SYSTEMS PROGRAMMING LAB REPORT

CLASS: UG-III SECTION: A1

GROUP NUMBER: C

GROUP MEMBERS

Bodhisattwa Halder (Roll: 15)

Masud Rahaman Laskar (Roll: 18)

Soumyadeep Barman (Roll: 19)

Anuran Chakraborty (Roll: 20)

Arighna Saha (Roll: 21)

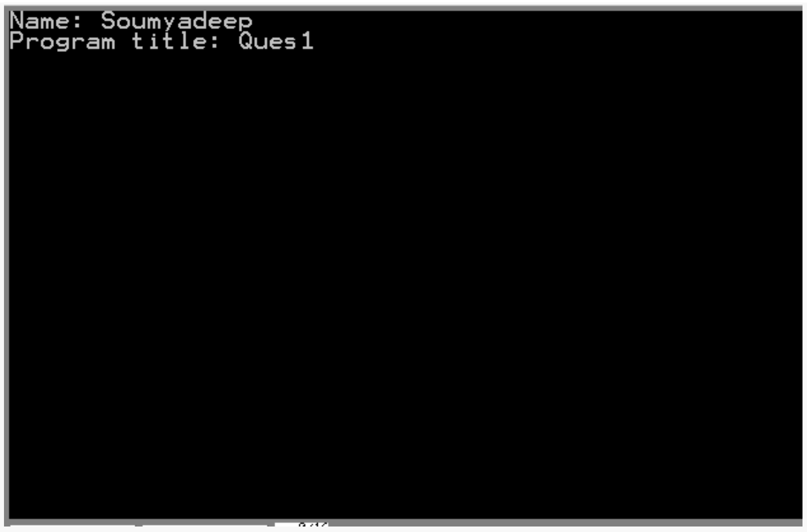
INDEX

ASSIGNMENT 1

1. **Write and test a MASM program to Display your name and program title on the output screen.**

   The name and program title are first defined in the data section as strings. In the main procedure the addresses of the strings are loaded in dx and printed out using interrupt 21h.

```
; Write and test a MASM program to Display your name and program title on the
output screen.
.model small
.stack 100h

.data
        name1 db "Name: Soumyadeep$"
        programTitle db "Program title: Ques1$"
.code
        mov ax,@data
        mov ds,ax

        ;display the name
        lea dx,name1
        mov ah,09h
        int 21h

        ;carriage return
        mov AH, 02h
        mov DL, 0DH
        int 21H

        ;line feed
        mov DL, 0AH
        int 21H

        ;display program title
        lea dx,programTitle
        mov ah,09h
        int 21h

        ;exit
        mov ah,4Ch
        int 21h
end
```

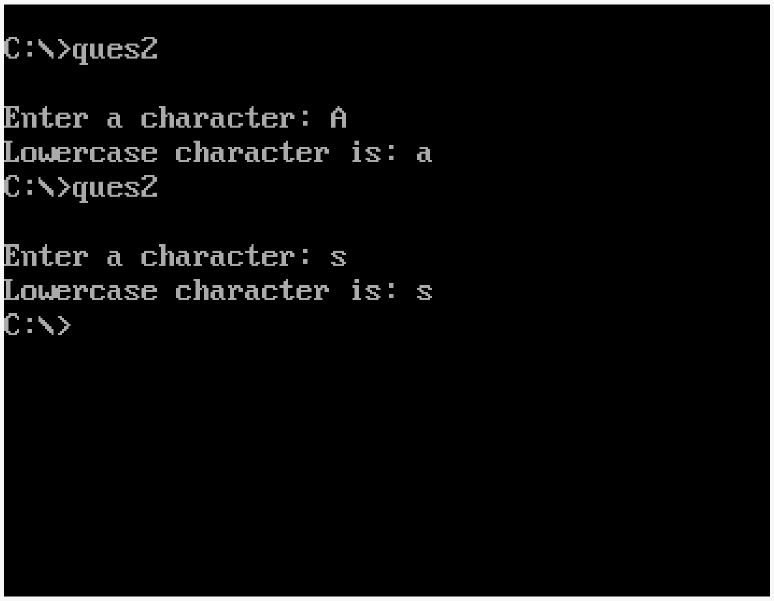


```
Name: Soumyadeep
Program title: Ques1
```

2. **Write and test a MASM program to convert a letter from uppercase to lowercase.**

First a character is taken as input using 01h and int 21h. Then it is checked if it is lowercase. If lowercase then it is converted to uppercase by subtracting 20h else no conversion is done.

```
; Write and test a MASM program to Convert a letter from uppercase to
lowercase.
.model small
.stack 100h

.data
msg1 db 10,13,"Enter a character: $"
msg2 db 10,13,"Lowercase character is: $"

.code

main proc

        mov ax,@data
        mov ds,ax

        ;display input prompt
        lea dx,msg1
        mov ah,09h
        int 21h

        ;accept a character
        mov ah,01h
        int 21h
        ;al has the character

        ;check if al is uppercase
        cmp al,'A'
        jl display

        cmp al,'Z'
        jg display

        add al,32

        display:
        ;display prompt
        lea dx,msg2
        mov ah,09h
        int 21h

        ;display the character
        mov dl,al
        mov ah,02h
        int 21h

        mov ah,4ch
        int 21h

main endp
end main
```



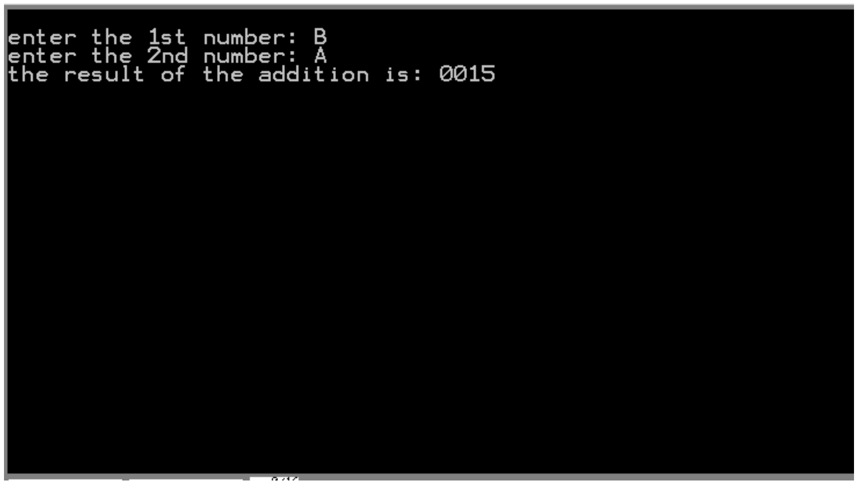
```
C:\>ques2

Enter a character: A
Lowercase character is: a
C:\>ques2

Enter a character: s
Lowercase character is: s
C:\>
```

3. Write and test a MASM program to add two Hexadecimal Numbers.

First two hexadecimal numbers are taken as input. Hexadecimal input is taken by accepting each character and converting to corresponding number. For every subsequent digit the register is left shifted by 4 bits and the next digit is stored. The addition of the registers is performed and printed out taking into account carry flag.

```
;write and test a masm program to add two hexadecimal numbers.

.model small
.stack 100h
.data
    prompt1 db 13,10,"enter the 1st number: $"
    prompt2 db 13,10,"enter the 2nd number: $"
    prompt3 db 13,10,"the result of the addition is: $"

.code
   main proc
          mov ax,@data                                  ;for moving data to
data segment
          mov ds,ax

          xor bx,bx                                     ;initially bx value is
equal to 0
          mov cl,4

          lea dx, prompt1                               ;show num1 prompt
          mov ah, 9
          int 21h

          mov ah,1                                      ;for taking input
          int 21h



      input1:
          cmp al,0dh                                    ;compare whether the
pressed key is 'enter' or not
          je line1                                      ;if it is equal to
'enter' then stop taking first value

          cmp al,39h                                    ;compare the input
whether it is letter or digit.39h is the ascii value of 9
          jg letter1

          and al,0fh                                    ;if it is digit then
convert it's ascii value to real value by masking
```

```
          jmp shift1

      letter1:                                      ;if it is letter then
subtract 37h from it to find it's real value
          sub al,37h

      shift1:
          shl bx, cl
          or  bl,al                                 ;making 'or' will add
the current value with previous value

          int 21h
          jmp input1
      line1:
          lea dx, prompt2                           ;show num2 prompt
          mov ah, 9
          int 21h

          xor dx,dx                                 ;set dx value zero

          mov ah,1
          int 21h



      input2:
          cmp al,0dh                                ;compare whether the
pressed key is 'enter' or not
          je line2                                  ;if it is equal to
'enter' then stop taking first value

          cmp al,39h                                ;compare the input
whether it is letter or digit.39h is the ascii value of 9
          jg letter2

          and al,0fh                                ;if it is digit then
convert it's ascii value to real value by masking
          jmp shift2

      letter2:                                      ;if it is letter then
subtract 37h from it to find it's real value
          sub al,37h

      shift2:
          shl dx, cl
          or  dl,al                                  ;making 'or' will add
the current value with previous value

          int 21h
          jmp input2
      line2:
          xor cx,cx
          mov cx,dx

          mov dh,16

      sum:
          add bx,cx                                ;add two number which are
stored in bx and cs register
          jc pc1                                   ;if the register is
overflowed then print an extra 1

          mov cl, 4

          lea dx, prompt3                          ;show answer prompt
          mov ah, 9
          int 21h

          output:                                  ;level for printing their
sum

          mov ch,bh
          shr ch, cl
          and ch,0fh

          cmp ch,10                                ;convert decimal to
binary
          add ch,'0'
          cmp ch,':'
          jl tag
          add ch,7
      tag:mov dl,ch
          mov ah,2
          int 21h

          mov ch,bh
          and ch,0fh
          cmp ch,10
          add ch,'0'
          cmp ch,':'
          jl tag1
          add ch,7
      tag1:mov dl,ch
          mov ah,2
          int 21h

          mov ch,bl
          shr ch, cl
          and ch,0fh
          cmp ch,10
          add ch,'0'
          cmp ch,':'
          jl tag2
          add ch,7
      tag2:mov dl,ch
          mov ah,2
          int 21h

          mov ch,bl
          and ch,0fh
          cmp ch,10
          add ch,'0'
          cmp ch,':'
          jl tag3
          add ch,7
      tag3:mov dl,ch
          mov ah,2
          int 21h

          jmp exit
      pc1:                                        ;level for printing
overflowed 1
          mov dl,'1'
          mov ah,2
          int 21h
          jmp output

      exit:
          mov ah, 4ch                             ;return control to dos
          int 21h

    main endp
   end main
```



```
enter the 1st number: B
enter the 2nd number: A
the result of the addition is: 0015
```

4. Write and test a MASM program to find the second max and second min from an array.

First the size of the array is taken as input then elements of the array are input. Next the maximum and minimum elements of the array are found out by comparing the current element with the last maximum and minimum element and accordingly update the max and min variables. After the max and min are found out another loop is run to find out the second max and second min by comparing with the current second max and second min and the already fount out max and min. Then the two elements are printed out.

