov len1+1,ah
lea dx,str1
mov ah,09h
int 21h
push cx
lea si,st1
call read
mov len2,al
mov len2+1,ah
mov bh,len1+1
mov bl,len1
pop dx
lea si,st0
lea di,st1
mov bx,00h
mlp:  cmp dx,cx
      jb flr
      push cx
lp:   mov al,si[bx]
      cmp al,[di]
      jne incr
      inc di
      inc bx
      loop lp
      jmp suc
incr: inc bx
      dec dx
      jmp mlp
flr:  lea dx,fail
      jmp disp
suc:  lea dx,succ
disp: mov ah,09h
      int 21h
      lea dx,ln1
      int 21h
      lea dx,ln2
      int 21h
      mov ah,4ch
      int 21h

      read proc
      mov bx,00h
lp1:  mov ah,01h
      int 21h
      cmp al,08h
      je bck
      cmp al,0dh
      je lp2
      inc bx
      mov si[bx],al
      jmp lp1
bck:  mov ah,02h

```

```

        mov dl,' '
        int 21h
        mov dl,08h
        int 21h
        cmp bx,00h
        je lp1
        dec bx
        jmp lp1
lp2:    mov al,'$'
        mov si[bx],al
        mov ax,bx
        mov cx,ax
        mov bl,0ah
        div bl
        add ah,'0'
        add al,'0'
        ret
        read endp
end start

```

10b.Keypad

Title Keypad(8X3)

```
.model small
```

```

.data
msg db "0123456789ABCDEFGHIJ"
rd db 13,10,"Read character is = $"
rw db 13,10,"Row number is = "
row db ?
cl1 db 13,10,"Column number is = "
col db ?, '$'
en db 13,10,"Enter 20 characters from keypad.$"

```

```
.code
```

```

start: mov ax,@data
        mov ds,ax
        mov dx,303h
        mov al,90h
        out dx,al
        lea dx,en
        mov ah,09h
        int 21h
        mov cx,14h
lp:     mov dx,302h
        mov al,07h
        out dx,al
        mov dx,300h
lp1:    in al,dx
        cmp al,00h
        je lp1
        call cvt
        mov bx,0403h
lp2:    mov al,bh
        mov dx,302h

```

```
out dx,al
mov dx,300h
in al,dx
ror bh,01h
dec bl
cmp al,00h
je lp2
add bl,'1'
mov col,bl
call disp
loop lp
mov ah,4ch
int 21h

cvt proc
push cx
mov cx,08h
lp3:  rol al,01h
      jc lp4
      loop lp3
lp4:  add cl,'0'
      mov row,cl
      pop cx
      ret
cvt endp

disp proc
mov al,col
sub al,'1'
mov bl,08h
mov ah,00h
mul bl
mov bl,row
sub bl,'1'
add al,bl
mov bx,ax
lea dx,rd
mov ah,09h
int 21h
lea si,msg
mov dl,si[bx]
mov ah,02h
int 21h
lea dx,rw
mov ah,09h
int 21h
push cx
push bx
mov bx,011h
lp5:  loop lp5
      dec bx
      jnz lp5
      pop bx
      pop cx
      ret
disp endp
end start
```

11a.Fibonacci numbers(method 1)

Title Fibonacci numbers

.model small

.data

no db ?

no1 dw ?

no2 dw ?

msg db 13,10,"Enter the number of Fibonacci numbers to be displayed \$"

zerr db 13,10,"Pls enter any other number other than 0\$"

ms1 db 13,10,"The fibonacci numbers are",13,10,'\$'

.code

start: mov ax,@data

mov ds,ax

lp1: mov ah,09h

lea dx,msg

int 21h

mov ah,01h

int 21h

sub al,'0'

mov bl,al

int 21h

sub al,'0'

mov ah,00h

xchg al,bl

mov bh,0ah

mul bh

add al,bl

mov no,al

mov cl,al

mov ch,00h

cmp al,00h

jne cnt

lea dx,zerr

mov ah,09h

int 21h

jmp lp1

cnt: lea dx,ms1

mov ah,09h

int 21h

mov ax,00h

mov bx,01h

mov no1,ax

mov no2,bx

lp2: call disp

mov ax,no1

mov bx,no2

mov dx,ax

add dx,bx

mov ax,bx

mov bx,dx

mov no1,ax

mov no2,bx

loop lp2

mov ah,4ch

