System Programming Lab Assignment-2

Name: Ritabroto Ganguly
Roll: 001910501090

1. Write and test a MASM program to add and subtract two 16 bit numbers.

```
; Add and Subtract two 16 bit numbers

new_line macro

        mov ah,02h

        mov dl,0dh

        int 21h

        mov dl,0ah

        int 21h

endm


;macro to print space

space macro

        mov ah,02h

        mov dl,' '

        int 21h

endm


;macro to print a message

printm macro mess

        lea dx,mess

        mov ah,09h

        int 21h
```

```
endm

;macro to exit
exitp macro
        mov ah,4ch
        int 21h
endm

;macro for hex input
hex_input macro
        local skip,input,letter,shift
        ; output: bx
        xor bx,bx
        mov ah,01h
        int 21h
        cmp al,0dh
        je skip
        input:
                xor ah,ah
                cmp ax,'A'
                jge letter
                sub ax,'0'
                jmp shift
                letter:
                        sub ax,55
```

```
            shift:
                shl bx,1
                shl bx,1
                shl bx,1
                shl bx,1
            or bx,ax
            ; take input
            mov ah,01h
            int 21h
            cmp al,0dh
            jne input
        skip:
endm

;macro for hex_output
hex_output macro
        local output,display_loop,letter,line
        ; input: bx
        mov ah,02h
        mov cx,0
        output:
            mov dx,bx
            and dx,0fh
            cmp dx,10
            jge letter
```

```
            add dx,'0'
            jmp line
        letter:
                add dx,55
        line:
            push dx
            inc cx
            shr bx,1
            shr bx,1
            shr bx,1
            shr bx,1
        jnz output
        mov cx,cx
        display_loop:
            pop dx
            int 21h
        loop display_loop
endm

.model small
.stack 100h
.data
    inpmsg1 db "Enter 1st number in hex: $"
    inpmsg2 db "Enter 2nd number in hex: $"
    oupmsg1 db "Their sum in hex is: $"
```

```
        oupmsg2 db "Their difference in hex is: $"
         num1 dw ?
         num2 dw ?
.code
         main proc
                  mov ax,@data
                  mov ds,ax
                  ;input prompt
                  printm inpmsg1
                  hex_input
                  mov num1,bx
                  printm inpmsg2
                  hex_input
                  mov num2,bx
                  ;calculating sum
                  printm oupmsg1
                  mov cx,num1
                  add bx,cx
                  jnc display
                  carry_disp:
                           ;display carry
                           mov ah,02h
                           mov dl,'1'
                           int 21h
                  display:
```

```
                hex_output
        ;calculating difference
        new_line
        printm oupmsg2
        mov bx,num1
        mov cx,num2
        sub bx,cx
        hex_output
        exitp
    main endp
end main
```

```
C:\ASS2>a2q1.exe
Enter 1st number in hex: 12
Enter 2nd number in hex: 34
Their sum in hex is: 46
Their difference in hex is: FFDE
```

2. Write and test a MASM program to convert Binary digit to Decimal and vice versa.

```
;Program to Convert a Binary digit to Decimal and vice versa
printm macro mess
    lea dx,mess
    mov ah,09h
    int 21h
endm
```

```
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

dec_input macro
        local input,skip
        ; output: bx
        xor bx,bx
        mov ah,01h
        int 21h
        ;if \r
        cmp al,0dh
        je skip
        input:
                and ax,000fh
                push ax
                ; bx=bx*10+ax
                mov ax,10
                mul bx
                mov bx,ax
```

```
            pop ax
            add bx,ax
            ; take input
            mov ah,01h
            int 21h
            cmp al,0dh
            jne input
        skip:
endm


dec_output macro
        local start,repeat,display
        start:                  ; jump label
            mov ax, bx              ; set ax=bx
            xor cx, cx              ; clear cx
            mov bx, 10              ; set bx=10
        repeat:                 ; loop label
            xor dx, dx              ; clear dx
            div bx                  ; divide ax by bx
            push dx                 ; push dx onto the stack
            inc cx                  ; increment cx
            or ax, ax               ; take or of ax with ax
            jne repeat              ; jump to label repeat if zf=0
            mov ah, 2               ; set output function
```

```
        display:                ; loop label
                pop dx                  ; pop a value from stack to dx
                or dl, 30h              ; convert decimal to ascii code
                int 21h                 ; print a character
                loop display
endm

bin_input macro
        local skip,input
        ; output: bx
        xor bx,bx
        mov ah,01h
        int 21h
        cmp al,0dh
        je skip
        input:
                xor ah,ah
                sub ax,'0'
                shl bx,1
                or bx,ax
                ; take input
                mov ah,01h
                int 21h
                cmp al,0dh
                jne input
```

```
        skip:
endm

; macro to take binary output
bin_output macro
        local output,display_loop
        ; input: bx
        mov ah,02h
        mov cx,0
        output:
                mov dx,bx
                and dx,01h
                add dx,'0'
                push dx
                inc cx
                shr bx,1
        jnz output
        mov cx,cx
        display_loop:
                pop dx
                int 21h
        loop display_loop
endm

exitp macro
```

```
        mov ah,4ch
        int 21h
endm

.model small
.stack 100h
.data
        inpmsg1 db "Enter binary number: $"
        inpmsg2 db "Enter decimal number: $"
        oupmsg1 db "Equivalent decimal number: $"
        oupmsg2 db "Equivalent binary number: $"
.code
        main proc
                mov ax,@data
                mov ds,ax
                ;binary to decimal
                ;input
                printm inpmsg1
                bin_input
                ;output
                printm oupmsg1
                dec_output
                new_line
                ;decimal to binary
                ;input
```

```
        printm inpmsg2
        dec_input
        ;output
        printm oupmsg2
        bin_output
        exitp
    main endp
end main
```

```
C:\ASS2>a2q2.exe
Enter binary number: 1010
Equivalent decimal number: 10
Enter decimal number: 9
Equivalent binary number: 1001
```