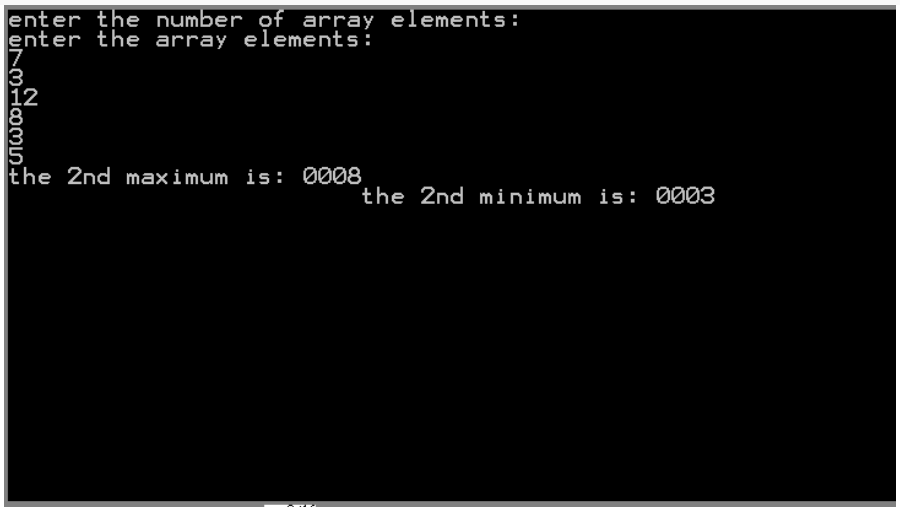
```
.model small
.stack 100h

.data
prompt_0  db  'enter the number of array elements :',0dh,0ah,'$'
prompt_1  db  'enter the array elements :',0dh,0ah,'$'
prompt_2  db  'the 2nd maximum is : $'
prompt_3  db  'the 2nd minimum is : $'

array   dw  50 dup(0)

s dw ?
max dw ?
min dw ?

.code
main proc

               mov ax, @data                 ; initialize ds
               mov ds, ax

               lea dx, prompt_0              ; load and display the string
prompt_0
```

```
               mov ah, 9
               int 21h

               mov ah,1                                    ;for taking input
               int 21h

               input1:
               cmp al,0dh                                  ;compare whether
the pressed key is 'enter' or not
               je line1                                    ;if it is equal to
'enter' then stop taking first value

               and al,0fh                                  ;convert it's
ascii value to real value by masking

               shl bx, 1
               shl bx, 1
               shl bx, 1
               shl bx, 1
               or  bl,al                                   ;making 'or' will
add the current value with previous value

               int 21h
               jmp input1

               line1:
               lea dx, prompt_1              ; load and display the string
prompt_1
               mov ah, 9
               int 21h

               lea si, array                 ; set si=offset address of array
               mov s,bx
               mov cx, bx                      ; set cx=bx


               @read_array:                    ; loop label

               mov ah,1                                    ;for taking input
               int 21h

               xor dx,dx

               input2:
               cmp al,0dh                                  ;compare whether
the pressed key is 'enter' or not
               je line2                                   ;if it is equal to
'enter' then stop taking first value

               and al,0fh                                 ;convert it's ascii
value to real value by masking

               shl dx,1
               shl dx,1
               shl dx,1
               shl dx,1
               or  dl,al                                   ;making 'or' will
add the current value with previous value

               int 21h
               jmp input2

               line2:
               mov [si], dx                 ; set [si]=ax
               add si, 2                    ; set si=si+2

               mov dl, 0ah                  ; line feed
               mov ah, 2                    ; set output function
               int 21h                      ; print a character

               loop @read_array               ; jump to label @read_array
while cx!=0
               ; array input done

               lea si,array
               mov ax,bx
               dec ax
               xor bx,bx
               xor cx,cx
               mov bx,word ptr[si]    ;store the maximum
               mov cx,word ptr[si]    ;store the 2nd
               add si, 2

               ; loop to find max and 2nd max
               arrayloop2:

               cmp word ptr[si],bx
               jl max2
               mov cx,bx
               mov bx,word ptr[si]

               max2:
               cmp word ptr[si],cx
               jl incre
               cmp word ptr[si],bx
               je incre
               mov cx,word ptr[si]

               incre:
               add si, 2
               dec ax

               jnz arrayloop2

               ; now bx has max cx has 2nd max
               mov max,bx
               ; displaying the prompt
               lea dx,prompt_2
               mov ah,09h
               int 21h

               ; display contents of cx
               mov bx,cx

               mov dh,bh
               shr dh, 1
               shr dh, 1
               shr dh, 1
               shr dh, 1
               and dh,0fh

               add dh,'0'
               mov dl,dh
               mov ah,2
               int 21h

               mov dh,bh
               and dh,0fh
               add dh,'0'
               mov dl,dh
               mov ah,2
               int 21h

               mov dh,bl
```

```
                shr dh, 1
                shr dh, 1
                shr dh, 1
                shr dh, 1
                and dh,0fh
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                mov dh,bl
                and dh,0fh
                cmp dh,10
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                mov dl, 0ah                    ; line feed
                mov ah, 2                      ; set output function
                int 21h                        ; print a character


        ;======================================================================
============
                lea si,array
                mov ax,s
                dec ax
                mov bx,max


                ; loop to find min and 2nd min
                arrayloop3:

                cmp word ptr[si],bx
                jg min2
                mov cx,bx
                mov bx,word ptr[si]

                min2:
                cmp word ptr[si],cx
                jg incre2
                cmp word ptr[si],bx
                je incre2
                mov cx,word ptr[si]
```

```
incre2:
add si, 2
dec ax

jnz arrayloop3

; now bx has min cx has 2nd min

; displaying the prompt
lea dx,prompt_3
mov ah,09h
int 21h

; display contents of cx
mov bx,cx

mov dh,bh
shr dh, 1
shr dh, 1
shr dh, 1
shr dh, 1
and dh,0fh

add dh,'0'
mov dl,dh
mov ah,2
int 21h

mov dh,bh
and dh,0fh
add dh,'0'
mov dl,dh
mov ah,2
int 21h

mov dh,bl
shr dh, 1
shr dh, 1
shr dh, 1
shr dh, 1
and dh,0fh
add dh,'0'
mov dl,dh
mov ah,2
```

```
                int 21h

                mov dh,bl
                and dh,0fh
                cmp dh,10
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h


                exit:
                mov ah, 4ch                                   ;return control to
dos
                int 21h

main endp
end main
```

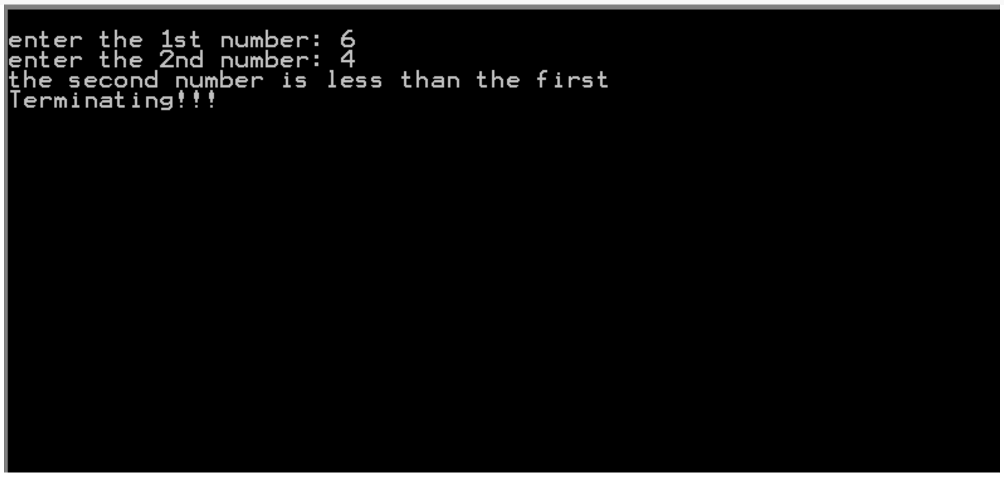


```
enter the number of array elements:
enter the array elements:
7
3
12
8
3
5
the 2nd maximum is: 0008
                        the 2nd minimum is: 0003
```

5. Write and test a MASM program to display a terminating message.

In this program a task has been performed and a terminating message has been displayed when the task is complete.

```
; Write and test a MASM program to display a terminating message.

.model small
.stack 100h
.data
    prompt1 db 13,10,"enter the 1st number: $"
    prompt2 db 13,10,"enter the 2nd number: $"
    promptyes db 13,10,"the second number is less than the first$"
    promptno db 13,10,"the second number is not less than the first$"
    promptter db 13,10,"Terminating!!!$"

.code
   main proc
          mov ax,@data                                ;for moving data to
data segment
          mov ds,ax

          xor bx,bx                                   ;initially bx value is
equal to 0
          mov cl,4

          lea dx, prompt1                             ;show num1 prompt
          mov ah, 9
          int 21h

          mov ah,1                                    ;for taking input
          int 21h


      input1:
          cmp al,0dh                                  ;compare whether the
pressed key is 'enter' or not
          je line1                                    ;if it is equal to
'enter' then stop taking first value

          cmp al,39h                                  ;compare the input
whether it is letter or digit.39h is the ascii value of 9
          jg letter1

          and al,0fh                                  ;if it is digit then
convert it's ascii value to real value by masking
          jmp shift1

      letter1:                                        ;if it is letter then
subtract 37h from it to find it's real value
          sub al,37h

      shift1:
          shl bx, cl
          or  bl,al                                   ;making 'or' will add
the current value with previous value

          int 21h
          jmp input1

      line1:
          lea dx, prompt2                                ;show num2 prompt
          mov ah, 9
          int 21h

          xor dx,dx                                      ;set dx value zero

          mov ah,1
          int 21h


      input2:
          cmp al,0dh                                     ;compare whether the 
pressed key is 'enter' or not
          je line2                                       ;if it is equal to 
'enter' then stop taking first value

          cmp al,39h                                     ;compare the input 
whether it is letter or digit.39h is the ascii value of 9
          jg letter2

          and al,0fh                                     ;if it is digit then 
convert it's ascii value to real value by masking
          jmp shift2

      letter2:                                           ;if it is letter then 
subtract 37h from it to find it's real value
          sub al,37h

      shift2:
          shl dx, cl
          or  dl,al                                      ;making 'or' will add 
the current value with previous value

          int 21h
          jmp input2
      line2:
          xor cx,cx
          mov cx,dx

          mov dh,16

      compare_nums:
          cmp bx,cx                                             ;add two number which are 
stored in bx and cs register
          jg pc1

          lea dx, promptno                                       ;show answer prompt
          mov ah, 9
          int 21h
          jmp exit                                              ;if the register is 
overflowed then print an extra 1

      pc1:
          lea dx, promptyes                                       ;show answer prompt
          mov ah, 9
          int 21h

      exit:
          lea dx, promptter                                      ;show terminating prompt
          mov ah, 9
          int 21h

          mov ah, 4ch                                           ;return control to dos
          int 21h

    main endp
   end main
```