```

        int 21h

        disp proc
        push cx
        mov cx,05h
        mov bx,0ah
lp:      mov dx,00h
        div bx
        push dx
        loop lp
        mov ah,02h
        mov cx,05h
lp3:     pop dx
        add dl,'0'
        int 21h
        loop lp3
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
        pop cx
        ret
        disp endp
end start

```

11a.Fibonacci numbers(method 2)

Title Generate first n fibonacci numbers

```
.model small
```

```
.data
msg db 13,10,"Enter the value of n(1<=n<=300) $"
n dw ?
suc db 13,10,"Fibonacci num.....",13,10,'$'
fai db 13,10,"The value of entered n is 0!!!$"
no1 db 66 dup('0')
no2 db 66 dup('0')

```

```
.code
start: mov ax,@data
        mov ds,ax
        lea dx,msg
        mov ah,09h
        int 21h
        mov ah,01h
        int 21h
        sub al,'0'
        mov bl,al
        int 21h
        sub al,'0'
        mov bh,al
        int 21h
        sub al,'0'
        mov cl,al
        mov ch,64h

```

```
    mov al,bl
    mov ah,00h
    mul ch
    mov n,ax
    mov ah,00h
    mov al,bh
    mov ch,0ah
    mul ch
    add n,ax
    mov ch,00h
    add n,cx
    mov dx,n
    cmp dx,00h
    jne succ
    lea dx,fai
    mov ah,09h
    int 21h
ex:  mov ah,4ch
    int 21h
succ: lea dx,suc
    mov ah,09h
    int 21h
    mov dl,'0'
    mov ah,02h
    int 21h
    mov dl,13
    int 21h
    mov dl,10
    int 21h
    lea si,no2
    mov bx,41h
    mov al,'1'
    mov si[bx],al
lp:  mov dx,n
    dec dx
    mov n,dx
    cmp dx,00h
    je ex
    lea si,no2
    mov bx,41h
    add si,bx
    lea bx,no2
    dec bx
    mov ch,42h
    mov ah,00h
lp1: mov al,[si]
    sub al,'0'
    push bx
    mov bl,[bx]
    sub bl,'0'
    add al,bl
    add al,ah
    mov bl,0ah
    mov ah,00h
    div bl
    xchg al,ah
    add al,'0'
    pop bx
    mov [bx],al
```

```
dec si
dec bx
dec ch
cmp ch,00h
jne lp1
lea si,no2
call disp
mov dx,n
dec dx
mov n,dx
cmp dx,00h
je ter
lea bx,no2
mov si,41h
add bx,si
lea si,no2
dec si
mov ch,42h
mov ah,00h
lp2:  mov al,[si]
      sub al,'0'
      push bx
      mov bl,[bx]
      sub bl,'0'
      add al,bl
      add al,ah
      mov bl,0ah
      mov ah,00h
      div bl
      xchg al,ah
      add al,'0'
      pop bx
      mov [bx],al
      dec si
      dec bx
      dec ch
      cmp ch,00h
      jne lp2
      lea si,no1
      call disp
      jmp lp
ter:  mov ah,4ch
      int 21h

disp proc
mov ch,42h
mov ah,02h
mov cl,00h
lp3:  mov dl,[si]
      cmp dl,'0'
      jne lp4
      cmp cl,00h
      je lp5
lp4:  inc cl
      int 21h
lp5:  inc si
      dec ch
      cmp ch,00h
      jne lp3
```

```

        mov dl,13
        int 21h
        mov dl,10
        int 21h
        ret
    disp endp
end start

```

11b. Calculator using 8X3 keypad

Title Calculator using 8X3 keypad

```
.model small
```

```

.data
ms1 db 13,10,"Enter operand 1 $"
msg db 13,10,"Enter operator $"
ms2 db 13,10,"Enter operand 2 $"
op1 db ?
opr db ?
op2 db ?
rst db "Result = "
res db ?,?,'$'
fai db 13,10,"Invalid!!$"
row db ?
col db ?

```

