System Programming Lab Report

Class – BCSE
Year – 3rd year 1st semester
Session – 2018-19

Group C –

Arnab Sur – Roll 001610501073
Suvam Dubey – Roll 001610501074
Tamoghna Mukherjee – Roll 001610501075
Sourav Dutta – Roll 001610501076
Soham Mukherjee – Roll 001610501077

ASSIGNMENT - 1

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

```
.model small
.stack 100h
.data
name1 db 0AH,0DH,'NAME: SOURAV DUTTA$'
title1 db 0AH,0DH,'PROGRAM TITLE: A1Q1.ASM$'
.code
print macro msg         ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print name1       ;invoking print macro to display name
      print title1      ;invoking print macro to display title

      mov ah, 4ch       ;terminate the program
      int 21h

main endp

end main
```

OUTPUT :

```
C:\>A1Q1

NAME: SOURAV DUTTA
PROGRAM TITLE: A1Q1.ASM
```

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

```
.model small
.stack 100h
.data
msg1 db 0DH,0AH,'Enter a character: $'
msg2 db 0DH,0AH,'Lower case character: $'
.code
print macro msg         ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
```

```
      mov ds,ax

      print msg1
      mov ah,01h  ; read character
      int 21h

      cmp al,'A'
      jl exit
      cmp al,'Z'
      jg exit

      add al,32   ; convert uppercase to lowercase by adding 32 to its ascii

      exit:
      print msg2
      mov dl,al   ; display character
      mov ah,02h
      int 21h
      mov ah, 4ch
      int 21h

main endp

end main
```

OUTPUT :

```
C:\>A1Q2

Enter a character: D
Lower case character: d
C:\>A1Q2

Enter a character: g
Lower case character: g
```

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

```
.model small
.stack 100h
.data
      msg1 db 0AH,0DH,'Enter first 16 bit hex number: $'
      msg2 db 0AH,0Dh,'Enter second 16 bit hex number: $'
      msg3 db 0AH,0DH,'Result after adding: $'
.code
print macro msg               ;macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm
main proc
      mov ax, @data           ; initialize data section
      mov ds, ax

      print msg1
      call readhex            ; Read first number
      mov cx, ax
      print msg2
      call readhex            ; Read second number
      print msg3
      add ax,cx               ; add two numbers
```

```
      call writehex           ; display the result

      mov ah, 4cH             ; terminate Program
     int 21H
main endp

readhex proc near
      ; this will input a 16 bit hexadecimal number
      ; output : AX

      push bx
      push cx
      push dx

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

      pop dx
      pop cx
      pop bx
      ret
readhex endp

writehex proc near
      ; this procedure is to display number in hexadecimal
      ; Input : AX
      push bx
      push cx
      push dx

      mov dx, 0000h
      jnc notcarry
      inc dx
      notcarry:
      mov si, ax
      mov bx, dx         ; Result in reg bx
      mov dh, 2
      l1:   mov ch, 04h     ; Count of digits to be displayed
      mov cl, 04h        ; Count to roll by 4 bits
      l2:   rol bx, cl          ; roll bl so that msb comes to lsb
      mov dl, bl         ; load dl wth data to be displayed
      and dl, 0fH        ; get only lsb
      cmp dl, 09         ; check if digit is 0-9 or letter A-F
      jbe l4
      add dl, 07         ; if letter add 37H else only add 30H
      l4:   add dl, 30H
```

```
      mov ah, 02          ; Function 2 under INT 21H (Display character)
      int 21H
      dec ch              ; Decrement Count
      jnz l2
      dec dh
      cmp dh, 0
      mov bx, si
      jnz l1

      pop dx
      pop cx
      pop bx
      ret
writehex endp

end main
```

OUTPUT :

```
C:\>A1Q3

Enter first 16 bit hex number: FFFF

Enter second 16 bit hex number: FFFF

Result after adding: 0001FFFE
```

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

```
.MODEL SMALL
.STACK 300H
.DATA
ARRAY1 DB 11,22,33,44,55
MSG1 DB 0AH,0DH,'Enter size of the array: $'
MSG2 DB 0AH,0DH,'Second Minimum value in array: $'
MSG3 DB 0AH,0DH,'Second Maximum value in array: $ '
ENDL DB 0AH,0DH,'$'

min dw 99
min2 dw 99
max dw 0
max2 dw 0
SE DB 33H
COUNT DB 00H

.CODE

PRINT MACRO MSG               ; macro to print a string
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      pop dx
      pop ax
ENDM

MAIN PROC
      MOV AX,@DATA
      MOV DS,AX

START:
```

```
      PRINT MSG1
      call readnum            ; read the size of array
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum      ; read an element
            mov ARRAY1[BX],AL ; and storing it in array
            inc BX
      loop rdnxt

      LEA SI, ARRAY1
      call findminmax         ; calling procedure to find min2 and max2

      print msg2
      mov ax, min2            ; second minimum is stored in min2
      call writenum           ; print the result

      print msg3
      mov ax, max2            ; second maximum is stored in max2
      call writenum           ; print the result

      mov ah, 4ch
      int 21h

MAIN ENDP

findminmax PROC
   ; this procedure will print the elements of a given array
   ; input : SI=offset address of the array
   ;       : BX=size of the array
   ; output : none

   PUSH AX                        ; push AX onto the STACK
   PUSH CX                        ; push CX onto the STACK
   PUSH DX                        ; push DX onto the STACK
   push SI
   MOV CX, BX                     ; set CX=BX

   @PRINT_ARRAY:                  ; loop label
     XOR AH, AH                   ; clear AH
     MOV AL, [SI]                 ; set AL=[SI]

      cmp min, ax
      jl notminupdate         ; if min >=  ax
        mov bx, min
        mov min2, bx          ; copy min to min2
      mov min, ax             ; copy ax to min
      jmp update1
      notminupdate:
      cmp min2, ax
      jl update1              ; if min2 >= ax
      cmp ax,min
      je update1              ; and if min2 != ax
      mov min2, ax            ; copy ax to min2
      update1:
      cmp max, ax
      jg notmaxupdate         ; if max <= ax
      mov bx, max
      mov max2, bx            ; copy max to max2
      mov max, ax             ; copy ax to max
      jmp update2
      notmaxupdate:
      cmp max2, ax
```

```
      jg update2              ; if max2 <= ax
      cmp ax, max
      je update2              ; and if max2 != ax
      mov max2, ax            ; copy ax to max2

      update2:

     MOV AH, 2                    ; set output function
     MOV DL, 20H                  ; set DL=20H
     INT 21H                      ; print a character

     INC SI                       ; set SI=SI+1
   LOOP @PRINT_ARRAY              ; jump to label @PRINT_ARRAY while CX!=0

   pop SI
   POP DX                         ; pop a value from STACK into DX
   POP CX                         ; pop a value from STACK into CX
   POP AX                         ; pop a value from STACK into AX

   RET                            ; return control to the calling procedure
findminmax ENDP


readnum proc near
      ; this procedure is to read a decimal number
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   push bx                        ; push BX onto the STACK
   push cx                        ; push CX onto the STACK
   push dx                        ; push DX onto the STACK

   XOR CX, CX                     ; clear CX
   MOV BX, 10                     ; set BX=10
```

```
   @OUTPUT:                        ; 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 @OUTPUT                     ; jump to label @OUTPUT 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

   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP BX                          ; pop a value from STACK into BX

   RET                             ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT :