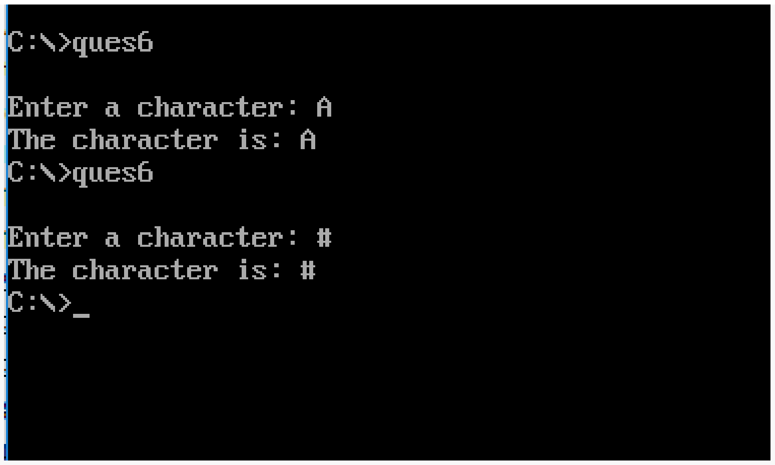
```
enter the 1st number: 6
enter the 2nd number: 4
the second number is less than the first
Terminating!!!
```

6. Write and test a MASM program to Take a character from keyboard and print it.

First a character is taken as input from keyboard using 01h, 21h and then it is printed out using 02h, 21h.

```
; Write and test a MASM program to Take a character from keyboard and print 
it.
```

```
.model small
.stack 100h

.data
msg1 db 10,13,"Enter a character: $"
msg2 db 10,13,"The character is: $"

.code

main proc

        mov ax,@data
        mov ds,ax

        ;display input prompt
        lea dx,msg1
        mov ah,09h
        int 21h

        ;accept a character
        mov ah,01h
        int 21h

        ;al has the character
        ;display prompt
        lea dx,msg2
        mov ah,09h
        int 21h

        ;display the character
        mov dl,al
        mov ah,02h
        int 21h

        mov ah,4ch
        int 21h

main endp
end main
```



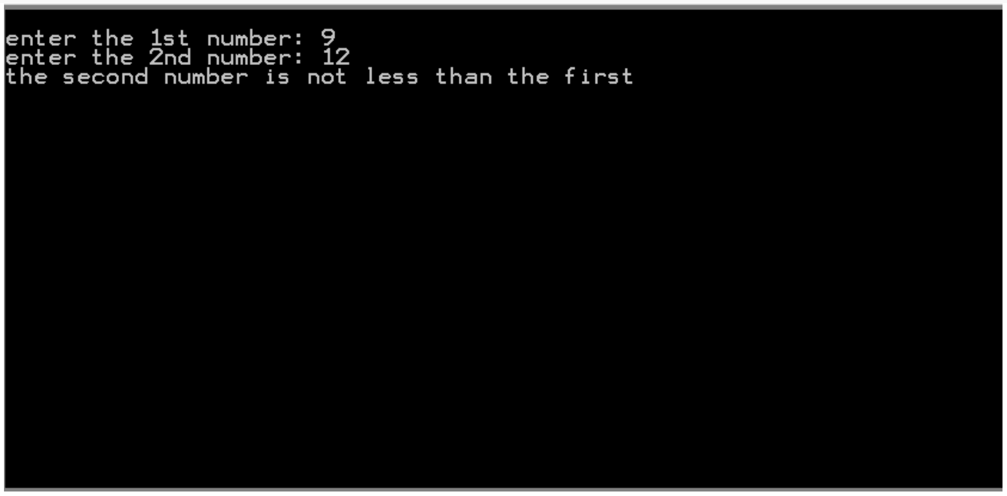
```
C:\>ques6

Enter a character: A
The character is: A
C:\>ques6

Enter a character: #
The character is: #
C:\>_
```

7. Write and test a MASM program to validate second numbers is less than the first.

Two numbers are taken as input using the previously described input procedure and the numbers are compared using the cmp instruction and if the second number is greater than the first an appropriate message is displayed using 09h, 21h.

```
; Write and test a MASM program to validate second numbers is less than the
first.

.model small
.stack 100h
.data
    prompt1 db 13,10,"enter the 1st number: $"
    prompt2 db 13,10,"enter the 2nd number: $"
    promptyes db 13,10,"the second number is less than the first$"
    promptno db 13,10,"the second number is not less than the first$"

.code
   main proc
          mov ax,@data                              ;for moving data to
data segment
          mov ds,ax

          xor bx,bx                                 ;initially bx value is
equal to 0
          mov cl,4

          lea dx, prompt1                           ;show num1 prompt
          mov ah, 9
          int 21h
          mov ah,1                                          ;for taking input
          int 21h


      input1:
          cmp al,0dh                                        ;compare whether the
pressed key is 'enter' or not
          je line1                                          ;if it is equal to
'enter' then stop taking first value

          cmp al,39h                                        ;compare the input
whether it is letter or digit.39h is the ascii value of 9
          jg letter1

          and al,0fh                                        ;if it is digit then
convert it's ascii value to real value by masking
          jmp shift1

      letter1:                                              ;if it is letter then
subtract 37h from it to find it's real value
          sub al,37h

      shift1:
          shl bx, cl
          or  bl,al                                         ;making 'or' will add
the current value with previous value

          int 21h
          jmp input1

      line1:
          lea dx, prompt2                                   ;show num2 prompt
          mov ah, 9
          int 21h

          xor dx,dx                                         ;set dx value zero

          mov ah,1
          int 21h


      input2:
          cmp al,0dh                                        ;compare whether the
pressed key is 'enter' or not
```

```
          je line2                                    ;if it is equal to 
'enter' then stop taking first value

          cmp al,39h                                  ;compare the input 
whether it is letter or digit.39h is the ascii value of 9
          jg letter2

          and al,0fh                                  ;if it is digit then 
convert it's ascii value to real value by masking
          jmp shift2

      letter2:                                        ;if it is letter then 
subtract 37h from it to find it's real value
          sub al,37h

      shift2:
          shl dx, cl
          or  dl,al                                   ;making 'or' will add 
the current value with previous value

          int 21h
          jmp input2
      line2:
          xor cx,cx
          mov cx,dx

          mov dh,16

      compare_nums:
          cmp bx,cx                                 ;add two number which are 
stored in bx and cs register
          jg pc1

          lea dx, promptno                           ;show answer prompt
          mov ah, 9
          int 21h
          jmp exit                                  ;if the register is 
overflowed then print an extra 1

      pc1:
          lea dx, promptyes                           ;show answer prompt
          mov ah, 9
          int 21h

      exit:
          mov ah, 4ch                                ;return control to dos
          int 21h

    main endp
   end main
```



```
enter the 1st number: 9
enter the 2nd number: 12
the second number is not less than the first
```