```

.code
start: mov ax,@data
        mov ds,ax
        mov dx,303h
        mov al,90h
        out dx,al
        lea dx,ms1
        mov ah,09h
        int 21h
        call read
        call cvt
        call chop
        jc fail
        mov op1,al
        lea dx,msg
        mov ah,09h
        int 21h
        call read
        call cvt
        call chor
        jc fail
        mov opr,al
        lea dx,ms2
        mov ah,09h
        int 21h
        call read
        call cvt
        call chop
        jc fail

```



```
    mov op2,al
    cmp opr,0ch
    je lp1
    cmp opr,0dh
    je lp2
    cmp opr,0eh
    je lp3
    cmp op2,00h
    je fail
    mov al,op1
    mov bl,op2
    mov ah,00h
    div bl
    add al,'0'
    mov res,'0'
    mov res+1,al
    call disp
ter:  mov ah,4ch
      int 21h
fail: lea dx,fai
      mov ah,09h
      int 21h
      jmp ter
lp1:  mov al,op1
      add al,op2
      mov ah,00h
      mov bl,0ah
      div bl
      add ax,3030h
      mov res,al
      mov res+1,ah
      call disp
      jmp ter
lp2:  mov al,op1
      mov bl,op2
      cmp al,bl
      jb blw
      sub al,bl
      add al,'0'
      mov res,'0'
      mov res+1,al
      call disp
      jmp ter
blw:  mov res,'-'
      sub bl,al
      add bl,'0'
      mov res+1,bl
      call disp
      jmp ter
lp3:  mov al,op1
      mov bl,op2
      mov ah,00h
      mul bl
      mov bl,0ah
      div bl
      add ax,3030h
      mov res,al
      mov res+1,ah
      call disp
```

```
        jmp ter

        read proc
        mov dx,302h
        mov al,07h
        out dx,al
        mov dx,300h
lp4:    in al,dx
        cmp al,00h
        je lp4
        mov cx,08h
lp:     rol al,01h
        jc lp5
        loop lp
lp5:    mov row,cl
        mov cx,03h
        mov bh,04h
lp6:    mov al,bh
        mov dx,302h
        out dx,al
        mov dx,300h
        in al,dx
        ror bh,01h
        dec cx
        cmp al,00h
        je lp6
        mov col,cl
        push cx
        push bx
        mov bx,11h
lp7:    loop lp7
        dec bx
        jnz lp7
        pop bx
        pop cx
        ret
        read endp

        cvt proc
        mov ah,00h
        mov al,col
        mov bl,08h
        mul bl
        add al,row
        ret
        cvt endp

        chop proc
        dec al
        cmp al,0ah
        jnb lp8
        clc
        jmp rtn
lp8:    stc
rtn:    ret
        chop endp

        chor proc
        cmp al,0ch
```

```
        jb lp9
        cmp al,10h
        jnb lp9
        stc
        jmp rtn1
lp9:    clc
rtn1:   ret
        chor endp

        disp proc
        lea dx,rst
        mov ah,09h
        int 21h
        ret
        disp endp
end start
```

12a.System time

Title System time

.model small

.data

msg db "system time is:","\$"

.code

```
start:  mov ax,@data
        mov ds,ax
        mov ah,09h
        lea dx,msg
        int 21h
        mov ah,2ch
        int 21h
        mov bl,0ah
        mov al,ch
        call disp
        mov al,cl
        call disp
        mov al,dh
        call disp1
        mov ah,4ch
        int 21h
```

```
disp proc
call disp1
mov dl,':'
mov ah,02h
int 21h
ret
disp endp
```

```
disp1 proc
mov ah,00h
div bl
mov dl,'0'
```

```
xchg al,ah
add dl,ah
mov ah,02h
push ax
int 21h
pop ax
mov dl,al
add dl,'0'
int 21h
ret
disp1 endp
end start
```

12b.Sine wave using DAC

Title Sine wave