3. Write and test a program to print pairs of even numbers where the summation of the numbers in each pair is 100.

```
;Program to print pairs of even numbers where the sum of the numbers in each pair is 100.
printm macro mess
    lea dx,mess
    mov ah,09h
    int 21h
endm

new_line macro
    mov ah,02h
```

```
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

; macro for decimal output
dec_output macro
        local start,repeat,display
        start:                  ; jump label
                mov ax, bx              ; set ax=bx
                xor cx, cx              ; clear cx
                mov bx, 10              ; set bx=10
        repeat:                 ; loop label
                xor dx, dx              ; clear dx
                div bx                  ; divide ax by bx
                push dx                 ; push dx onto the stack
                inc cx                  ; increment cx
                or ax, ax               ; take or of ax with ax
                jne repeat              ; jump to label repeat if zf=0
                mov ah, 2               ; set output function
        display:                ; loop label
                pop dx                  ; pop a value from stack to dx
                or dl, 30h              ; convert decimal to ascii code
                int 21h                 ; print a character
```

```
                loop display
endm

exitp macro
        mov ah,4ch
        int 21h
endm

space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm



.model small
.stack 100h
.data
        oupmsg db "Even pairs with sum 100: $"
        tempb dw ?
        tempc dw ?
.code
        main proc
                mov ax,@data
                mov ds,ax
```

```
        printm oupmsg
        new_line
        mov bx,0
        mov cx,100
        @print_loop:
            mov tempb,bx
            mov tempc,cx
            dec_output
            space
            mov bx,tempc
            dec_output
            new_line
            mov bx,tempb
            mov cx,tempc
            inc bx
            inc bx
            dec cx
            dec cx
            cmp bx,50
        jle @print_loop
        exitp
    main endp
end main
```

```
Even pairs with sum 100:
0 100
2 98
4 96
6 94
8 92
10 90
12 88
14 86
16 84
18 82
20 80
22 78
24 76
26 74
28 72
30 70
32 68
34 66
36 64
38 62
40 60
42 58
44 56
46 54
48 52
50 50
```

4. Write and test a MASM program to multiply two 32 bit numbers.

```
;Program to multiply two numbers
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

dec_input macro
        local input,skip
```

```
        ; output: bx
        xor bx,bx
        mov ah,01h
        int 21h
        ;if \r
        cmp al,0dh
        je skip
        input:
                and ax,000fh
                push ax
                ; bx=bx*10+ax
                mov ax,10
                mul bx
                mov bx,ax
                pop ax
                add bx,ax
                ; take input
                mov ah,01h
                int 21h
                cmp al,0dh
                jne input
        skip:
endm

dec_output macro
```

```
        local start,repeat,display
        start:                  ; jump label
                mov ax, bx              ; set ax=bx
                xor cx, cx              ; clear cx
                mov bx, 10              ; set bx=10
        repeat:                 ; loop label
                xor dx, dx              ; clear dx
                div bx                  ; divide ax by bx
                push dx                 ; push dx onto the stack
                inc cx                  ; increment cx
                or ax, ax               ; take or of ax with ax
                jne repeat              ; jump to label repeat if zf=0
                mov ah, 2               ; set output function
        display:                ; loop label
                pop dx                  ; pop a value from stack to dx
                or dl, 30h              ; convert decimal to ascii code
                int 21h                 ; print a character
                loop display
endm

exitp macro
        mov ah,4ch
        int 21h
endm
```

```
.model small
.stack 100h
.data
        inpmsg1 db "Enter 1st number: $"
        inpmsg2 db "Enter 2nd number: $"
        oupmsg db "Product: $"
        num1 db ?
        num2 db ?
.code
        main proc
                mov ax,@data
                mov ds,ax
                xor bh,bh
                ;input prompt
                printm inpmsg1
                dec_input
                mov num1,bl
                xor bh,bh
                printm inpmsg2
                dec_input
                mov num2,bl
                xor bh,bh
                xor ah,ah
                mov al,num1
                mul bx
```

```
            mov bx,ax
            printm oupmsg
            dec_output
            exitp
        main endp
end main
```

```
C:\ASS2>a2q4.exe
Enter 1st number: 12
Enter 2nd number: 34
Product: 408
```