```
C:\>A1Q4

Enter size of the array: 5
2
1
4
5
3
Second Minimum value in array: 2
Second Maximum value in array: 4
```

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

```
.model small
.stack 100h
.data
msg1 db 0AH,0DH,'ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): $'
msg2 db 0AH,0DH,'PROGRAM TERMINATED.$'
.code
print macro msg         ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax, @data
      mov ds, ax

      l1:
      print msg1
      mov ah, 01h       ; read a character
      int 21h
```

```
      cmp al,13          ; compare with ASCII value of enter key
      jne l1             ; continue until enter key is not pressed

      print msg2         ; print terminating message

      mov ah,4CH
      int 21h
main endp

end main
```

OUTPUT :

```
C:\>A1Q5

ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): J
ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): D
ENTER A CHARACTER (PRESS ENTER KEY TO EXIT):

PROGRAM TERMINATED.
```

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

```
.model small
.stack 100h
.data
msg1 db 0DH,0AH,'ENTER A CHARACTER: $'
msg2 db 0DH,0AH,'OUTPUT CHARACTER: $'
.code
print macro msg          ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax, @data
      mov ds, ax

      print msg1
      mov ah,01h         ;read character
      int 21h

      print msg2
      mov dl, al         ;display character
      mov ah, 02h
      int 21h

      mov ah,4ch
      int 21h
main endp

end main
```

OUTPUT :

```
C:\>A1Q6

ENTER A CHARACTER: V
OUTPUT CHARACTER: V
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter first decimal number: $'
msg2 db 0AH,0DH,'Enter second decimal number: $'
msg3 db 0AH,0DH,'Second number is less than first number$'
msg4 db 0AH,0DH,'Second number is not less than first number$'
.code
print macro msg               ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1
      call readdecimal        ; read first number, value is stored in ax
      mov cx, ax              ; copy first number to cx register

      print msg2
      call readdecimal

      cmp ax,cx               ; compare second with first number
      jl less
      print msg4              ; print message if second number is < first
      jmp exit
      less:                   ; print message if second number is >= first
      print msg3

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

readdecimal proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX

      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
```

```
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readdecimal endp

end main
```

OUTPUT :

```
C:\>A1Q7

Enter first decimal number: 34

Enter second decimal number: 21

Second number is less than first number
C:\>A1Q7

Enter first decimal number: 12

Enter second decimal number: 71

Second number is not less than first number
```

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

```
.MODEL SMALL
.STACK 300H
.DATA
ARRAY1 DB 11,22,33,44,55
MSG1 DB 0AH,0DH,'Enter size of the array: $'
MSG2 DB 0AH,0DH,'Minimum value in array: $'
MSG3 DB 0AH,0DH,'Maximum value in array: $ '
ENDL DB 0AH,0DH,'$'

min dw 99
max dw 0
SE DB 33H
COUNT DB 00H

.CODE

PRINT MACRO MSG         ; macro to print a string
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      pop dx
      pop ax
ENDM

MAIN PROC
```

```
      MOV AX,@DATA
      MOV DS,AX

START:

      PRINT MSG1
      call readnum            ; read the size of array
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum      ; read each array element
            mov ARRAY1[BX],AL ; storing it in array
            inc BX
      loop rdnxt

      LEA SI, ARRAY1
      call findminmax         ; calling procedure to find min and max

      print msg2
      mov ax, min             ; minimum value is stored in min
      call writenum           ; print the result

      print msg3
      mov ax, max             ; maximum value is stored in max
      call writenum           ; print the result

      mov ah, 4ch
      int 21h

MAIN ENDP

findminmax PROC
   ; this procedure will print the elements of a given array
   ; input : SI=offset address of the array
   ;       : BX=size of the array
   ; output : none

   PUSH AX                        ; push AX onto the STACK
   PUSH CX                        ; push CX onto the STACK
   PUSH DX                        ; push DX onto the STACK
   push SI
   MOV CX, BX                     ; set CX=BX

   @PRINT_ARRAY:                  ; loop label
     XOR AH, AH                   ; clear AH
     MOV AL, [SI]                 ; set AL=[SI]

      cmp min, ax
      jl notminupdate             ; if min >= ax
      mov min, ax                 ; copy ax to min
      notminupdate:

      cmp max, ax
      jg notmaxupdate             ; if max <= ax
      mov max, ax                 ; copy ax to max
      notmaxupdate:

     MOV AH, 2                    ; set output function
     MOV DL, 20H                  ; set DL=20H
     INT 21H                      ; print a character

     INC SI                       ; set SI=SI+1
   LOOP @PRINT_ARRAY              ; jump to label @PRINT_ARRAY while CX!=0
```

```
   pop SI
   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP AX                          ; pop a value from STACK into AX

   RET                             ; return control to the calling procedure
findminmax ENDP

readnum proc near

      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   push bx                         ; push BX onto the STACK
   push cx                         ; push CX onto the STACK
   push dx                         ; push DX onto the STACK

   XOR CX, CX                      ; clear CX
   MOV BX, 10                      ; set BX=10

   @OUTPUT:                        ; 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 @OUTPUT                     ; jump to label @OUTPUT 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
```

```
   POP DX                          ; pop a value from STACK into DX
   POP CX                          ; pop a value from STACK into CX
   POP BX                          ; pop a value from STACK into BX

   RET                             ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT :

```
C:\>A1Q8

Enter size of the array: 5
78
93
43
2
24
Minimum value in array: 2
Maximum value in array: 93
```

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

```
.model small
.stack 100h
.data
msg1 db 0AH,0DH,'ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): $'
msg2 db 0AH,0DH,'PROGRAM TERMINATED.$'
.code
print macro msg         ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax, @data
      mov ds, ax

      l1:
      print msg1
      mov ah, 01h ; read a character
      int 21h

      cmp al,13         ; compare with ASCII value of enter key
      jne l1            ; continue until enter key is not pressed

      print msg2        ; print terminating message

      mov ah,4CH
      int 21h
main endp

end main
```

OUTPUT :

```
C:\>A1Q5

ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): J
ENTER A CHARACTER (PRESS ENTER KEY TO EXIT): D
ENTER A CHARACTER (PRESS ENTER KEY TO EXIT):

PROGRAM TERMINATED.
```

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

```
.model small
.stack 100h
.data
.code
printchar macro char    ; macro to display a character
      push ax
      push dx
      mov dl,char
      mov ah,02h
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      mov cl,64         ;cl = 64 (ascii value of character just before 'A')
      l1:
      inc cl
      printchar cl      ;print alphabet
      printchar 20h     ;print space (ASCII - 20H)
      cmp cl,'Z'
      jne l1            ;loop until Z occurs

      mov ah, 4CH
      int 21h

main endp
end main
```

OUTPUT :

```
C:\>A1Q10
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
```

ASSIGNMENT - 2

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'ENTER 1ST NUMBER: $'
msg2 db 0AH,0DH,'ENTER 2ND NUMBER: $'
msg3 db 0AH,0DH,'THE RESULT AFTER ADDITION IS: $'
msg4 db 0AH,0DH,'THE RESULT AFTER SUBTRACTION IS: $'
space db ' $'
endl db 0AH,0DH,'$'

val1 dw ?
```

```
val2 dw ?

.code
print macro msg         ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1
      call readhex      ; reading first hex number
      mov val1, ax
      print msg2
      call readhex      ; reading second hex number
      mov val2, ax

      print msg3
        mov ax, val1
        mov bx, val2
        add ax,bx       ; adding first number with second number
      call writehex     ; printing the result

        print msg4
        mov ax, val1
        mov bx, val2
        sub ax,bx       ; subtract second number from first number
        call writehex   ; printing the result

      mov ah, 4ch
      int 21h

main endp

readhex proc near
      ; this will input a 16 bit hexadecimal number
      ; output : AX

      push bx
      push cx
      push dx

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

      pop dx
      pop cx
      pop bx
      ret
readhex endp

writehex proc near
      ; this procedure is to display number in hexadecimal
      ; Input : AX
      push bx
      push cx
      push dx

      mov dx, 0000h
      jnc notcarry
      inc dx
      notcarry:
      mov si, ax
            mov bx, dx         ; Result in reg bx
            mov dh, 2
l1:     mov ch, 04h     ; Count of digits to be displayed
            mov cl, 04h        ; Count to roll by 4 bits
l2:     rol bx, cl          ; roll bl so that msb comes to lsb
            mov dl, bl         ; load dl wth data to be displayed
            and dl, 0fH        ; get only lsb
            cmp dl, 09         ; check if digit is 0-9 or letter A-F
            jbe l4
            add dl, 07         ; if letter add 37H else only add 30H
l4:     add dl, 30H
            mov ah, 02         ; Function 2 under INT 21H (Display character)
            int 21H
            dec ch             ; Decrement Count
            jnz l2
            dec dh
            cmp dh, 0
            mov bx, si
            jnz l1

      pop dx
      pop cx
      pop bx
      ret
writehex endp
end main
```

OUTPUT :

```
C:\>A2Q1

ENTER 1ST NUMBER: ABCF

ENTER 2ND NUMBER: 1234

THE RESULT AFTER ADDITION IS: 0000BE03
THE RESULT AFTER SUBTRACTION IS: 0000999B
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter binary number: $'
msg2 db 0AH,0DH,'Decimal: $'
msg3 db 0AH,0DH,'Enter Decimal number: $'
msg4 db 0AH,0DH,'Binary: $'

space db ' $'
endl db 0AH,0DH,'$'
binno db 17
        db ?
        db 17 dup(0)

str1 db 20 dup('$')
str2 db 20 dup('$')

.code

val1 dw ?
val2 dw ?