8. Write and test a MASM program to find maximum and minimum from an array.

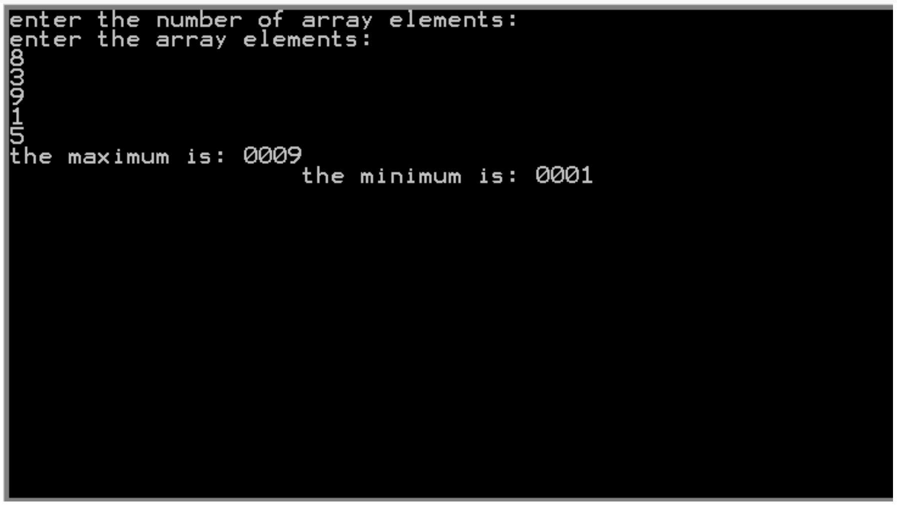
First the size of the array is taken as input then elements of the array are input. Next the maximum and minimum elements of the array are found out by comparing the current element with the last maximum and minimum element and accordingly update the max and min variables. Then max and min are printed out.

```
.model small
.stack 100h

.data
prompt_0   db  'enter the number of array elements :',0dh,0ah,'$'
prompt_1   db  'enter the array elements :',0dh,0ah,'$'
prompt_2   db  'the maximum is : $'
prompt_3   db  'the minimum is : $'

array    dw   50 dup(0)

s dw ?

.code
main proc

               mov ax, @data                ; initialize ds
               mov ds, ax
```

```
                lea dx, prompt_0              ; load and display the string
prompt_0
                mov ah, 9
                int 21h

                mov ah,1                                     ;for taking input
                int 21h

                input1:
                cmp al,0dh                                   ;compare whether
the pressed key is 'enter' or not
                je line1                                     ;if it is equal to
'enter' then stop taking first value

                and al,0fh                                   ;convert it's
ascii value to real value by masking

                shl bx, 1
                shl bx, 1
                shl bx, 1
                shl bx, 1
                or  bl,al                                    ;making 'or' will
add the current value with previous value

                int 21h
                jmp input1

                line1:
                lea dx, prompt_1              ; load and display the string
prompt_1
                mov ah, 9
                int 21h

                lea si, array                 ; set si=offset address of array

                mov cx, bx                      ; set cx=bx


                @read_array:                    ; loop label

                mov ah,1                                     ;for taking input
                int 21h

                xor dx,dx
                input2:
                cmp al,0dh                                          ;compare whether
the pressed key is 'enter' or not
                je line2                                           ;if it is equal to
'enter' then stop taking first value

                and al,0fh                                         ;convert it's ascii
value to real value by masking

                shl dx,1
                shl dx,1
                shl dx,1
                shl dx,1
                or  dl,al                                           ;making 'or' will
add the current value with previous value

                int 21h
                jmp input2

                line2:
                mov [si], dx                ; set [si]=ax
                add si, 2                   ; set si=si+2

                mov dl, 0ah                 ; line feed
                mov ah, 2                   ; set output function
                int 21h                     ; print a character

                loop @read_array              ; jump to label @read_array
while cx!=0
                ; array input done

                lea si,array
                mov ax,bx
                dec ax
                xor bx,bx
                xor cx,cx
                mov bx,word ptr[si]    ;store the maximum
                mov cx,word ptr[si]    ;store the minimum
                add si, 2

                ; loop to find max and min
                arrayloop2:

                cmp word ptr[si],bx
                jg maximum
                cmp word ptr[si],cx
                jl minimum

                jmp incre
                maximum:
                mov bx,word ptr[si]
                jmp incre

                minimum:
                mov cx,word ptr[si]

                incre:
                add si, 2
                dec ax

                jnz arrayloop2

                ; displaying the prompt
                lea dx,prompt_2
                mov ah,09h
                int 21h

                ; display contents of bx
                output:                                     ;level for printing
their sum

                mov dh,bh
                shr dh, 1
                shr dh, 1
                shr dh, 1
                shr dh, 1
                and dh,0fh

                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                mov dh,bh
                and dh,0fh
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h
```

```
mov dh,bl
shr dh, 1
shr dh, 1
shr dh, 1
shr dh, 1
and dh,0fh
add dh,'0'
mov dl,dh
mov ah,2
int 21h

mov dh,bl
and dh,0fh
cmp dh,10
add dh,'0'
mov dl,dh
mov ah,2
int 21h

mov dl, 0ah                    ; line feed
mov ah, 2                      ; set output function
int 21h                        ; print a character

; displaying the prompt
lea dx,prompt_3
mov ah,09h
int 21h

; display contents of cx
mov bx,cx

mov dh,bh
shr dh, 1
shr dh, 1
shr dh, 1
shr dh, 1
and dh,0fh

add dh,'0'
mov dl,dh
mov ah,2
int 21h

mov dh,bh
```

```
                and dh,0fh
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                mov dh,bl
                shr dh, 1
                shr dh, 1
                shr dh, 1
                shr dh, 1
                and dh,0fh
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                mov dh,bl
                and dh,0fh
                cmp dh,10
                add dh,'0'
                mov dl,dh
                mov ah,2
                int 21h

                exit:
                mov ah, 4ch                             ;return control to
dos
                int 21h

main endp
end main
```



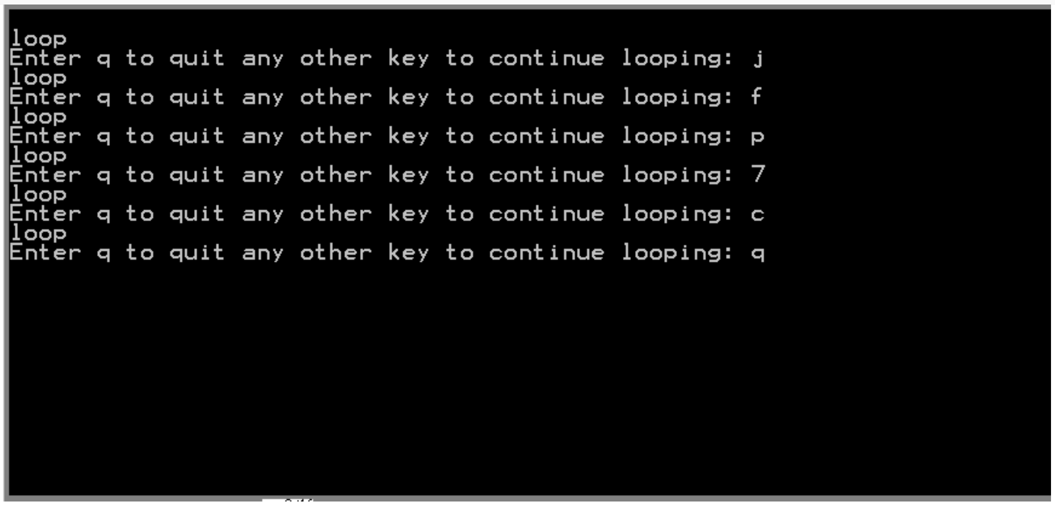
```
enter the number of array elements:
enter the array elements:
8
3
9
1
5
the maximum is: 0009
                    the minimum is: 0001
```

9. Write and test a MASM program to loop until the user decides to quit.

A loop is run infinitely and a message is displayed inside the loop and a character is taken as input. If the character is q then the program terminates else the looping continues.

```
;Write and test a MASM program to loop until the user decides to quit
.model small
.stack 100h

.data
msg db 10,13,"Enter q to quit any other key to continue looping: $"
looping db 10,13,"loop$"

.code

main proc

        mov ax,@data
        mov ds,ax

        label1:

                ;display loop message
                lea dx,looping
                mov ah,09h
                int 21h

                ;display input prompt
                lea dx,msg
                mov ah,09h
                int 21h

                ;accept a character
                mov ah,01h
                int 21h

                ; check if character is q
                cmp al,'q'
                jne label1

        ;exit
        mov ah,4Ch
        int 21h
main endp
```

```
end main
```

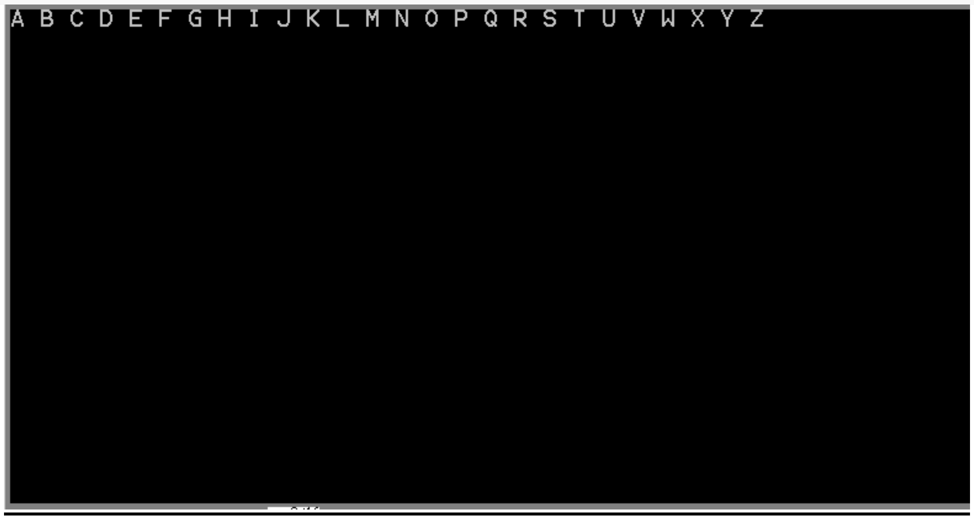


```
loop
Enter q to quit any other key to continue looping: j
loop
Enter q to quit any other key to continue looping: f
loop
Enter q to quit any other key to continue looping: p
loop
Enter q to quit any other key to continue looping: 7
loop
Enter q to quit any other key to continue looping: c
loop
Enter q to quit any other key to continue looping: q
```

10. Write and test a MASM program to print all the characters from A-Z.

A loop is run starting from ‘A’ and ending at ‘Z’. For every loop iteration the contents of bx are displayed using 02h, 21h.

```
; Write and test a MASM program to Print all the characters from A-Z.
.model small
.stack 100h

.data
space db ' '
.code

main proc

        mov ax,@data
        mov ds,ax

        mov bx,65
        mov cx,0

        label1:
                ;print the character
                mov ah,02h
                mov dl,bl
                int 21h

                ;print the character
                mov ah,02h
                mov dl,space
                int 21h

                ;increment
                inc bx
                inc cx
                cmp cx,26

        jne label1

        mov ah,4ch
        int 21h

main endp

end main
```



```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

<u>ASSIGNMENT 2</u>

<u>N.B.: For the following assignments a macro table mtab.asm has been created and used.</u>

mtab.asm

```
 ;macro to print new line
new_line macro
        mov ah,02h
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
endm
```

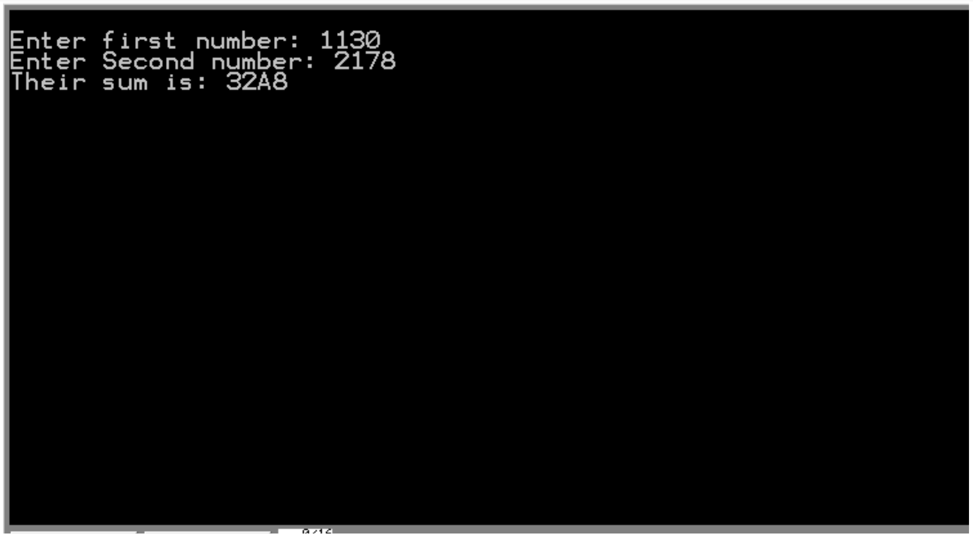
```
;macro to print space
space macro
        mov ah,02h
        mov dl,' '
        int 21h
endm