5. Write and test a MASM program to divide a 16 bit number by an 8 bit number.

```
;Program to divide a 16 bit number by a 8 bit number.
;include mtab.asm

printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

;macro for hex input
hex_input macro
        local skip,input,letter,shift
```

```
; output: bx
xor bx,bx
mov ah,01h
int 21h
cmp al,0dh
je skip
input:
        xor ah,ah
        cmp ax,'A'
        jge letter
        sub ax,'0'
        jmp shift
        letter:
                sub ax,55
        shift:
                shl bx,1
                shl bx,1
                shl bx,1
                shl bx,1
        or bx,ax
        ; take input
        mov ah,01h
        int 21h
        cmp al,0dh
        jne input
```

```
        skip:
endm

exitp macro
        mov ah,4ch
        int 21h
endm

;macro for hex_output
hex_output macro
        local output,display_loop,letter,line
        ; input: bx
        mov ah,02h
        mov cx,0
        output:
                mov dx,bx
                and dx,0fh
                cmp dx,10
                jge letter
                add dx,'0'
                jmp line
        letter:
                        add dx,55
        line:
                push dx
```

```
            inc cx
            shr bx,1
            shr bx,1
            shr bx,1
            shr bx,1
        jnz output
        mov cx,cx
        display_loop:
            pop dx
            int 21h
        loop display_loop
endm

pushall macro
        push ax
        push bx
        push cx
        push dx
endm

popall macro
        pop dx
        pop cx
        pop bx
        pop ax
```

```
endm

.model small
.stack 100h
.data
        inpmsg1 db 10,13,"Enter a 16-bit number in hex: $"
        inpmsg2 db 10,13,"Enter a 8-bit number in hex: $"
        oupmsg1 db 10,13,"Quotient in hex: $"
        oupmsg2 db 10,13,"Remainder in hex: $"
         num1 dw ?
.code
         main proc
                mov ax,@data
                mov ds,ax
                ;input
                printm inpmsg1
                hex_input
                mov num1,bx
                printm inpmsg2
                hex_input
                mov ax,num1
                xor dx,dx
                div bx
                ;output
                mov bx,ax
```

```
            mov num1,dx
            printm oupmsg1
            pushall
            hex_output
            popall
            mov bx,num1
            printm oupmsg2
            pushall
            hex_output
            popall
            exitp
        main endp
end main
```

```
C:\ASS2>a2q5.exe

Enter a 16-bit number in hex: 1234

Enter a 8-bit number in hex: 12

Quotient in hex: 102
Remainder in hex: 10
```

6. Write and test a MASM program to Print Fibonacci series up to 10 terms.

```
;Program to Print Fibonacci series up to 10 terms
printm macro mess
        lea dx,mess
```

```
        mov ah,09h
        int 21h
endm
exitp macro
        mov ah,4ch
        int 21h
endm

new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm

pushall macro
        push ax
        push bx
```

```
        push cx
        push dx
endm

popall macro
        pop dx
        pop cx
        pop bx
        pop ax
endm

dec_output macro
        local start,repeat,display
        start:                  ; jump label
                mov ax, bx              ; set ax=bx
                xor cx, cx              ; clear cx
                mov bx, 10              ; set bx=10
        repeat:                 ; loop label
                xor dx, dx              ; clear dx
                div bx                  ; divide ax by bx
                push dx                 ; push dx onto the stack
                inc cx                  ; increment cx
                or ax, ax               ; take or of ax with ax
                jne repeat              ; jump to label repeat if zf=0
                mov ah, 2               ; set output function
```

```
        display:                ; loop label
                pop dx                  ; pop a value from stack to dx
                or dl, 30h              ; convert decimal to ascii code
                int 21h                 ; print a character
                loop display
endm



.model small
.stack 100h
.data
        msg db "The fibonacci series upto 10 terms is: $"
        f1 dw 1
        f2 dw 1
        f3 dw ?
.code
        main proc
                mov ax,@data
                mov ds,ax
                mov bx,1
                mov dx,1
                printm msg
                new_line
                pushall
```

```
        dec_output
        space
        popall
        pushall
        dec_output
        space
        popall
        mov bx,1
        mov dx,1
        mov cx,8
        @loop_fibo:
            mov f1,bx
            mov f2,dx
            add bx,dx
            mov f3,bx   ;f3=f1+f2
            pushall
            dec_output
            space
            popall
            mov bx,f2   ;f1=f2
            mov dx,f3   ;f2=f3
        loop @loop_fibo
        exitp
    main endp
end main
```

```
C:\ASS2>a2q6.exe
The fibonacci series upto 10 terms is:
1 1 2 3 5 8 13 21 34 55
```