.code
print macro msg         ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm
read macro memloc       ; macro to read a binary number
      push ax
      push cx
      push dx
      mov ah, 0ah
      lea dx, memloc
      int 21h
      lea si, memloc + 1 ;NUMBER OF CHARACTERS ENTERED.
      mov cl, [si] ;MOVE LENGTH TO CL.
      mov ch, 0      ;CLEAR CH TO USE CX.
      inc cx ;TO REACH CHR(13).
      add si, cx ;NOW SI POINTS TO CHR(13).
      mov al, '$'
      mov [si], al ;REPLACE CHR(13) BY '$'.

      pop dx
      pop cx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      start:

      print msg1

      read binno        ; bin no is stOred in binno

      print  msg2
      mov ax,0000h
      mov bx,0000h
```

```
    lea si, binno + 1
    mov cl, [si]
    mov ch, 00h
    inc  si
    ;add si, cx
    mov ax,cx
    ;call writenum
    ;print endl
    mov ax,00h

    loop1:
          mov bl, [si]
          sub bl, '0'
          mov bh, 00h
          mov dx,02h
          mul dx
          add ax, bx
          ;call writenum
          ;print endl
          inc si
    loop loop1


    call writenum           ; printing the decimal value of given binary number

    print endl
    print msg3
    call readnum            ; reading a decimal number

    lea si, str1

    mov bh, 00
    mov bl,2

    l1:
    div bl
    add ah,'0'
    mov byte ptr[si],ah
    mov ah, 00
    inc si
    inc bh
    cmp al,00
    jne l1

    mov cl,bh
    lea si, str1
    lea di, str2
    mov ch, 00
    add si, cx
    dec si

    l2:
    mov ah,byte ptr[si]
    mov byte ptr[di],ah
    dec si
    inc di
    loop l2

    print msg4
    print str2        printing the binary value of given decimal number

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

```
main endp

readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none

      ; output : AX


      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none

      push ax
      push bx
      push cx
      push dx

      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
      jne @output

      mov ah, 02h                     ; set output function

      @display:
            pop dx                      ; pop a value(remainder) from STACK to DX
            or dl, 30h                  ; convert decimal to ascii code
            int 21h
      loop @display
```

```
      pop dx
      pop cx
      pop bx
      pop ax

      ret
writenum endp

end main
```

OUTPUT :

```
C:\>A2Q2

Enter binary number: 10101010
Decimal: 170

Enter Decimal number: 221

Binary: 11011101
```

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

```
.model small
.stack 300h
.data
char1 db '($'
char2 db ')$'
space db ' $'
val1 dw ?

.code
print macro msg         ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      mov bx, 100       ; storing the decimal value 100
      mov ax, 100
      loop1:
            print char1       ; print opening bracket
            call writenum     ; print first number of pair
            print space
            mov val1, ax

            mov ax, bx
            mov cx, val1
            sub ax, cx  ; subtract first number with 100 to get second number of pair
            call writenum     ; print second number of pair
            print char2       ; print closing bracket
            print space       ; print space

            mov ax, val1
```

```
            sub ax,2          ; subtract first value by 2
            jnz loop1         ; loop until first value becomes 0

      print char1
      call writenum
      print space
      mov ax, 64h
      call writenum
      print char2

      mov ah, 4ch
      int 21h

main endp

readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none
      push ax
      push bx
      push cx
      push dx

      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
```

```
      jne @output

      mov ah, 02h                       ; set output function

      @display:
            pop dx                      ; pop a value(remainder) from STACK to DX
            or dl, 30h                  ; convert decimal to ascii code
            int 21h
      loop @display

      pop dx
      pop cx
      pop bx
      pop ax
      ret
writenum endp

end main
```

OUTPUT :

```
C:\>A2Q2
(100 0) (98 2) (96 4) (94 6) (92 8) (90 10) (88 12) (86 14) (84 16) (82 18) (80
20) (78 22) (76 24) (74 26) (72 28) (70 30) (68 32) (66 34) (64 36) (62 38) (60
40) (58 42) (56 44) (54 46) (52 48) (50 50) (48 52) (46 54) (44 56) (42 58) (40
60) (38 62) (36 64) (34 66) (32 68) (30 70) (28 72) (26 74) (24 76) (22 78) (20
80) (18 82) (16 84) (14 86) (12 88) (10 90) (8 92) (6 94) (4 96) (2 98) (0 100)
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'ENTER 1ST HEX NUMBER: $'
msg2 db 0AH,0DH,'ENTER 2ND HEX NUMBER: $'
msg3 db 0AH,0DH,'THE RESULT AFTER MULTIPLYING IS: $'
val1 dw ?
val2 dw ?

.code
print macro msg                 ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1
      call readhex              ; read first hex number
      mov val1, ax
      print msg2
      call readhex              ; read second hex number

      print msg3
      mul val1                  ; multiply first number with second number
      call writehex             ; printing the result

      mov ah, 4ch
```

```
      int 21h

main endp

readhex proc near
      ; this will input a 16 bit hexadecimal number
      ; output : AX

      push bx
      push cx
      push dx

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

      pop dx
      pop cx
      pop bx
      ret
readhex endp

writehex proc near
      ; this procedure is to display number in hexadecimal
      ; Input : AX
      push bx
      push cx
      push dx

      mov si, ax
            mov bx, dx         ; Result in reg bx
            mov dh, 2
l1:     mov ch, 04h     ; Count of digits to be displayed
            mov cl, 04h        ; Count to roll by 4 bits
l2:     rol bx, cl          ; roll bl so that msb comes to lsb
            mov dl, bl         ; load dl wth data to be displayed
            and dl, 0fH        ; get only lsb
            cmp dl, 09         ; check if digit is 0-9 or letter A-F
            jbe l4
            add dl, 07         ; if letter add 37H else only add 30H
l4:     add dl, 30H
            mov ah, 02         ; Function 2 under INT 21H (Display character)
            int 21H
            dec ch             ; Decrement Count
            jnz l2
            dec dh
```

```
            cmp dh, 0
            mov bx, si
            jnz l1

      pop dx
      pop cx
      pop bx
      ret
writehex endp

end main
```