;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

        ; input : bx
        ; output : none


        ; cmp bx, 0                      ; compare bx with 0
        ; jge start                     ; jump to label start if bx>=0
        ; mov ah, 2                      ; set output function
        ; mov dl, "-"                    ; set dl='-'
        ; int 21h                        ; print the character

        ; neg bx                         ; take 2's complement of bx

        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

; macro to take binary input
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

;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

dec_input_with_neg macro
   local @read,@error,@minus,@plus,@inpit,@end,@exit
   jmp @read                         ; jump to label @read

   @error:                           ; jump label
   lea dx, illegal                   ; load and display the string illegal
   mov ah, 9
   int 21h

   @read:                            ; jump label
   xor bx, bx                        ; clear bx
   xor cx, cx                        ; clear cx

   mov ah, 1                         ; set input function
   int 21h                           ; read a character

   cmp al, "-"                       ; compare al with "-"
   je @minus                         ; jump to label @minus if al="-"

   cmp al, "+"                       ; compare al with "+"
   je @plus                          ; jump to label @plus if al="+"
   jmp @input                        ; jump to label @input

   @minus:                           ; jump label
   mov cx, 1                         ; set cx=1

   @plus:                            ; jump label
   int 21h                           ; read a character
   cmp al, 0dh                       ; compare al with cr
   je @end                           ; jump to label @end if al=cr

   @input:                           ; jump label
     cmp al, 30h                     ; compare al with 0
     jl @error                       ; jump to label @error if al<0

     cmp al, 39h                     ; compare al with 9
     jg @error                        ; jump to label @error if al>9

     and ax, 000fh                    ; convert ascii to decimal code

     push ax                          ; push ax onto the stack

     mov ax, 10                       ; set ax=10
     mul bx                           ; set ax=ax*bx
     mov bx, ax                       ; set bx=ax

     pop ax                           ; pop a value from stack into ax

     add bx, ax                       ; set bx=ax+bx

     mov ah, 1                        ; set input function
     int 21h                          ; read a character

     cmp al, 0dh                      ; compare al with cr
     jne @input                       ; jump to label if al!=cr

   @end:                              ; jump label

   or cx, cx                          ; check cx is 0 or not
   je @exit                           ; jump to label @exit if cx=0
   neg bx

   @exit:

endm

dec_output_with_neg macro

   cmp bx, 0                          ; compare bx with 0
   jge @start                         ; jump to label @start if bx>=0
   mov ah, 2                          ; set output function
   mov dl, "-"                        ; set dl='-'
   int 21h                            ; print the character
   neg bx                             ; take 2's complement of bx

   @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                   ; jump to label @display if cx!=0

endm

pushall macro
        push ax
        push bx
        push cx
        push dx
endm

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

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

First two numbers are taken as input using the hex_input macro defined in mtab.asm given above. Then addition is performed and the sum is output using hex_output macro.

```
include mtab.asm

.model small
.stack 100h

.data
        iprompt1 db 10,13,"Enter first number: $"
        iprompt2 db 10,13,"Enter second number: $"
        oprompt1 db 10,13,"Their sum is: $"
        oprompt2 db 10,13,"Their difference is: $"
        num1 dw ?
        num2 dw ?
.code

        main proc
                mov ax,@data
                mov ds,ax

                ;input prompt
                printm iprompt1
                hex_input
                mov num1,bx
                printm iprompt2
                hex_input
                mov num2,bx

                ;********************** SUM
**************************************
                printm oprompt1

                mov cx,num1
                add bx,cx

                jnc display

                carry_disp:
                        ;display carry
                        mov ah,02h
                        mov dl,'1'
                        int 21h

                display:
                        hex_output
                ;********************** SUM **********************************

                ;********************** DIFF *********************************
                new_line
                printm oprompt2

                mov bx,num1
                mov cx,num2
                sub bx,cx
```

```
                    hex_output
                    ;********************** DIFF *********************************

                    exitp

        main endp
end main
```



```
Enter first number: 1130
Enter Second number: 2178
Their sum is: 32A8
```

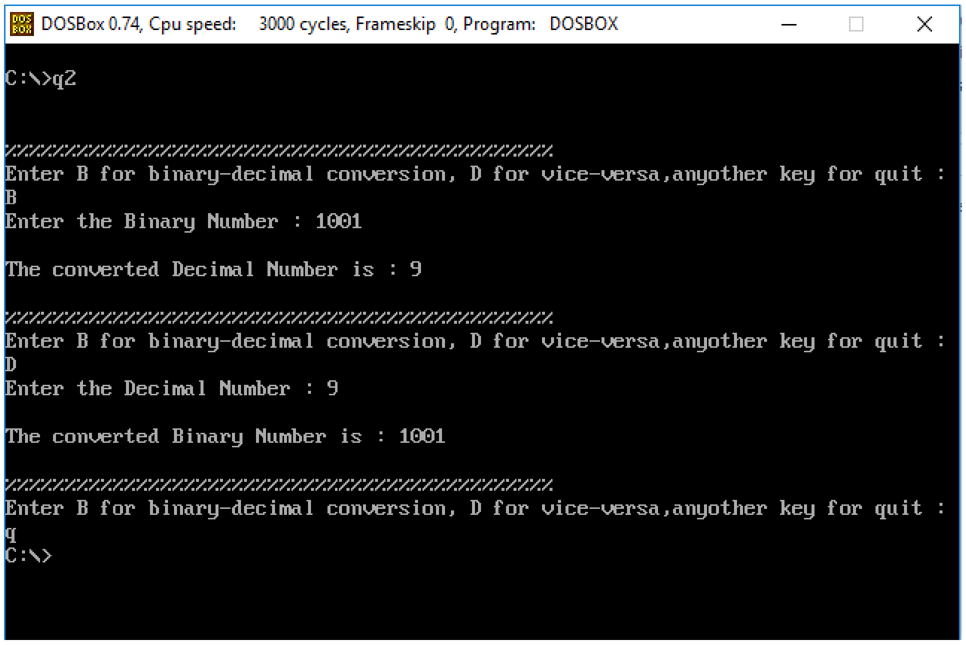
2. Write and test a MASM program to Convert Binary digit to Decimal.

A menu is displayed and a character is taken as input for choice if B is the character then binary input is taken using the bin_input procedure. This is done by taking each character and shifting by one bit and storing each digit. The input is continued until new line. Then the number is output using the dec_output procedure. The dec_output procedure gives decimal output by taking mod 10 of the register and dividing by 10. Every digit is pushed into the stack and printed out ono by ono by popping.

```
;12. Write and test a program to Convert a Binary digit to Decimal and vice
versa.

.model small
.stack 100h

.data
        prompt_0 DB
10,13,10,13,'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%$ '
        prompt_1 DB 10,13,'Enter B for binary-decimal conversion, D for vice-
versa, any other key for quit : $'
        prompt_2 DB 10,13,'Enter the Binary Number : $'
        prompt_3 DB 10,13,'Enter the Decimal Number : $'
        prompt_4 DB 10,13,'The converted Binary Number is : $'
        prompt_5 DB 10,13,'The converted Decimal Number is : $'
        display macro msg
                mov ah,9
                lea dx,msg
                int 21h
        endm

        ch_input macro
                mov ah,1
                int 21h
        endm

.code
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;Binary Input;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        bin_input proc
                xor bx,bx
                xor cx,cx

                @binput:
                mov ah,1
                int 21h

                cmp al,13
                je @bend

                sub ax,30h
                shl bx,1
                or bl,al
                jmp @binput

                @bend:
                ret
        bin_input endp
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;Binary Output;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        bin_output proc
                mov ax, bx                            ; set ax=bx
                xor cx, cx                            ; clear cx
                mov bx, 2                     ; set bx=2
```

```
          @brepeat:                                 ; 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 @brepeat                      ; jump to label @repeat if zf=0
          mov ah, 2                                 ; set output function

          @bdisplay:                                ; 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 @bdisplay                            ; jump to label @display
if cx!=0

          ret
     bin_output endp
     ;;;;;;;;;;;;;;;;;;;;Decimal Input;;;;;;;;;;;;;;;;
     dec_input proc

          @dread:                                   ; jump label
          xor bx, bx                                ; clear bx
          xor cx, cx                                ; clear cx
          mov ah,1
          int 21h                                   ; read a character
          cmp al, 0dh                               ; compare al with cr
          je @dend                                  ; jump to label @end if
al=cr

          @dinput:                                  ; jump label
          and ax, 000fh                     ; convert ascii to decimal code

          push ax                           ; push ax onto the stack

          mov ax, 10                        ; set ax=10
          mul bx                            ; set ax=ax*bx
          mov bx, ax                        ; set bx=ax

          pop ax                            ; pop a value from stack into ax

          add bx, ax                        ; set bx=ax+bx

          mov ah, 1                         ; set input function
          int 21h                           ; read a character
```

```
                cmp al, 0dh                     ; compare al with cr
                jne @dinput                     ; jump to label if al!=cr


                @dend:                           ; jump label
                ret                              ; return control to the calling
procedure
        dec_input endp
        ;;;;;;;;;;;;;decimal output;;;;;;;;;;;;;;;;;;;;;;
        dec_output proc
                mov ax, bx                              ; set ax=bx
                xor cx, cx                              ; clear cx
                mov bx, 10                              ; set bx=10

                @drepeat:                               ; 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 @drepeat                            ; jump to label @repeat
if zf=0

                mov ah, 2                               ; set output function

                @ddisplay:                              ; 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 @ddisplay                          ; jump to label @display
if cx!=0

                ret
        dec_output endp
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        main proc
                mov ax,@data
                mov ds,ax

                @start:
                display prompt_0
                display prompt_1
                ch_input

                cmp al,'D'
                je @dec2bin
                cmp al,'B'
                je @bin2dec
                jmp @main_exit

                @bin2dec:
                display prompt_2
                call bin_input
                display prompt_5
                call dec_output
                jmp @start

                @dec2bin:
                display prompt_3
                call dec_input
                display prompt_4
                call bin_output
                jmp @start

                @main_exit:
                mov ah,4ch
                int 21h
        main endp
end main
```