7. Write and test a MASM program for substring deletion from a given string.

```
;Program for substring deletion from a given string
.model medium
.stack 100h
.data
        prompt_1    db 10,13,'Enter the string : $'
        prompt_2    db 10,13,'Enter the substring to be deleted : $'
        prompt_3    db 10,13,'The final string is : $'
        newline     db 10,13,'$'
        ;input string
        buffersize_1        db 51                           ; 50 char + return
        inputlength_1       db 0                            ; number of read
characters
        string          db 51 dup(0)                ; actual buffer
        end_1                       db '$'
        index1                      db 0                            ;index for looping
        ;input substring
        buffersize_2        db 21                           ; 20 char + return
        inputlength_2       db 0                            ; number of read
characters
        substring           db 21 dup(0)                ; actual buffer
```

```
index2                  db 0                    ;index for looping
;modified output string
index3              db 0                        ;index for looping
newstring   db 50 dup('$')
;macro to display prompt and print string
display macro msg
        mov ah,9
        lea dx,msg
        int 21h
endm

;macro for string input
get_string macro buffer_
        mov dx, offset buffer_          ; load our pointer to the beginning of
the structure
        mov ah, 0ah                             ; getline function
        int 21h
        mov si, offset buffer_ + 1      ;move pointer to the input string size
        mov cl, [ si ]                          ;move input string size to cl
        mov ch, 0                       ;clear ch to use cx
        inc cx
        add si, cx                              ;move pointer to the next byte
of the last input
        mov al, '$'
        mov [ si ], al                          ;add '$' after the input string
endm
```

```
;macro for copying character from input string to output string
string_copy macro
        mov di,offset newstring         ; load our pointer to the beginning of
the structure
        mov al,index3
        xor ah,ah                       ;load the index in ax register
        add di,ax                       ;go to the next location where
the character is to be copied
        mov dl,[ si ]
        mov [ di ],dl                   ;copy from input string to
output string
        inc al
        mov index3,al                   ;increment the index
endm


;macro  to check whether two character of the input string and substring
are same or not
compare macro
        mov dl,[ si ]                   ; load the character of input
string in dl
        mov di,offset substring
        mov al,index2
        mov ah,ah
        add di,ax
        mov dh,[ di ]                   ; load the character of input
substring in dh
        cmp dl,dh                       ; compare dl and dh
```

```
        endm


.code
        main proc
                mov ax,@data
                mov ds,ax
                display prompt_1
                get_string buffersize_1                         ; input the string
                display prompt_2
                get_string buffersize_2                         ; input the substring
                mov si,offset string                            ; load our pointer to the
beginning of the structure
                mov cl,inputlength_1                            ; move length of the
string in cl
                @loop1:
                        mov di,offset substring         ; load our pointer to the
beginning of the structure
                        mov index2,0
                        string_copy
                        compare
                        jne @label1
                        mov bl,inputlength_2
                        xor bh,bh
                        dec bx
                                @loop2:
                                        inc si
```

```
                    dec cl
                    inc index2
                    string_copy
                    compare
                    jne     @label1
                    dec bl
                    jne @loop2
                ;if the substring is present
                mov bl,inputlength_2    ;move substring length to bl
                mov al,index3                   ; move new string index
to al
                sub al,bl                       ; subtract bl from al
                mov index3,al                   ; save al in new string
index
        @label1:
        inc si
        loop @loop1
        @print:
        string_copy                             ; add '$' after the output
string
        display prompt_3
        display newstring                       ; display the output string
        mov ah,4ch
        int 21h
    main endp
end main
```

```
C:\ASS2>a2q7.exe

Enter the string : hello
Enter the substring to be deleted : ell
The final string is : ho
```

8. Write and test a MASM program to identify the GCD and LCM of three numbers.

```
;Program to identify the GCD and LCM of three numbers

new_line macro

        mov ah,02h

        mov dl,0dh

        int 21h

        mov dl,0ah

        int 21h

endm


printm macro mess

        lea dx,mess

        mov ah,09h

        int 21h

endm


;macro to exit

exitp macro

        mov ah,4ch
```

```
        int 21h
endm

dec_input macro
        local input,skip
        ; output: bx
        xor bx,bx
        mov ah,01h
        int 21h
        ;if \r
        cmp al,0dh
        je skip
        input:
                and ax,000fh
                push ax
                ; bx=bx*10+ax
                mov ax,10
                mul bx
                mov bx,ax
                pop ax
                add bx,ax
                ; take input
                mov ah,01h
                int 21h
                cmp al,0dh
```

```
            jne input
        skip:
endm

; macro for decimal output
dec_output macro
        local start,repeat,display
        start:                  ; jump label
            mov ax, bx              ; set ax=bx
            xor cx, cx              ; clear cx
            mov bx, 10              ; set bx=10
        repeat:                 ; loop label
            xor dx, dx              ; clear dx
            div bx                  ; divide ax by bx
            push dx                 ; push dx onto the stack
            inc cx                  ; increment cx
            or ax, ax               ; take or of ax with ax
            jne repeat              ; jump to label repeat if zf=0
            mov ah, 2               ; set output function
        display:                ; loop label
            pop dx                  ; pop a value from stack to dx
            or dl, 30h              ; convert decimal to ascii code
            int 21h                 ; print a character
            loop display
endm
```

```
.model small
.stack 100h
.data
        inpmsg1 db 10,13,"Enter 1st number: $"
        inpmsg2 db 10,13,"Enter 2nd number: $"
        inpmsg3 db 10,13,"Enter 3rd number: $"
        oupmsg1 db 10,13,"GCD: $"
        oupmsg2 db 10,13,"LCM: $"
        num1 dw ?
        num2 dw ?
        num3 dw ?
        gcdn dw ?
        lcmn dw ?
.code
        ;procedure to find gcd of two numbers
        gcd proc
        ; input: bx,ax
        ; output: gcd
        ; Assumption: cx is greater than bx
          up:
                cmp ax,bx       ;compare the two numbers.
                je exit         ;if equal, go to exit label.
                jb excg         ;if first number is below than second,
                                        ;go to excg label.
```