OUTPUT :

```
C:\>A2Q4

ENTER 1ST NUMBER: FF

ENTER 2ND NUMBER: FF

THE RESULT AFTER MULTIPLYING IS: 0000FE01
```

15. Write and test a program to Convert Binary digit to Hex digit and vice versa.

```
.MODEL SMALL
.STACK 1000h
.DATA
  HEX_Map   DB  '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
  HEX_Out   DB  "00", 13, 10, '$'   ; string with line feed and '$'-terminator

msg1 db 0AH,0DH,'Enter binary number: $'
msg2 db 0AH,0DH,'Hexadecimal: $'
msg3 db 0AH,0DH,'Enter hexadecimal number: $'
msg4 db 0AH,0DH,'Binary: $'
space db ' $'
endl db 0AH,0DH,'$'
binno db 17
        db ?
        db 17 dup(0)
str1 db 20 dup('$')
str2 db 20 dup('$')

.code
val1 dw ?
val2 dw ?

.CODE
print macro msg
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm
read macro memloc
      push ax
      push cx
      push dx
      mov ah, 0ah
      lea dx, memloc
      int 21h
      lea si, memloc + 1 ;NUMBER OF CHARACTERS ENTERED.
```

```
      mov cl, [si] ;MOVE LENGTH TO CL.
      mov ch, 0      ;CLEAR CH TO USE CX.
      inc cx ;TO REACH CHR(13).
      add si, cx ;NOW SI POINTS TO CHR(13).
      mov al, '$'
      mov [si], al ;REPLACE CHR(13) BY '$'.

      pop dx
      pop cx
      pop ax
endm

main PROC
      mov ax, @DATA                   ; Initialize DS
      mov ds, ax

      print msg1
      read binno        ; bin no stored in binno

      print  msg2
      mov ax,0000h
      mov bx,0000h
      lea si, binno + 1
      mov cl, [si]
      mov ch, 00h
      inc  si
      ;add si, cx
      mov ax,cx
      ;call writenum
      ;print endl
      mov ax,00h

      loop1:
            mov bl, [si]
            sub bl, '0'
            mov bh, 00h
            mov dx,02h
            mul dx
            add ax, bx
            ;call writenum
            ;print endl
            inc si
      loop loop1



      ; Example No. 1 with output
      mov di, OFFSET HEX_Out          ; First argument: pointer
      ;mov ax, 10101100b               ; Second argument: Integer
      call IntegerToHexFromMap        ; Call with arguments
      mov ah, 09h                     ; Int 21h / 09h: Write string to STDOUT
      mov dx, OFFSET HEX_Out          ; Pointer to '$'-terminated string
      int 21h                         ; Call MS-DOS

      mov ax, 4C00h               ; Int 21h / 4Ch: Terminate program (Exit code = 00h)
      int 21h                         ; Call MS-DOS
main ENDP

IntegerToHexFromMap PROC
      mov si, OFFSET Hex_Map          ; Pointer to hex-character table

      mov bx, ax                      ; BX = argument AX
      and bx, 00FFh                   ; Clear BH (just to be on the safe side)
      shr bx, 1                       ; Isolate high nibble (i.e. 4 bits)
      SHR BX,1
```

```
      SHR BX,1
      SHR BX,1
      mov dl, [si+bx]                 ; Read hex-character from the table
      mov [di+0], dl          ; Store character at the first place in the output string

      mov bx, ax                      ; BX = argument AX (just to be on the safe side)
      and bx, 00FFh                   ; Clear BH (just to be on the safe side)
      and bl, 0Fh                     ; Isolate low nibble (i.e. 4 bits)
      mov dl, [si+bx]                 ; Read hex-character from the table
      mov [di+1], dl  ; Store character at the second place in the output string
      ret
IntegerToHexFromMap ENDP

readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum
      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none
      push ax
      push bx
      push cx
      push dx
      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
      jne @output
      mov ah, 02h                     ; set output function
      @display:
            pop dx                      ; pop a value(remainder) from STACK to DX
```

```
            or dl, 30h                  ; convert decimal to ascii code
            int 21h
      loop @display
      pop dx
      pop cx
      pop bx
      pop ax
      ret
writenum endp

END main                            ; End of assembly with entry-procedure
```

OUTPUT :

```
C:\>A2Q5

Enter binary number: 11010110
Hexadecimal: D6
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'ENTER 1ST NUMBER: $'
msg2 db 0AH,0DH,'ENTER 2ND NUMBER: $'
msg3 db 0AH,0DH,'THE RESULT AFTER DIVIDING IS: $'
val1 dw ?
val2 dw ?

.code
print macro msg
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1
      call readnum            ;read first number
      mov val1, ax

      print msg2
      call readnum            ; read second number
      mov val2, ax

      print msg3
      mov ax, val1
      mov bx, val2
      div bx                  ; dividing first number by second number

      call writenum           ; printing the result

      mov ah, 4ch
      int 21h

main endp
```

```
readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none

      push ax
      push bx
      push cx
      push dx

      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
      jne @output

      mov ah, 02h                     ; set output function

      @display:
            pop dx                      ; pop a value(remainder) from STACK to DX
            or dl, 30h                  ; convert decimal to ascii code
            int 21h
      loop @display

      pop dx
      pop cx
      pop bx
```

```
      pop ax

      ret
writenum endp

end main
```

OUTPUT :

```
C:\>A2Q6

ENTER 1ST NUMBER: 12

ENTER 2ND NUMBER: 4

THE RESULT AFTER DIVIDING IS: 3
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter number of steps: $'
msg2 db 0AH,0DH,'Fibonacci sequence: $'
space db ' $'
endl db 0AH,0DH,'$'

val db ?

.code
print macro msg               ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1

      call readnum      ; read the number of terms to be printed

      mov val, al
      mov bx, 00h
      mov dx, 01h
      mov cl, val
      mov ch, 00h
      mov ax, 00h
      print msg2
      print endl
      loop1:
            mov ax, bx
            call writenum     ; printing each term
            print space
            add ax, dx
            mov dx, bx
            mov bx, ax
      loop loop1              ;  loop n times ( n is stored in cl )
```

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

main endp

readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none

      push ax
      push bx
      push cx
      push dx

      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
      jne @output

      mov ah, 02h                     ; set output function

      @display:
            pop dx                      ; pop a value(remainder) from STACK to DX
            or dl, 30h                  ; convert decimal to ascii code
```

```
            int 21h
      loop @display

      pop dx
      pop cx
      pop bx
      pop ax

      ret
writenum endp

end main
```

OUTPUT :

```
C:\>A2Q7

Enter number of steps: 10

Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
```

18. Write and test a program for sub-string deletion from a given string.