```
.model small
```

```
.data
```

```
sin db 00h,16h,2bh,40h,51h,61h,6dh,77h,7dh,7fh
```

```
.code
```

```
start: mov ax,@data
       mov ds,ax
       mov al,80h
       mov dx,303h
       out dx,al
       mov dx,300h
       mov bx,00h
lp1:   mov al,sin [bx]
       add al,80h
       out dx,al
       inc bx
       cmp bx,09h
       jb lp1
lp2:   mov al,sin[bx]
       add al,80h
       out dx,al
       dec bx
       cmp bx,00h
       jne lp2
lp3:   mov al,80h
       sub al,sin[bx]
       out dx,al
       inc bx
       cmp bx,09h
       jb lp3
lp4:   mov al,80h
       sub al,sin[bx]
       out dx,al
       dec bx
       cmp bx,00h
       jne lp4
       loop lp1
       mov ah,4ch
```

```
        int 21h
end start
```

13a.Decimal upcounter

Title Decimal up counter

```
.model small

.data
cnt db 64h
msg db "BCD upcounter"
cr db 13,10,'$'

.code
start: mov ax,@data
        mov ds,ax
        lea dx,msg
        mov ah,09h
        int 21h
        mov cl,cnt
        mov al,00h
lp1:    call disp
        loop lp1
        mov ah,4ch
        int 21h

        disp proc
        mov al,64h
        sub al,cl
        mov bl,0ah
        mov ah,00h
        div bl
        xchg al,ah
        mov dl,ah
        add dl,'0'
        mov ah,02h
        push ax
        int 21h
        pop ax
        mov dl,al
        add dl,'0'
        int 21h
        mov dl,0dh
        int 21h
        push cx
        mov bx,01aah
lp:     loop lp
        dec bx
        jnz lp
        pop cx
        ret
        disp endp
end start
```

13b. Half rectified sine wave using DAC

Title Half rectified sine wave

```
.model small
```

```
.data
```

```
sin db 00h,16h,2bh,40h,51h,61h,6dh,77h,7dh,7fh
```

```
.code
```

```
start: mov ax,@data
      mov ds,ax
      mov al,80h
      mov dx,303h
      out dx,al
      mov dx,300h
      mov bx,00h
lp1:   mov al,sin [bx]
      add al,80h
      out dx,al
      inc bx
      cmp bx,09h
      jb lp1
lp2:   mov al,sin[bx]
      add al,80h
      out dx,al
      dec bx
      cmp bx,00h
      jne lp2
lp3:   mov al,80h
      sub al,00h
      out dx,al
      inc bx
      cmp bx,09h
      jb lp3
lp4:   mov al,80h
      sub al,00h
      out dx,al
      dec bx
      cmp bx,00h
      jne lp4
      loop lp1
      mov ah,4ch
      int 21h
end start
```

14a. Move to the specified co-ordinate on screen

Title Move to the specified co-ordinate on screen

```
.model small
```

```
.data
```

```
col db 13,10,"Enter column no(BCD) $"
```

```
cl1 db ?,?
```

```
row db 13,10,"Enter row no(BCD) $"
```

```
rw db ?,?  
msg db 01h,"You are here$"  
bin db ?,?
```

```
.code
```

```
start: mov ax,@data  
       mov ds,ax  
       lea dx,row  
       mov ah,09h  
       int 21h  
       call read  
       mov rw,cl  
       mov rw+1,al  
       lea dx,col  
       mov ah,09h  
       int 21h  
       call read  
       mov cl,cl  
       mov cl+1,al  
       call cvt  
       mov ah,06h  
       mov al,00h  
       mov bh,07h  
       mov cx,00h  
       mov dx,1850h  
       int 10h  
       mov ah,02h  
       mov bh,00h  
       mov dh,bin  
       mov dl,bin+1  
       int 10h  
       lea dx,msg  
       mov ah,09h  
       int 21h  
       mov bx,0h  
lp:    loop lp  
       dec bx  
       jnz lp  
       mov ah,4ch  
       int 21h  
  
cvt proc  
       mov al,rw  
       mov ah,00h  
       mov bl,0ah  
       mul bl  
       mov ah,rw+1  
       add al,ah  
       mov bin,al  
       mov al,cl  
       mov ah,00h  
       mov bl,0ah  
       mul bl  
       mov ah,cl+1  
       add al,ah  
       mov bin+1,al  
       ret  
cvt endp
```