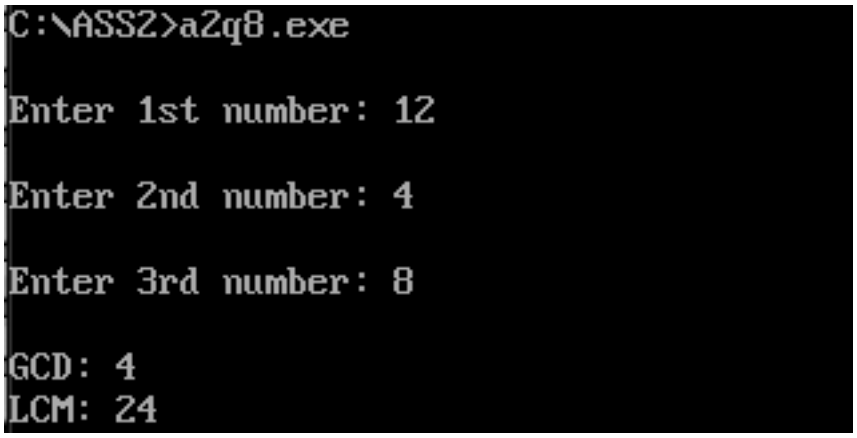
```
  up1:
        mov dx,0h       ;initialize the dx.
        div bx          ;divide the first number by second number.
        cmp dx,0        ;compare remainder is zero or not.
        je exit         ;if zero, jump to exit label.
        mov ax,dx       ;if non-zero, move remainder to ax.
        jmp up
                        ;jump to up label.
  excg:
        xchg ax,bx      ;exchange the remainder and quotient.
        jmp up1         ;jump to up1.
  exit:
        mov gcdn,bx     ;store the result in gcd.
        ret
gcd endp

lcm proc
; input: bx,ax
; output: dx
      xor dx,dx
      mul bx        ;product of numbers
      div gcdn      ;by gcd of numbers
      mov lcmn,ax
      ret
lcm endp
```

```
main proc
        mov ax,@data
        mov ds,ax
        ;input
        ; first number
        printm inpmsg1
        dec_input
        mov num1,bx
        ;second number
        printm inpmsg2
        dec_input
        mov num2,bx
        ;third number
        printm inpmsg3
        dec_input
        mov num3,bx
        ; finding GCD
        ;find GCD of first two
        mov ax,num1
        mov bx,num2
        call gcd
        ; GCD of GCD and third
        mov ax,num3
        mov bx,gcdn
```

```
call gcd
;output
printm oupmsg1
mov bx,gcdn
dec_output
; finding LCM
; find gcd of two
mov ax,num1
mov bx,num2
call gcd
; find lcm
mov ax,num1
mov bx,num2
call lcm
; find gcd of two
mov ax,lcmn
mov bx,num3
call gcd
; find lcm
mov ax,lcmn
mov bx,num3
call lcm
;output
printm oupmsg2
mov bx,lcmn
```

```
            dec_output
            exitp
        main endp
end main
```



9. Write and test a MASM program to Implement Linear search and Binary Search.

<u>Binary Search</u>

```
;Program to implement binary search
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm
```

```
;macro to print space
space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm

;macro to print a message
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

;macro to exit
exitp macro
        mov ah,4ch
        int 21h
endm

; macro for decimal input
dec_input macro
        local input,skip
        ; output: bx
```

```
        xor bx,bx
        mov ah,01h
        int 21h
        ;if \r
        cmp al,0dh
        je skip
        input:
                and ax,000fh
                push ax
                ; bx=bx*10+ax
                mov ax,10
                mul bx
                mov bx,ax
                pop ax
                add bx,ax
                ; take input
                mov ah,01h
                int 21h
                cmp al,0dh
                jne input
        skip:
endm

; macro for decimal output
dec_output macro
```

```
        local start,repeat,display
        start:                  ; jump label
                mov ax, bx              ; set ax=bx
                xor cx, cx              ; clear cx
                mov bx, 10              ; set bx=10
        repeat:                 ; loop label
                xor dx, dx              ; clear dx
                div bx                  ; divide ax by bx
                push dx                 ; push dx onto the stack
                inc cx                  ; increment cx
                or ax, ax               ; take or of ax with ax
                jne repeat              ; jump to label repeat if zf=0
                mov ah, 2               ; set output function
        display:                ; loop label
                pop dx                  ; pop a value from stack to dx
                or dl, 30h              ; convert decimal to ascii code
                int 21h                 ; print a character
                loop display
endm

pushall macro
        push ax
        push bx
        push cx
        push dx
```

```
endm

popall macro
        pop dx
        pop cx
        pop bx
        pop ax
endm

.model small
.stack 100h
.data
        inpmsg1 db 10,13,"Enter size of array: $"
        inpmsg2 db 10,13,"Enter elements of array in sorted order: $"
        inpmsg3 db 10,13,"Enter element to be searched: $"
        oupmsg1 db 10,13,"element found at: $"
        oupmsg2 db 10,13,"element not found $"
        arr dw 50 dup(?)
        s dw ?
        start dw ?
        stop dw ?
        min_idx dw ?
        temp dw ?
.code
        main proc
```

```
mov ax,@data
mov ds,ax
;accept size
printm inpmsg1
dec_input
;accept elements
mov s,bx
lea si,arr
mov cx,bx
printm inpmsg2
new_line
@array_input:
        pushall
        dec_input
        mov word ptr[si],bx
        popall
        inc si
        inc si
loop @array_input
call sort
; enter element to search
printm inpmsg3
dec_input
lea si,arr
mov cx,s
```

```
dec cx
mov start,00h
mov stop,cx
;binary search
@binary_search:
        ;find out the middle index
        lea si,arr
        mov cx,stop
        add cx,start
        shr cx,1            ;cx is the index for the middle element
        add si,cx           ;si=si+cx
        add si,cx
        push bx
        push cx
        mov bx,cx
        pop cx
        pop bx
        space
        push bx
        push cx
        mov bx,word ptr[si]
        pop cx
        pop bx
        cmp bx,word ptr[si]
        je @found           ; if middle element then found
```

```
                jg @greater
                ;if less
                @lesser:
                        dec cx
                        mov stop,cx
                        jmp @compare
                @greater:
                        inc cx
                        mov start,cx
                @compare:
                        mov cx,stop
                        cmp cx,start
        jge @binary_search
        ;if not found
        printm oupmsg2
        jmp @exit
        @found:
                printm oupmsg1
                mov bx,cx
                inc bx
                dec_output
        @exit:
                exitp
main endp
sort proc
```

```
;Selection sort used
lea si,arr
mov cx,s
dec cx
@outer_loop:
        mov dx,cx                       ; dx is the inner loop counter
        mov di,si
        inc di
        inc di
        mov min_idx,si
        push si
        @inner_loop:
                mov si,min_idx
                mov bx,word ptr[si]
                cmp word ptr[di],bx
                jge @incr
                ; else set min_idx the elements
                mov min_idx,di
                @incr:
                inc di
                inc di
                dec dx
        jnz @inner_loop
        ;swap
        pop si
```

```
                mov di,min_idx
                mov bx,word ptr[di]
                xchg word ptr[si],bx
                mov word ptr[di],bx
                inc si
                inc si
                push si
                push cx
                pop cx
                pop si
            loop @outer_loop
            ret
        sort endp
end main
```

```
C:\ASS2>a2q9-bs.exe

Enter size of array: 5

Enter elements of array in sorted order:
3
4
5
6
7

Enter element to be searched: 6

element found at: 4
```