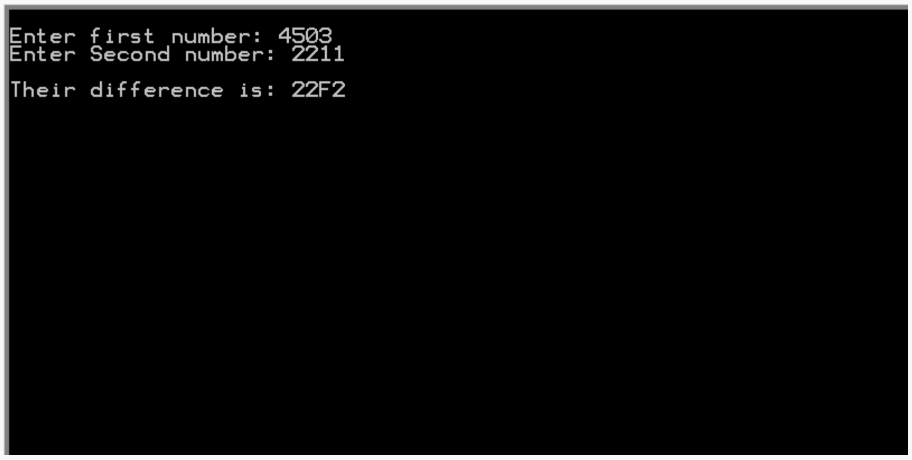

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
C:\>q2


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Enter B for binary-decimal conversion, D for vice-versa,anyother key for quit :
B
Enter the Binary Number : 1001

The converted Decimal Number is : 9

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Enter B for binary-decimal conversion, D for vice-versa,anyother key for quit :
D
Enter the Decimal Number : 9

The converted Binary Number is : 1001

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Enter B for binary-decimal conversion, D for vice-versa,anyother key for quit :
q
C:\>
```

3. Write and test a MASM program to perform subtraction of two 16 bit numbers.

First two numbers are taken as input using the hex_input macro defined in mtab.asm given above. Then addition is performed and the difference is output using hex_output macro.

```
include mtab.asm

.model small
.stack 100h

.data
        iprompt1 db "Enter first number: $"
        iprompt2 db "Enter first number: $"
        oprompt2 db "Their difference is: $"
        num1 dw ?
        num2 dw ?
.code

        main proc
                mov ax,@data
                mov ds,ax

                ;input prompt
                printm iprompt1
                hex_input
                mov num1,bx
                printm iprompt2
                hex_input
                mov num2,bx
                ;********************** DIFF *********************************
                new_line
                printm oprompt2

                mov bx,num1
                mov cx,num2
                sub bx,cx

                hex_output
                ;********************** DIFF *********************************

                exitp

        main endp
end main
```



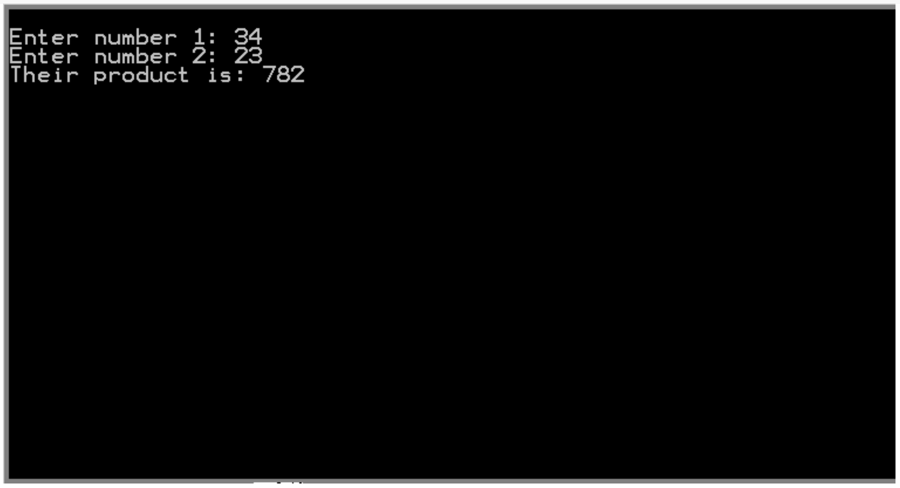
```
Enter first number: 4503
Enter Second number: 2211

Their difference is: 22F2
```

4. Write and test a MASM program to multiply two 8 bit numbers.

First two numbers are taken as input using the dec_input procedure defined in mtab.asm given above. Then addition is performed and the product is output using dec_output procedure.

```
;14.Write and test a program to multiply two 8 bit numbers.

include mtab.asm

.model small
.stack 100h

.data
```

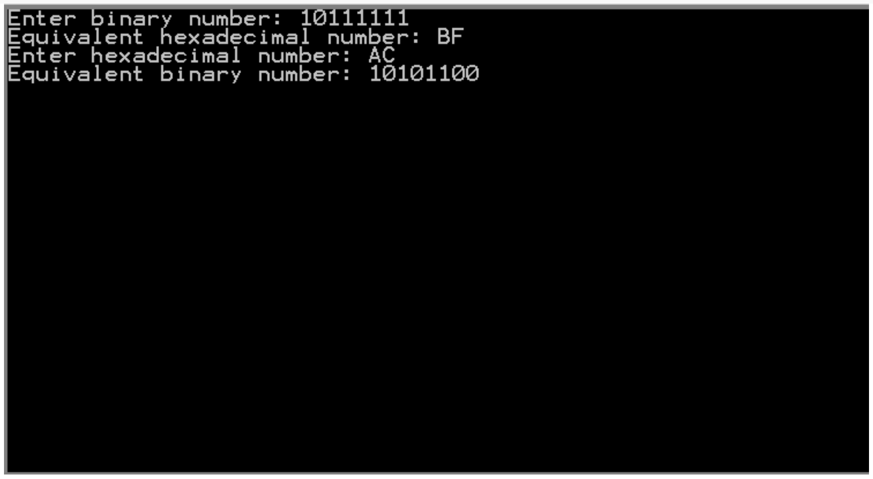
```
        iprompt1 db 10,13,"Enter number 1: $"
        iprompt2 db 10,13,"Enter number 2: $"
        oprompt1 db "Their product is: $"
        num1 db ?
        num2 db ?
.code

        main proc
                mov ax,@data
                mov ds,ax
                xor bh,bh
                ;input prompt
                printm iprompt1
                dec_input
                mov num1,bl
                xor bh,bh
                printm iprompt2
                dec_input
                mov num2,bl
                xor bh,bh
                xor ah,ah
                mov al,num1
                mul bx

                mov bx,ax
                new_line
                printm oprompt1
                dec_output

                exitp
        main endp
end main
```



```
Enter number 1: 34
Enter number 2: 23
Their product is: 782
```

5. Write and test a MASM program to Convert Binary digit to Hex digit.

```
;Write and test a program to Convert a Binary digit to HexaDecimal and vice
versa
include mtab.asm

.model small
.stack 100h

.data
        iprompt1 db "Enter binary number: $"
        iprompt2 db "Enter hexadecimal number: $"

        oprompt1 db "Equivalent hexadecimal number: $"
        oprompt2 db "Equivalent binary number: $"

.code

        main proc
                mov ax,@data
                mov ds,ax

                ;******************** BINARY TO HEXADECIMAL
******************
                ;input
                printm iprompt1

                bin_input       ; binary number in bx

                ;output
                new_line
                printm oprompt1
                hex_output
                ;***********************************************************

                ;******************** HEXADECIMAL TO BINARY
******************
                ;input
                new_line
                printm iprompt2

                hex_input       ; binary number in bx

                ;output
                new_line
                printm oprompt2
                bin_output
                ;************************************************************
                exitp
        main endp
end main
```

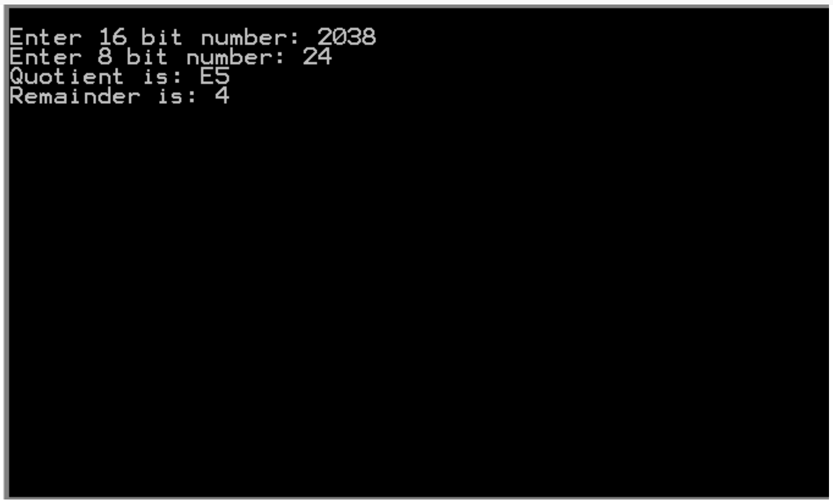


```
Enter binary number: 10111111
Equivalent hexadecimal number: BF
Enter hexadecimal number: AC
Equivalent binary number: 10101100
```

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

First two numbers are taken as input using the hex_input macro defined in mtab.asm given above. Then addition is performed and the quotient is output using hex_output macro.

```
;Write and test a program to divide a 16 bit number by a 8 bit number.
include mtab.asm
.model small
.stack 100h

.data
        iprompt1 db 10,13,"Enter 16 bit number: $"
        iprompt2 db 10,13,"Enter 8 bit number: $"
        oprompt1 db 10,13,"Quotient is: $"
        oprompt2 db 10,13,"Remainder is: $"
        num1 dw ?
.code

        main proc

                mov ax,@data
                mov ds,ax

                ;input
                printm iprompt1
                hex_input

                mov num1,bx

                printm iprompt2
                hex_input

                mov ax,num1
                xor dx,dx
                div bx

                ;output
                mov bx,ax
                mov num1,dx

                printm oprompt1
                pushall
                hex_output
                popall

                mov bx,num1

                printm oprompt2
                pushall
                hex_output
                popall

                exitp
        main endp
end main
```

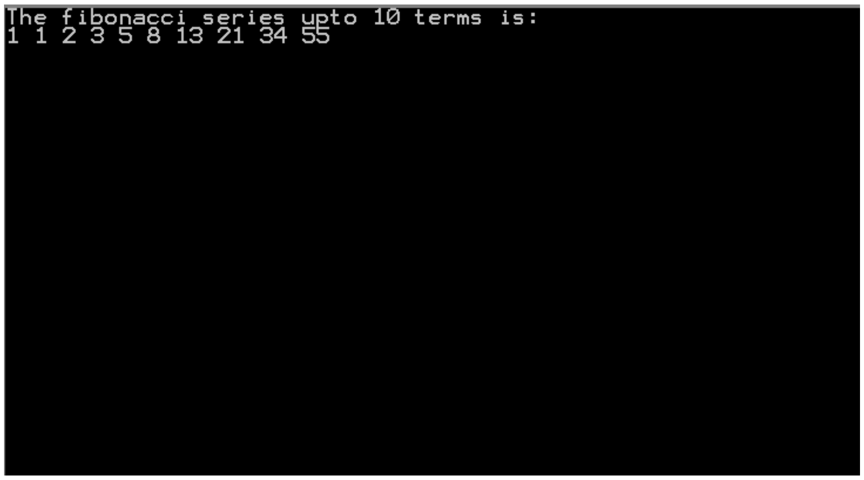


```
Enter 16 bit number: 2038
Enter 8 bit number: 24
Quotient is: E5
Remainder is: 4
```

7. Write and test a MASM program to Print Fibonacci series.