```
.MODEL SMALL
.STACK 100H
.DATA
      MESS1 DB 10,13, "Enter your string : $"
      MESS2 DB 10,13, "Enter your substring that you want to be delete : $"
      MESS3 DB 10,13, "The string after deletion is : $"
      MESS4 DB 10,13, "Substring is not contained in string.$"
      STRING DB 50 DUP(?)
      SUBSTRING DB 50 DUP(?)
      NUM DW ?
      LEN1 DB ?
      LEN2 DB ?
      STARTINDEX DW ?
      ENDINDEX DW ?
.CODE
      MOV AX, @DATA
      MOV DS, AX

      LEA DX, MESS1
      MOV AH, 09H
      INT 21H

      MOV SI, 0
      MOV CX, 0
      MOV AH, 01H
      IN1: INT 21H
            CMP AL, 0DH
            JE OUT1
            MOV STRING[SI], AL
            INC SI
            INC CX
      JMP IN1

      OUT1:
      MOV LEN1, CL
      LEA DX, MESS2
        MOV AH, 09H
        INT 21H

        MOV SI, 0
      MOV CX, 0
```

```
        MOV AH, 01H
        IN2: INT 21H
                CMP AL, 0DH
                JE OUT2
                MOV SUBSTRING[SI], AL
                INC SI
            INC CX
        JMP IN2

      OUT2:

      MOV LEN2, CL

      MOV DH, 0
      MOV DL, LEN1
      SUB DL, LEN2
      ADD DL, 1

      MOV CH, 0
      MOV CL, LEN2

      MOV SI, 0
      EQUL: MOV STARTINDEX, SI
            MOV AL, STRING[SI]
            MOV BL, SUBSTRING[0]
            CMP AL, BL
            JNE NEXXTT

            MOV DI, 0
            EQULN:
                  MOV AL, STRING[SI]
                  MOV BL, SUBSTRING[DI]
                  CMP AL, BL
                  JNE NEXT
                  ADD SI, 1
                  ADD DI, 1
                  LOOP EQULN
            NEXT: CMP CX, 0
                  JBE FIND

                  ;MOV NUM, SI
                  ;CALL OUTPUT

                  MOV SI, STARTINDEX
            NEXXTT:     INC SI
                  MOV CH, 0
                  MOV CL, LEN2

                  DEC DX
            JNE EQUL

      JMP NOTFIND
      FIND: MOV CL, LEN1
            MOV BH, LEN2
            CMP CL, BH
            JB NOTFIND
            LEA DX, MESS3
            MOV AH, 09H
            INT 21H

            SUB SI, 1
            MOV ENDINDEX, SI                    ;ENDINDEX WILL BE SI+LENGTH OF
SUBSTRING

            MOV CH, 0
```

```
            MOV CL, LEN1

            MOV DI, 0
            MOV AH, 02H
            PRINT: CMP DI, STARTINDEX
                  JB PRINTC
                  CMP DI, ENDINDEX
                  JA  PRINTC
                  JMP NEXTT
                  PRINTC:MOV DL, STRING[DI]
                        INT 21H
                  NEXTT: ADD DI, 1
            LOOP PRINT

            JMP EXITT

      NOTFIND: LEA DX, MESS4
            MOV AH, 09H
            INT 21H

      EXITT: MOV AH, 4CH
            INT 21H

OUTPUT PROC
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX

      MOV AX, NUM
      AND AL, 00001111B
      MOV BH, AL
      MOV AX, NUM
      AND AL, 11110000B
      RCR AL, 1
      RCR AL, 1
      RCR AL, 1
      RCR AL, 1

      MOV AH, 02H
      MOV DL, AL
      ADD DL, 30H
      INT 21H
      MOV DL, BH
      ADD DL, 30H
      INT 21H

      MOV DL, 0AH
      INT 21H
      MOV DL, 0DH
      INT 21H
      POP DX
      POP CX
      POP BX
      POP AX
      RET
OUTPUT ENDP
END
```

OUTPUT :

```
C:\>A2Q8

Enter your string : HEY THERE WHATS YOUR NAME

Enter your substring that you want to be delete : THERE

The string after deletion is : HEY  WHATS YOUR NAME
```

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

```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter 3 numbers: $'
msg2 db 0AH,0DH,'GCD: $'
msg3 db 0AH,0DH,'LCM: $'
space db ' $'
endl db 0AH,0DH,'$'

val1 dw ?
val2 dw ?
val3 dw ?
num1 dw ?
num2 dw ?
num3 dw ?