Linear Search

```
;Program to Implement Linear search.
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

;macro to print space
space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm

;macro to print a message
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm
```

```
;macro to exit
exitp macro
        mov ah,4ch
        int 21h
endm
pushall macro
        push ax
        push bx
        push cx
        push dx
endm

popall macro
        pop dx
        pop cx
        pop bx
        pop ax
endm

; macro for decimal input
dec_input macro
        local input,skip
        ; output: bx
        xor bx,bx
        mov ah,01h
```

```
        int 21h
        ;if \r
        cmp al,0dh
        je skip
        input:
                and ax,000fh
                push ax
                ; bx=bx*10+ax
                mov ax,10
                mul bx
                mov bx,ax
                pop ax
                add bx,ax
                ; take input
                mov ah,01h
                int 21h
                cmp al,0dh
                jne input
        skip:
endm

; macro for decimal output
dec_output macro
        local start,repeat,display
        start:                  ; jump label
```

```
        mov ax, bx              ; set ax=bx
        xor cx, cx              ; clear cx
        mov bx, 10              ; set bx=10
    repeat:                     ; loop label
        xor dx, dx              ; clear dx
        div bx                  ; divide ax by bx
        push dx                 ; push dx onto the stack
        inc cx                  ; increment cx
        or ax, ax               ; take or of ax with ax
        jne repeat              ; jump to label repeat if zf=0
        mov ah, 2               ; set output function
    display:                    ; loop label
        pop dx                  ; pop a value from stack to dx
        or dl, 30h              ; convert decimal to ascii code
        int 21h                 ; print a character
        loop display
endm
.model small
.stack 100h
.data
    inpmsg1 db 10,13,"Enter size of array: $"
    inpmsg2 db 10,13,"Enter elements of array: $"
    inpmsg3 db 10,13,"Enter element to be searched: $"
    oupmsg1 db 10,13,"element found at: $"
    oupmsg2 db 10,13,"element not found $"
```

```
        arr dw 50 dup(?)
        s dw ?
.code
        main proc
                mov ax,@data
                mov ds,ax
                ;input size
                printm inpmsg1
                dec_input
                ;input elements
                printm inpmsg2
                new_line
                mov s,bx
                lea si,arr
                mov cx,bx
                @array_input:
                        pushall
                        dec_input
                        mov word ptr[si],bx
                        popall
                        inc si
                        inc si
                loop @array_input
                ; enter element to search
                printm inpmsg3
```

```
        dec_input
        lea si,arr
        mov cx,s
        @linear_search:
            cmp bx,word ptr[si]
            je @found
            inc si
            inc si
        loop @linear_search
        ;if not found
        printm oupmsg2
        jmp @exit
        @found:
            printm oupmsg1
            mov bx,s
            sub bx,cx
            inc bx
            dec_output
        @exit:
            exitp
    main endp
end main
```

```
C:\ASS2>a2q9-ls.exe

Enter size of array: 5

Enter elements of array:
3
2
1
6
5

Enter element to be searched: 1

element found at: 3
```