```
    read proc
    mov ah,01h
    int 21h
    sub al,'0'
    mov cl,al
    int 21h
    sub al,'0'
    ret
    read endp
end start
```

14b.Full rectified sine wave

Title Full rectified sine wave

```
.model small
```

```
.data
```

```
sin db 00h,16h,2bh,40h,51h,61h,6dh,77h,7dh,7fh
```

```
.code
```

```
start: mov ax,@data
       mov ds,ax
       mov al,80h
       mov dx,303h
       out dx,al
       mov dx,300h
       mov bx,00h
lp1:   mov al,sin [bx]
       add al,80h
       out dx,al
       inc bx
       cmp bx,09h
       jb lp1
lp2:   mov al,sin[bx]
       add al,80h
       out dx,al
       dec bx
       cmp bx,00h
       jne lp2
       loop lp1
       mov ah,4ch
       int 21h
end start
```

15a.Create and delete a file

Title Program to create and delete a file

```
.model small
```

```
.data
```

```
ent db 13,10,"Enter a file name",13,10,'$'
crt db 50 dup(0)
```



```
del db 50 dup(0)
cr db 13,10,"File creation successful$"
crf db 13,10,"File creation unsuccessful$"
dl1 db 13,10,"File deletion successful$"
dlf db 13,10,"File deletion unsuccessful$"

.code
start: mov ax,@data
        mov ds,ax
        lea dx,ent
        mov ah,09h
        int 21h
        lea si,crt
        call read
        lea dx,ent
        mov ah,09h
        int 21h
        lea si,del
        call read
        mov cx,00h
        clc
        lea dx,crt
        mov ah,3ch
        int 21h
        jc er
        lea dx,cr
        jmp disp
er:      lea dx,crf
disp:    mov ah,09h
        int 21h
        clc
        mov cx,00h
        mov ah,41h
        lea dx,del
        int 21h
        jc err1
        lea dx,dl1
        jmp disp1
err1:    lea dx,dlf
disp1:   mov ah,09h
        int 21h
        mov ah,4ch
        int 21h

        read proc
        mov bx,00h
lp1:     mov ah,01h
        int 21h
        cmp al,0dh
        je rtn
        cmp al,08h
        je bck
        mov si[bx],al
        inc bx
        jmp lp1
bck:     mov dl,' '
        mov ah,02h
        int 21h
        mov dl,08h
```

```
        int 21h
        cmp bx,00h
        je lp1
        dec bx
        mov ah,00h
        mov si[bx],ah
        jmp lp1
rtn:    ret
        read endp
end start
```

15b.Elevator

Title Elevator

```
.model small
```

```
.data
clr db 0e0h,0d3h,0b6h,079h
```

```
.code
start:  mov ax,@data
        mov ds,ax
        mov al,82h
        mov dx,303h
        out dx,al
        mov al,00h
        mov dx,300h
        out dx,al
        mov al,0f0h
        out dx,al
        mov dx,301h
lp:      in al,dx
        and al,0fh
        cmp al,0fh
        je lp
        mov cx,00h
lp1:     ror al,01h
        inc cx
        jc lp1
        dec cx
        call ele
        mov ah,4ch
        int 21h
```

```
        ele proc
        push cx
        mov al,cl
        mov cl,03h
        mov ah,00h
        mul cl
        mov cx,ax
        mov dx,300h
        mov al,0f0h
lp2:     cmp cx,00h
        je lp3
```

```
        out dx,al
        inc al
        call delay1
        dec cx
        jmp lp2
lp3:    pop bx
        mov al,clr[bx]
        push bx
        out dx,al
        or al,0f0h
        out dx,al
        mov al,bl
        mov ah,00h
        mov cl,03h
        mul cl
        or al,0f0h
        mov cl,bl
lp4:    cmp cl,00h
        je rtn
        dec al
        out dx,al
        call delay1
        dec cl
        jmp lp4
rtn:    ret
        ele endp

        delay1 proc
        push cx
        push bx
        mov bx,00aah
lp5:    loop lp5
        dec bx
        jnz lp5
        pop bx
        pop cx
        ret
        delay1 endp
end start
```

Note: The ports used for part B are as below please change each occurrence of these ports to the ones in your college.

Port A: 300h

Port B: 301h

Port C:302h

Control word register: 303h

Create a free website with