.code
print macro msg                ; macro to print a string
      push ax
      push dx
      mov ah, 09h
      lea dx, msg
      int 21h
      pop dx
      pop ax
endm

main proc
      mov ax,@data
      mov ds,ax

      print msg1

      call readnum             ; read first number
      mov val1, ax

      call readnum             ; read second number
      mov val2, ax

      call readnum             ; read third number
      mov val3, ax

      mov dx, 0000h

      mov bx, val1
      mov cx, val2
      loopgcd:
            mov ax, bx
            mov dx, 0000h
            div cx
            cmp dx,0000h
            jz ans
            mov bx,cx
            mov cx,dx
            cmp cx, 0001h
```

```
      jnz loopgcd
      ans:
      mov num1, cx            ; storing gcd of 2 numbers in num1
      mov dx, 0000h

      mov bx, val3

      loopgcd1:
            mov ax, bx
            mov dx, 0000h
            div cx
            cmp dx, 0000h
            jz ans1
            mov bx, cx
            mov cx, dx
            cmp cx, 0001h
      jnz loopgcd1

      ans1:

      print msg2
      mov ax, cx              ; gcd of 3 numbers is stored in CX
      call writenum           ; printing gcd of 3 numbers

      mov ax, val1
      mov bx, val2
      mul bx
      mov bx, num1      ; calculating lcm of 2 numbers by (val1*val2)/gcd(val1,val2)
      div bx

      mov bx, val3      ; then lcm of 3 numbers is [lcm(val1,val2)*val3]/final gcd
      mul bx
      div cx

      print msg3
      call writenum

      exit:
    mov ah, 4ch
    int 21h

main endp

readnum proc near
      ; this procedure will take a number as input from user and store in AX
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
      loopnum:
            mov ah,01h
            int 21h
            cmp al,'0'
            jb skip
            cmp al,'9'
            ja skip
            sub al,'0'
            push ax
            mov ax,bx
            mul cx
            mov bx,ax
            pop ax
            mov ah,00h
```

```
            add bx,ax
      jmp loopnum

      skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum proc near
      ; this procedure will display a decimal number
      ; input : AX
      ; output : none

      push ax
      push bx
      push cx
      push dx

      xor cx, cx
      mov bx, 0ah

      @output:
            xor dx, dx
            div bx                      ; divide AX by BX
            push dx                     ; push remainder onto the STACK
            inc cx
            or ax, ax
      jne @output

      mov ah, 02h                     ; set output function

      @display:
            pop dx                    ; pop a value(remainder) from STACK to DX
            or dl, 30h                ; convert decimal to ascii code
            int 21h
      loop @display

      pop dx
      pop cx
      pop bx
      pop ax

      ret
writenum endp

end main
```

OUTPUT :

```
C:\>A2Q9

Enter 3 numbers: 22
44
66

GCD: 22
LCM: 132
```

20. Write and test a program to Implement Linear search.

```
.MODEL SMALL
.STACK 300H
```

```
.DATA
ARRAY1 DB 11,22,33,44,55
MSG4 DB 0AH,0DH,'Enter size of the array: $'
MSG1 DB 0AH,0DH,'Enter number to be searched: $'
MSG2 DB 0AH,0DH,'FOUND AT POSITION $ '
MSG3 DB 0AH,0DH,'NOT FOUND$'
ENDL DB 0AH,0DH,'$'

SE DB 33H
COUNT DB 00H

.CODE

PRINT MACRO MSG         ; macro to print a string
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      ;int 3
      pop dx
      pop ax
ENDM

MAIN PROC
      MOV AX,@DATA
      MOV DS,AX

START:

      PRINT MSG4
      call readnum                  ; read size of array
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum            ; read the array elements
            mov ARRAY1[BX],AL
            inc BX
      loop rdnxt

      mov cl, COUNT
      PRINT MSG1
      call readnum                  ; read the value to be searched
      mov se,al
      mov al,se
      mov ah,00h
      LEA SI, ARRAY1
      mov bh, 00h
UP:
      MOV BL,[SI]
      CMP AL, BL
      JZ FO
      INC SI
      inc bh
      loop UP
      PRINT MSG3                    ; print 'not found' message
      JMP END1
FO:
      PRINT MSG2                    ; print 'found' message
      mov al, bh
      call writenum                 ; print the position of the found element
END1:
      mov ah, 4ch
```

```
      int 21h

MAIN ENDP

readnum proc near
      ; this procedure is to read a decimal number
      ; input : none
      ; output : AX
      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   push bx                        ; push BX onto the STACK
   push cx                        ; push CX onto the STACK
   push dx                        ; push DX onto the STACK

   XOR CX, CX                     ; clear CX
   MOV BX, 10                     ; set BX=10

   @OUTPUT:                       ; 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 @OUTPUT                    ; jump to label @OUTPUT 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

   POP DX                        ; pop a value from STACK into DX
   POP CX                        ; pop a value from STACK into CX
```

```
   POP BX                          ; pop a value from STACK into BX

   RET                             ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT :

```
C:\>A2Q10

Enter size of the array: 5

56
43
23
87
32
Enter number to be searched: 87

FOUND AT POSITION 3
```