10. Write and test a MASM program to print prime numbers between 1 to 100.

```
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

;macro to print space
space macro
        mov ah,02h
        mov dl,' '
        int 21h
```

```
endm

;macro to print a message
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

;macro to exit
exitp macro
        mov ah,4ch
        int 21h
endm

; macro for decimal output
dec_output macro
        local start,repeat,display
        start:                  ; jump label
                mov ax, bx              ; set ax=bx
                xor cx, cx              ; clear cx
                mov bx, 10              ; set bx=10
        repeat:                 ; loop label
                xor dx, dx              ; clear dx
                div bx                  ; divide ax by bx
```

```
        push dx                 ; push dx onto the stack
        inc cx                  ; increment cx
        or ax, ax               ; take or of ax with ax
        jne repeat              ; jump to label repeat if zf=0
        mov ah, 2               ; set output function
    display:                    ; loop label
        pop dx                  ; pop a value from stack to dx
        or dl, 30h              ; convert decimal to ascii code
        int 21h                 ; print a character
        loop display
endm

.model small
.stack 100h
.data
    msg db "Prime numbers from 1 to 100 are: $"
    num db ?
.code
    main proc
        mov ax,@data
        mov ds,ax
        printm msg
        new_line
        mov cl,02h
        start:
```

```
mov num,cl
mov al,cl
mov bl,01h      ; the dividing starts from 2, hence bh is compare to 02h
mov dx,0000h    ; to avoid divide overflow error
mov ah,00h      ; to avoid divide overflow error
mov bh,00h

;loop to check for prime no
l1:
        div bl
        cmp ah,00h      ; remainder is compared with 00h (ah)
        jne next
        inc bh          ; bh is incremented if the number is divisible by current value of bl
        next:
                cmp bh,02h ; if bh > 02h, there is no need to proceed, it is not a prime
                je false        ; the no is not a prime no
                inc bl          ; increment bl
                mov ax,0000h    ; to avoid divide overflow error
                mov dx,0000h    ; to avoid divide overflow error
                mov al,cl       ; move the default no to al
                cmp bl,cl       ; run the loop until bl matches number. i.e, run loop x no of times, where x is the number given
                jne l1          ; jump to check again with incremented value of bl
```

```
                ;to display the given no is a prime no
                true:
                        mov ch,00h
                        mov bx,cx
                        dec_output
                        space
                false:
                        mov cl,num
                        inc cl
                        cmp cl,64h
                        jne start
        exitp
    main endp
end main
```

```
C:\ASS2>a2q10.exe
Prime numbers from 1 to 100 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

11. Write and test a MASM program to perform Insertion and Selection sort.

```
.model small
.stack 100
.data
linefeed    db 13, 10, "$"
prompt1     db "Enter size of array: $"
```

```
prompt2     db "Enter element: $"
msg1        db "Array is: $"
msg2        db "Using Selection Sort: $"
msg3        db "Using Insertion Sort: $"
len db ?
nums db 10 DUP(?), "$"
dec_out db 2 DUP(?), "$"
.code                   ; code segment
call main
mov  ax, 4c00h              ; terminate properly
int  21h
main proc
    mov ax, @data
    mov ds, ax
    call array_input
    call new_line
    call selection_sort
    mov dx, offset msg2
    call show_msg
    call new_line
    call array_output
    call new_line
    call new_line
    call array_input
    call new_line
```

```
    call insertion_sort
    mov dx, offset msg3
    call show_msg
    call new_line
    call array_output
    call new_line
    ret
main endp

; insertion sort
insertion_sort proc
    push ax
    push bx
    push cx
    push dx
    mov cl, 1
    mov bx, offset nums
        ins_outer:
            mov ch, 0
            mov di, cx
            mov dl, nums[di]
            mov si, di
            dec si
        ins_inner:
            cmp si, 0
```

```
            jl ins_outer_update
        cmp nums[si], dl
        jbe ins_outer_update
            mov ch, nums[si]
            mov nums[di], ch
            dec di
            dec si
        jmp ins_inner
        ins_outer_update:
            mov nums[si+1], dl
        inc cl
        cmp cl, len
        jl ins_outer
            pop dx
            pop cx
            pop bx
            pop ax
            ret
insertion_sort endp

; selection sort
selection_sort proc
    push ax
    push bx
    push cx
```

```
push dx
mov cl, len
mov bx, offset nums
    sel_outer:
        mov ch, 0
        inc ch
        mov dh, cl
        mov dl, [bx]
    sel_inner:
        push cx
        xchg cl, ch
        mov ch, 0
        add bx, cx
        mov al, [bx]
        cmp dl, al
        jbe sel_inner_upd
        mov dl, al
        mov dh, cl
    sel_inner_upd:
        sub bx, cx
        pop cx
        inc ch
        cmp ch, cl
        jl sel_inner
    sel_done_inner:
```

```
            mov ah, [bx]
            push bx
            add bl, dh
            adc bh, 0
            mov [bx], ah
            pop bx
            mov [bx], dl
            inc bx
            dec cl
            cmp cl, 1
            jg sel_outer
            pop dx
            pop cx
            pop bx
            pop ax
            ret
selection_sort endp

; get array as input
array_input proc
    push ax
    push bx
    push cx
    push dx
    mov dx, offset prompt1
```

```
    call show_msg
    call get_dec_val
    mov len, al
    call new_line
    mov cx, 0
        get_arr_elems_loop:
            mov bx, offset nums
            add bx, cx
            mov dx, offset prompt2
            call show_msg
            call get_dec_val
            mov [bx], al
            inc cl
            cmp cl, len
            jl get_arr_elems_loop
        done_get_arr_elems:
            pop dx
            pop cx
            pop bx
            pop ax
            ret
array_input endp

array_output proc
    push ax
```

```
    push bx
    push cx
    push dx
    mov cl, 0
    mov bx, offset nums
        array_output_loop:
            mov al, [bx]
            mov ah, 0
            call disp_dec_val
            mov al, 32
            call show_char
            inc bx
            inc cl
            cmp cl, len
            jl array_output_loop
            pop dx
            pop cx
            pop bx
            pop ax
            ret
array_output endp