```
;17. Write and test a program to Print Fibonacci series up to 10 terms.

.model small
.stack 100h

.data
        prompt db "The fibonacci series upto 10 terms is: $"
        new_line db 10,13,"$"
        space db " $"


        f1 dw 1
        f2 dw 1
        f3 dw ?

        ;macro to display prompt and print string
        display macro msg
                mov ah,9
                lea dx,msg
                int 21h
        endm

        ;macro to push all registers into stack
        pushall macro
                push ax
                push bx
                push cx
                push dx
        endm

        ;macro to pop all registers from stack
        popall macro
                pop dx
                pop cx
                pop bx
                pop ax
        endm

.code

        ;**************************************************
        ; decimal output
        ;**************************************************
        decimal_output proc
   ; this procedure will display a decimal number
   ; input : bx
   ; output : none
   ; uses : main

           cmp bx, 0                        ; compare bx with 0
           jge @start                       ; jump to label @start if bx>=0
           mov ah, 2                        ; set output function
           mov dl, "-"                      ; set dl='-'
           int 21h                          ; print the character

           neg bx                           ; take 2's complement of bx

           @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                    ; jump to label @display if cx!=0

           ret                              ; return control to the calling
procedure
         decimal_output endp


        main proc

               mov ax,@data
               mov ds,ax
```

```
                mov bx,1
                mov dx,1
                display prompt
                display new_line

                pushall
                call decimal_output
                display space
                popall

                pushall
                call decimal_output
                display space
                popall

                mov bx,1
                mov dx,1
                mov cx,8

                @loop:
                        mov f1,bx
                        mov f2,dx
                        add bx,dx
                        mov f3,bx       ;f3=f1+f2

                        pushall
                        call decimal_output
                        display space
                        popall

                        mov bx,f2       ;f1=f2
                        mov dx,f3       ;f2=f3

                loop @loop

                mov ah,4ch
                int 21h

        main endp
end main
```



```
The fibonacci series upto 10 terms is:
1 1 2 3 5 8 13 21 34 55
```

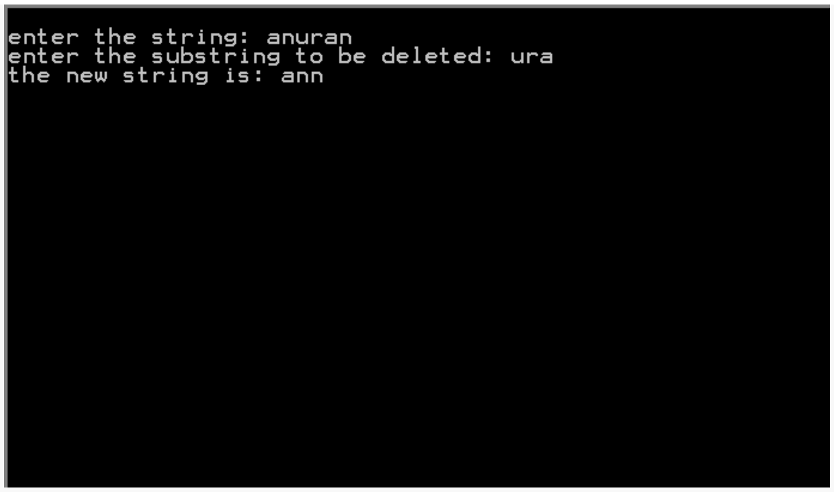
8. Write and test a MASM program for sub string deletion.

First the string is taken as input. Next the substring to be deleted is also taken as input. The original string is completely searched and the place where the substring where the given substring exists is found out and it is deleted. Finally the string is printed.

```
.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 new 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 copynig 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
```



```
enter the string: anuran
enter the substring to be deleted: ura
the new string is: ann
```

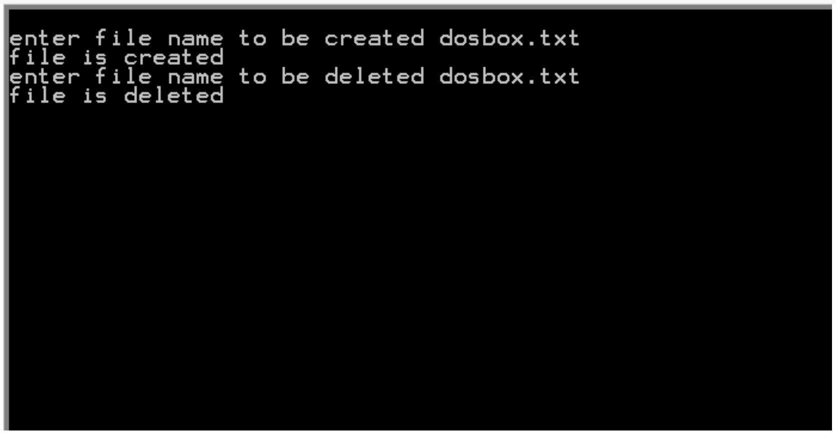
9. Write and test a MASM program to create and delete a file.

The file name for the file to be created is taken as a string input. The file is then created using 3ch, 21h interrupt. Similarly for the file to be deleted the file name is taken as input and deleted using 41h, 21h interrupt.

```
.model small
.stack 100h
.data
        msg1 db 10,13,'enter file name to be created $'
        msg2 db 10,13,'file is created$'
        msg3 db 10,13,'enter file name to be deleted $'
        msg4 db 10,13,'file is deleted$'
        msg5 db 10,13,'deletion error$'
        fnc db 50 dup(?)
        fnd db 50 dup(?)

.code
        pushall macro
                push ax
                push bx
                push cx
                push dx
        endm

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

```
print macro arg
        mov dx,offset arg
        mov ah,09h
        int 21h
endm

readstr macro arg
        local readlp,exit
        mov di,offset arg
        readlp:
                mov ah,01h
                int 21h
                cmp al,13
                je exit
                mov [di],al
                inc di
                jmp readlp
        exit:
endm

main proc
        mov ax,@data
        mov ds,ax

        print msg1
        pushall
        readstr fnc
        popall

        crte:
                mov cx,0
                mov dx,offset fnc
                mov ah,3ch
                int 21h
                print msg2

        print msg3
        pushall
        readstr fnd
        popall
        dlte:
                lea dx,fnd
                mov ah,41h
                int 21h
```

```
                        jc nfound
                        print msg4
                        jmp exit

                nfound:
                        print msg5

                exit:
                        mov ah,4ch
                        int 21h
        main endp
end main
```



```
enter file name to be created dosbox.txt
file is created
enter file name to be deleted dosbox.txt
file is deleted
```

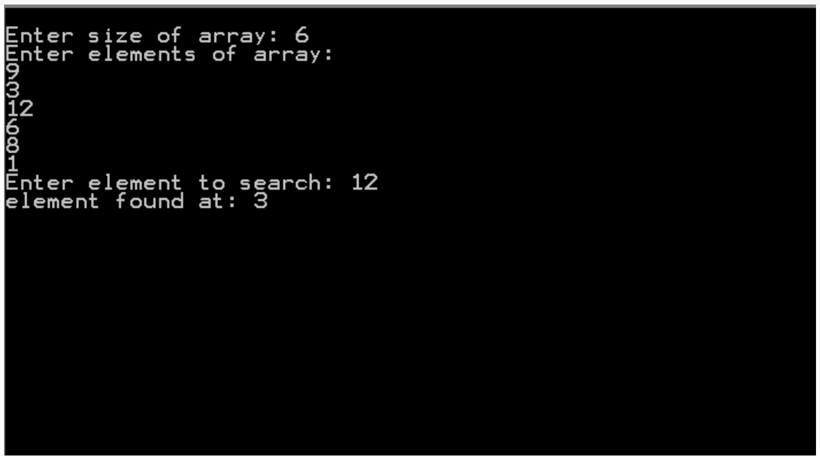
10. Write and test a MASM program to Implement Linear search.

First an array size is taken as input and then all elements of the array are input using the dec_input macro defined in mtab.asm. Then the whole array is scanned for the element and if found the index is displayed else not found is displayed.

```
;Write and test a program to Implement Linear search.
include mtab.asm

.model small
.stack 100h

.data
        prompts db 10,13,"Enter size of array: $"
        prompte db 10,13,"Enter elements of array: $"
        promptsr db 10,13,"Enter element to search: $"
        promptfound db 10,13,"element found at: $"
        promptnotfound db 10,13,"element not found $"
        arr dw 50 dup(?)
        s dw ?
.code
```

```
main proc

        mov ax,@data
        mov ds,ax

        ; display prompt for size
        printm prompts

        ;accept size
        dec_input
        ; bx has the size

        printm prompte

        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 promptsr
        dec_input

        ;bx has the element to be searched
        lea si,arr
        mov cx,s

        @linear_search:
                cmp bx,word ptr[si]
                je @found

                inc si
                inc si
        loop @linear_search
```

```
                ; not found case
                printm promptnotfound
                jmp @exit

                @found:
                        printm promptfound
                        mov bx,s
                        sub bx,cx
                        inc bx
                        dec_output
                        new_line

                @exit:
                        exitp
        main endp
end main
```



```
Enter size of array: 6
Enter elements of array:
9
3
12
6
8
1
Enter element to search: 12
element found at: 3
```

<u>ASSIGNMENT 3</u>

1. Write and test a MASM program to Implement Binary search. Show the steps. Each step will be succeeded by “*Enter*” key.