ASSIGNMENT - 3

21. Write and test a MASM program to Implement Binary search. Show the steps. Each step will be succeeded by "Enter" key.

```
.MODEL SMALL
.STACK 300H
.DATA
ARRAY1 DB 11,22,33,44,55
MSG1 DB 0AH,0DH,'Enter size of the array: $'
MSG2 DB 0AH,0DH,'Enter a number to be searched: $'
MSG3 DB 0AH,0DH,'Current array: $'
MSG4 DB 0AH,0DH,'Element found.$ '
MSG5 DB 0AH,0DH,'Element not found.$'
space db ' $'
ENDL DB 0AH,0DH,'$'
key dw ?
mididx dw ?
left dw ?
right dw ?

SE DB 33H
COUNT DB 00H

.CODE

PRINT MACRO MSG
      push ax
      push dx
      mov AH, 09H
      lea DX, MSG
      int 21H
      pop dx
      pop ax
ENDM

MAIN PROC
      MOV AX,@DATA
      MOV DS,AX
```

```
START:

      PRINT MSG1
      call readnum
      mov COUNT, al
      mov cl, COUNT
      mov bx, 00h
      rdnxt:
            PRINT ENDL
            call readnum
            mov ARRAY1[BX],AL
            inc BX
      loop rdnxt
      print msg2
      call readnum
      mov key, ax ;key to be searched

      mov dx, bx  ;last index
      mov bx, 0   ;first index
      LEA SI, ARRAY1
      call binsearch    ;calling proc to perform binary search
      mov ah, 4ch
      int 21h

MAIN ENDP

binsearch proc
      ;input -
      ;bx - left index
      ;dx - right index
      push ax
      push bx
      push cx
      push dx
      push si
      mov cx,key
      dec dx

      @startsearch:
      mov left, bx
      mov right, dx
      inc dx
      mov ah,01h
      int 21h
      @l1:
            xor ah,ah
            mov al,array1[bx]
            call writenum
            print space
            inc bx
            cmp bx,dx
            jne @l1
      print endl
      mov bx,left
      mov dx,right
      cmp bx, dx
      jg @notfound
      mov ax, bx
      add ax,dx   ;ax = bx+dx
      shr ax,1    ; ax = (l+r)/2
      mov left, bx      ; left = bx
      mov mididx,ax     ;mididx = ax
      mov bx, ax  ; bx = ax

      cmp cl, array1[bx]      ;compare key with midval
```

```
      je @found
      jg @bigpivot
      jmp @smallpivot

      @bigpivot:
      mov ax, mididx
      mov bx, left
      inc ax
      mov bx, ax        ;left index = mididx + 1
      jmp @startsearch

      @smallpivot:
      mov ax, mididx
      mov bx, left
      dec ax
      mov dx, ax        ; right index = mididx - 1
      jmp @startsearch

      @notfound:
      print msg5
      jmp @endsearch

      @found:
      print msg4

      @endsearch:
      pop si
      pop dx
      pop cx
      pop bx
      pop ax
      ret
binsearch endp

readnum proc near

      push bx
      push cx
      mov cx,0ah
      mov bx,00h
loopnum:
      mov ah,01h
      int 21h
      cmp al,'0'
      jb skip
      cmp al,'9'
      ja skip
      sub al,'0'
      push ax
      mov ax,bx
      mul cx
      mov bx,ax
      pop ax
      mov ah,00h
      add bx,ax
      jmp loopnum
skip:
      mov ax,bx
      pop cx
      pop bx
      ret
readnum endp

writenum PROC near
   ; this procedure will display a decimal number
```

```
   ; input : AX
   ; output : none

   push bx                          ; push BX onto the STACK
   push cx                          ; push CX onto the STACK
   push dx                          ; push DX onto the STACK

   XOR CX, CX                       ; clear CX
   MOV BX, 10                       ; set BX=10

   @OUTPUT:                         ; 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 @OUTPUT                      ; jump to label @OUTPUT 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

   POP DX                           ; pop a value from STACK into DX
   POP CX                           ; pop a value from STACK into CX
   POP BX                           ; pop a value from STACK into BX
   RET                              ; return control to the calling procedure
writenum ENDP

END MAIN
```

OUTPUT –

```
C:\>A3Q1

Enter size of the array: 5
1
3
5
7
9
Enter a number to be searched: 3
1 3 5 7 9
1 3
3

Element found.
```

22. 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.

```
print macro msg                 ;macro to print a string
      lea dx,msg
      mov ah,09h
      int 21h
endm
read macro n,j1,j2              ;macro to read a number
      j1: mov ah,01h
          int 21h
```

```
          cmp al,0dh
          je j2
          sub al,30h
          mov bl,al
          mov ax,n
          mov dx,0ah
          mul dx
          xor bh,bh
          add ax,bx
          mov n,ax
          jmp j1
      j2: nop
endm
printmul macro n1,l2,l3             ;macro to print a number
                mov bx,000ah
                mov ax,n1
                xor cx,cx
           l2: xor dx,dx
               div bx
               push dx
               inc cx
               cmp ax,0000h
               jne l2
           l3: pop dx
               add dl,30h
               mov ah,02h
               int 21h
               loop l3
endm
.model small
.stack 100h
.data
        num dw 100 dup(0)
        n dw 0
        m dw 0
        msg1 db 'Enter the number of elements:$'
        msg2 db 'Enter an element:$'
        msg3 db 'Current array:$'
        msg4 db '  $'
        msg6 db 10,13,' $'
        exitmsg db 10,13,'Program executed.$'
.code
main proc
        mov ax,@data
        mov  ds,ax
        print msg1
        read n,jump1,jump2    ;read the size of array
        mov cx,n
        mov ax,n
        dec ax
        mov m,ax
        mov si,0000h
        loop1:      print msg2
                    read num[si],jump3,jump4    ;read array elements
                    add si,02h
                    loop loop1
        print msg3
        call display   ;printing original array
        print msg6
        mov si,0000h
        xor cx,cx
        outerloop:  mov ax,num[si]
                    mov di,si
                    push cx
                    push si
```

```
                    mov si,di
                    innerloop:
                                add si,02h
                                cmp ax,num[si]
                                jl check
                                mov ax,num[si]
                                mov di,si

                         check:  inc cx
                                 cmp cx,m
                                 jl innerloop
                    pop si
                    pop cx
                    mov bx,num[si]
                    mov num[di],bx
                    mov num[si],ax
                    push cx
                    push si
                    mov ah,01h            ;enter a character
                    int 21h
                    cmp al,27             ;ascii value of esc key
                    je exit
                    print msg3
                    call display          ;printing current array
                    print msg6
                    pop si
                    pop cx
                    add si,02h
                    inc cx
                    cmp cx,m
                    jl outerloop
        exit: print exitmsg
        mov ah,4ch            ; terminate program
        int 21h
main endp
display proc            ; procedure to print the array
        mov cx,n
        mov si,00h
        l4: push cx
        print msg4
        printmul num[si],l5,l6
        add si,02h
        pop cx
        loop l4
        ret
display endp
end main
```

OUTPUT –

```
C:\>A3Q2
Enter the number of elements:5
Enter an element:4
Enter an element:3
Enter an element:1
Enter an element:5
Enter an element:2
Current array:  4  3  1  5  2
  Current array:  1  3  4  5  2
  Current array:  1  2  4  5  3
  Current array:  1  2  3  5  4
  Current array:  1  2  3  4  5

Program executed.
```