; get decimal value, store in ax
get_dec_val proc
    push bx
```

```
    push cx
    push dx
    mov dx, 0
        get_characters:
            call get_char
        cmp al, 13 ;cmp w/ [enter]
            je done
            sub al, 48
            mov bx, dx
            mov cl, 3
            shl bx, cl
            shl dx, 1
            add dx, bx
            add dl, al
            jnc get_characters
            add dh, 1
            jmp get_characters
        done:
            mov ax, dx
            pop dx
            pop cx
            pop bx
            ret
get_dec_val endp
```

```
; display ax value in decimal
disp_dec_val proc
    push ax
    push bx
    push cx
    push dx
    mov cl, 2
        disp_dec_val_loop:
            dec cl
            cmp cl, 0
        jl disp_dec_val_loop_done
    mov bx, offset dec_out
    push cx
    mov ch, 0
    add bx, cx
    pop cx

    mov ch, 10
    div ch
    push ax
    add ah, 48
    mov [bx], ah
    pop ax
    mov ah, 0
        jmp disp_dec_val_loop
```

```
        disp_dec_val_loop_done:
                mov dx, offset dec_out
                call show_msg
                pop dx
                pop cx
                pop bx
                pop ax
                ret
disp_dec_val endp

; show character, ascii value in al
show_char proc
    push ax
    push dx
    mov dl, al
    mov ah, 2
    int 21h
    pop dx
    pop ax
    ret
show_char endp

; show message, location in dx
show_msg proc
    push ax
```

```
    mov ah, 9
    int 21h
    pop ax
    ret
show_msg endp

; get a single character, modify ah, store in al
get_char proc
    mov ah, 1
    int 21h
    ret
get_char endp

; insert new-line
new_line proc
    push ax
    push dx
    lea dx,linefeed
    mov ah,9
    int 21h
    pop dx
    pop ax
    ret
new_line endp
end
```

```
C:\ASS2>a2q11.exe
Enter size of array: 5

Enter element: 3
Enter element: 4
Enter element: 5
Enter element: 1
Enter element: 2

Using Selection Sort:
01 02 03 04 05

Enter size of array: 5

Enter element: 6
Enter element: 5
Enter element: 1
Enter element: 2
Enter element: 8

Using Insertion Sort:
01 02 05 06 08
```

12. Write and test a MASM program to rename a file.

```
;macro to print a message
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

;macro to exit
exitp macro
        mov ah,4ch
```

```
        int 21h
endm
.model small
.stack 100h
.data
        oldfilename db "OLD.txt", 0
        newfilename db "NEW.txt", 0
        oupmsg db 10,13,"File renamed $"
.code
        main proc
                mov ax, @data
                mov ds, ax
                mov es, ax
                lea dx,oldfilename
                lea di,newfilename
                mov ah,56h;to rename file
                int 21h
                printm oupmsg
                exitp  ;exit
        main endp
end main
```

13. Write and test a MASM program to print the system time and date.

```
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm

;macro to print space
space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm

;macro to print a message
printm macro mess
        lea dx,mess
        mov ah,09h
        int 21h
endm

;macro to exit
```

```
exitp macro
        mov ah,4ch
        int 21h
endm



.model small
.stack 100h
.data
        oupmsg1 db 10,13,"System time in hh:mm:ss format is $"
        oupmsg2 db 10,13,"System date in dd/mm/yy format is $"
.code
        disp proc       ; Beginning of disp procedure
                aam             ; ASCII adjust after multiplication [ax register]
                mov bx, ax      ; loading adjusted value to bx
                add bx, 3030h ; Add 3030 to properly print the data
                mov dl, bh      ; To print first digit of data
                mov ah, 02h
                int 21h
                mov dl, bl      ; To print second digit of data
                mov ah, 02h
                int 21h
                ret
        disp endp
        main proc
```

```
        mov ax,@data
        mov ds,ax
        printm oupmsg1
mov ah, 2ch     ; To get system time [HH in ch, MM in cl, SS in dh]
int 21h
mov al, ch      ; hour in ch
call disp
mov dl, ':'     ; copy : to dl to print
mov ah, 02h
int 21h
mov al, cl      ; minutes in cl
call disp
mov dl, ':'     ; To print : as above
mov ah, 02h
int 21h
mov al, dh      ; seconds in dh as SS
call disp
new_line
        printm oupmsg1
mov ah, 2Ah     ; To get system date [DD in dl , MM in dh, YYYY in cx]
int 21h
mov al, dl      ; day in dl
call disp
mov dl, '/'     ; To print /
mov ah, 02h
```

```
        int 21h
        mov al, dh      ; month in dh
        call disp
        mov dl, '/'     ; To print /
        mov ah, 02h
        int 21h
        add cx, 0F830h  ; Add 0F830 to adjust hexadecimal effects on year
        mov ax, cx      ; year in ax
        call disp
        exitp
         main endp
end main
```

```
C:\ASS2>a2q13.exe

System time in hh:mm:ss format is 20:50:13

System time in hh:mm:ss format is 02/11/21
```