The size of the array is taken as input, next the elements of the array are also taken as input. The array is thereafter sorted by the sort procedure described in the next question. Then the element to be searched is taken as input. Then the element is searched using the binary search algorithm, comparing the middle element with the element to be searched and accordingly adjusting the limits of the portion of the array to be searched. If the element is found it is displayed else “not found” is displayed.

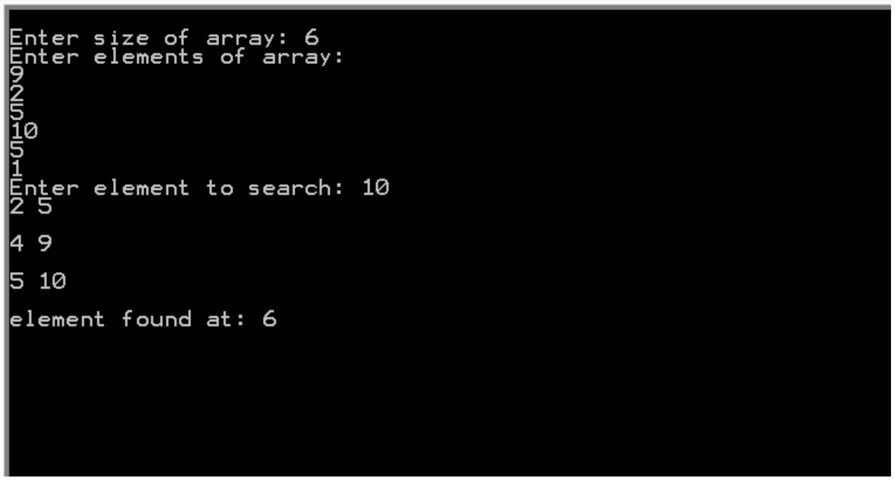
```
; MASM Program to implement binary search
```

```
include mtab.asm

.model small
.stack 100h

array_output macro arr
        local @array_print
        ;printing the array
        lea si,arr
        mov cx,s
        @array_print:
                mov bx,word ptr[si]
                mov temp,cx
                dec_output
                space
                inc si
                inc si
                mov cx,temp
        loop @array_print
endm

.data
        prompts db 10,13,"Enter size of array: $"
        prompte db 10,13,"Enter elements of array: $"
        promptsr db 10,13,"Enter element to search: $"
        promptfound db 10,13,"element found at: $"
        promptnotfound db 10,13,"element not found $"
        wrong_key db 10,13,"Invalid key entered: $"

        arr dw 50 dup(?)
        s dw ?
        strt dw ?
        stop dw ?
        min_idx dw ?
        temp dw ?
.code

        main proc

                mov ax,@data
                mov ds,ax

                ; display prompt for size
                printm prompts
```

```
            ;accept size
            dec_input
            ; bx has the size

            printm prompte

            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

            call sort
            ; enter element to search
            printm promptsr
            dec_input

            ;bx has the element to be searched
            lea si,arr
            mov cx,s
            dec cx
            mov strt,00h
            mov stop,cx

            ;************* BINARY SEARCH ********************
            @binary_search:
                    ;find out the middle index
                    lea si,arr
                    mov cx,stop
                    add cx,strt
                    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
                call deci_output
                pop cx
                pop bx
                space

                push bx
                push cx
                mov bx,word ptr[si]
                call deci_output
                pop cx
                pop bx
                new_line
                ;*************
                call ent

                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 strt,cx

                @compare:
                        mov cx,stop
                        cmp cx,strt

        jge @binary_search

        ;************************************************

        ; not found case
        printm promptnotfound
        jmp @exit
```

```
                @found:
                        printm promptfound
                        mov bx,cx
                        inc bx
                        dec_output
                        new_line

                @exit:
                        exitp
        main endp


        deci_output proc

                dec_output
                ret
        deci_output endp

        ent proc
                ;prompt for enter
                ;********** pressing enter will show next step esc will exit
************
                @error_enter:
                        mov ah,01h
                        int 21h

                        cmp al,1bh      ;check if esc is pressed
                        je @exit2
                        cmp al,0dh
                        je @compare2
                        printm wrong_key
                jmp @error_enter

        ;**********************************************************************
***
                @compare2:
                        ret

                @exit2:
                        exitp
        ent endp
        sort proc
                ;************** sorting ****************************
                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
                        ; here keyboard input inserted
                        @next_iter:
                        pop cx
                        pop si

                loop @outer_loop
                ret
                ;****************************************************
        sort endp

end main
```



```
Enter size of array: 6
Enter elements of array:
9
2
5
10
5
1
Enter element to search: 10
2 5

4 9

5 10

element found at: 6
```

2. **Write and test a MASM program to Implement Selection Sort. Show the steps. Each step will be succeeded by "*Enter*" key. The Program will terminate when the "*Esc*" key is pressed.**

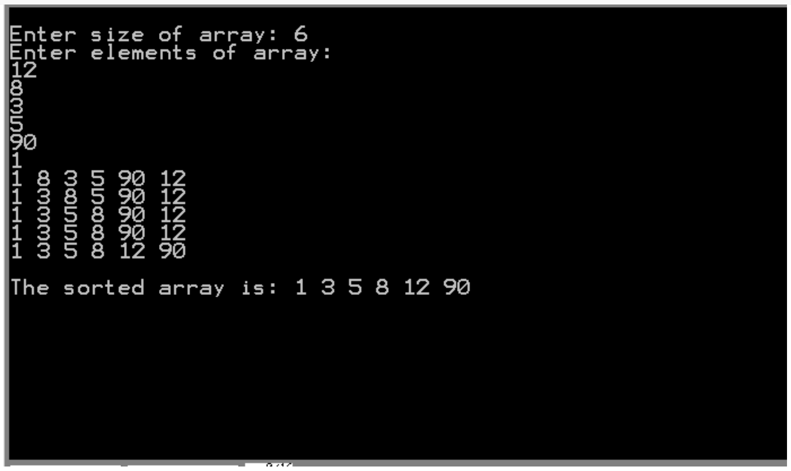
The size of the array is taken as input, next the elements of the array are also taken as input. The program sorts the array using selection sort algorithm. At each step the minimum element is found and swapped with the current element. At every step the array is printed out and if the input is enter then the next iteration is performed else if it is exit, the program terminates.

```
;Write and test a MASM program to Implement Selection Sort. Show the steps.
;Each step will be succeeded by "Enter" key. The Program will terminate when
the "Esc" key is pressed.
include mtab.asm

array_output macro arr
        local @array_print
        ;printing the array
        lea si,arr
        mov cx,s
        @array_print:
                mov bx,word ptr[si]
                mov temp,cx
                dec_output
                space
                inc si
                inc si
                mov cx,temp
        loop @array_print
endm
```

```
.model small
.stack 100h

.data
       prompts db 10,13,"Enter size of array: $"
       prompte db 10,13,"Enter elements of array: $"
       promptsr db 10,13,"The sorted array is: $"
       wrong_key db 10,13,"Invalid key entered: $"
       arr dw 50 dup(?)
       s dw ?
       temp dw ?
       min_idx dw ?
.code

       main proc

              mov ax,@data
              mov ds,ax

              ; display prompt for size
              printm prompts

              ;accept size
              dec_input
              ; bx has the size

              printm prompte

              mov s,bx
              lea si,arr
              mov cx,bx

              ;********** array input ***************************
              @array_input:
                     mov temp,cx
                     dec_input
                     mov word ptr[si],bx
                     mov cx,temp
                     inc si
                     inc si

              loop @array_input
              ;****************************************************
```

```
               ;************** sorting ***************************
               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

                       array_output arr
                       ; here keyboard input inserted
        ;******* pressing enter will show next step esc will exit ********
                       @error_enter:
                               mov ah,01h
                               int 21h

                               cmp al,1bh      ;check if esc is pressed
```

```
                                        je @exit
                                        cmp al,0dh
                                        je @next_iter
                                        printm wrong_key
                                jmp @error_enter

        ;*********************************************************************
                                @next_iter:
                                pop cx
                                pop si

                loop @outer_loop

                ;****************************************************
                ;******* array output ***************************
                printm promptsr
                array_output arr
                @exit:
                        exitp
        main endp
end main
```



```
Enter size of array: 6
Enter elements of array:
12
8
3
5
90
1
1 8 3 5 90 12
1 3 8 5 90 12
1 3 5 8 90 12
1 3 5 8 90 12
1 3 5 8 12 90

The sorted array is: 1 3 5 8 12 90
```

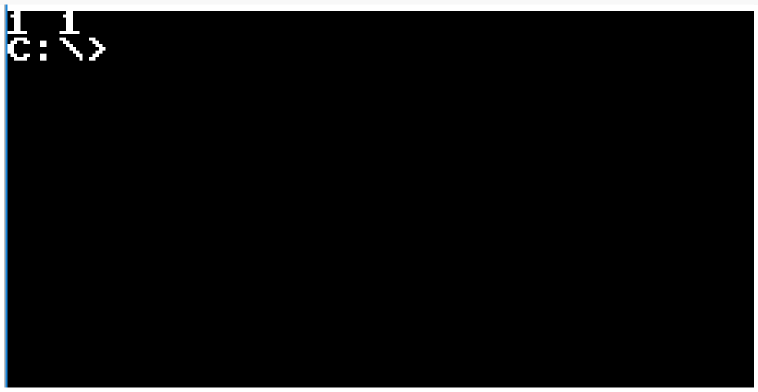
3. Write and test a MASM program to wait for left mouse clicks and display a text string at the exact clicked spot in the client area.

The mouse position is found out by the 03h, 33h interrupt. the cursor is then moved to the specific position and the string is printed on mouse click.

```
include mtab.asm

.model large
.stack 100
.data
```

```
        prompt1 db 'press left mouse button$'
        prompt2 db 'hello$'

.code
        main proc
                mov ax,@data
                mov ds,ax

                mov ah,00h
                mov al,13h
                int 10h

                xor cx,cx
                xor dx,dx
                mov ah,00h
                int 33h
                left_clk:
                        xor bx,bx
                        mov ax,3
                        int 33h
                        cmp bx,1
                        jne left_clk

                        ;mov dl,dh
                        ;mov dh,ch
                        ;mov bh,0
                        ;mov ah,2
                        ;int 10h
                        ;mov dx,offset prompt2
                        ;mov ah,09h
                        ;int 21h

                        pushall
                        dec_output cx
                        popall
                        mov ah,02h
                mov dl,20h
                int 21h
                        pushall
                        dec_output dx
                        popall

                mov ah,4ch
                int 21h
        main endp
```

```
end main
```



```
1  1
C:\>
```