System Programming

`

ASSIGNMENT 2

Name – Nikhil Badyal

Roll – 001810501069

Class – BCSE 3rd year

Group – A2

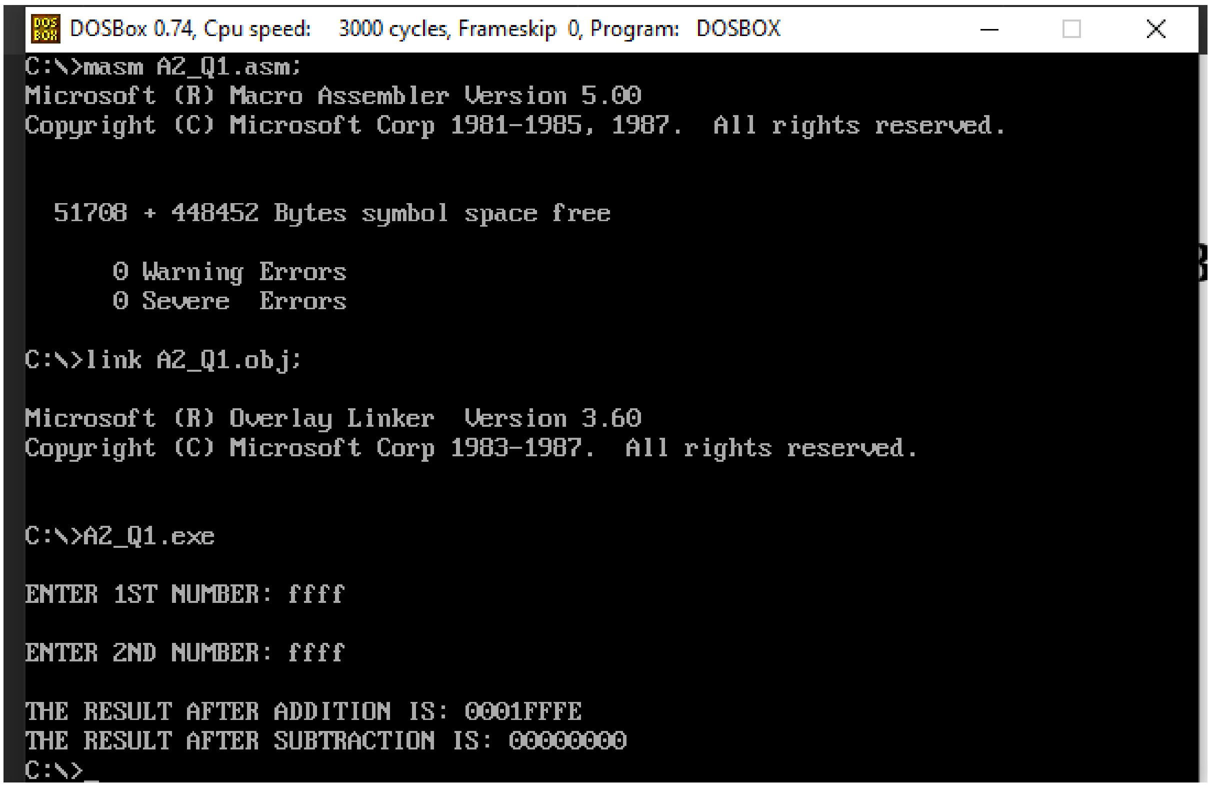
1. Write and test a MASM program to add and subtract two 16 bit numbers.

<u>Code:</u>

```
.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   ;printing first msg
    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   ;exit
        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 ;find whether it is letter or digit.39h is the ascii value of 9
          jg letter1
          and al,0fh ;if digit then convert it's ascii value to real value
          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 eg. A+37=41 else only add 30H
        l4: add dl, 30H   ;eg9+30=39  ascii of 9
         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:


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

```
C:\>masm A2_Q1.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51708 + 448452 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q1.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q1.exe

ENTER 1ST NUMBER: ffff

ENTER 2ND NUMBER: ffff

THE RESULT AFTER ADDITION IS: 0001FFFE
THE RESULT AFTER SUBTRACTION IS: 00000000
C:\>_
```

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

Code:

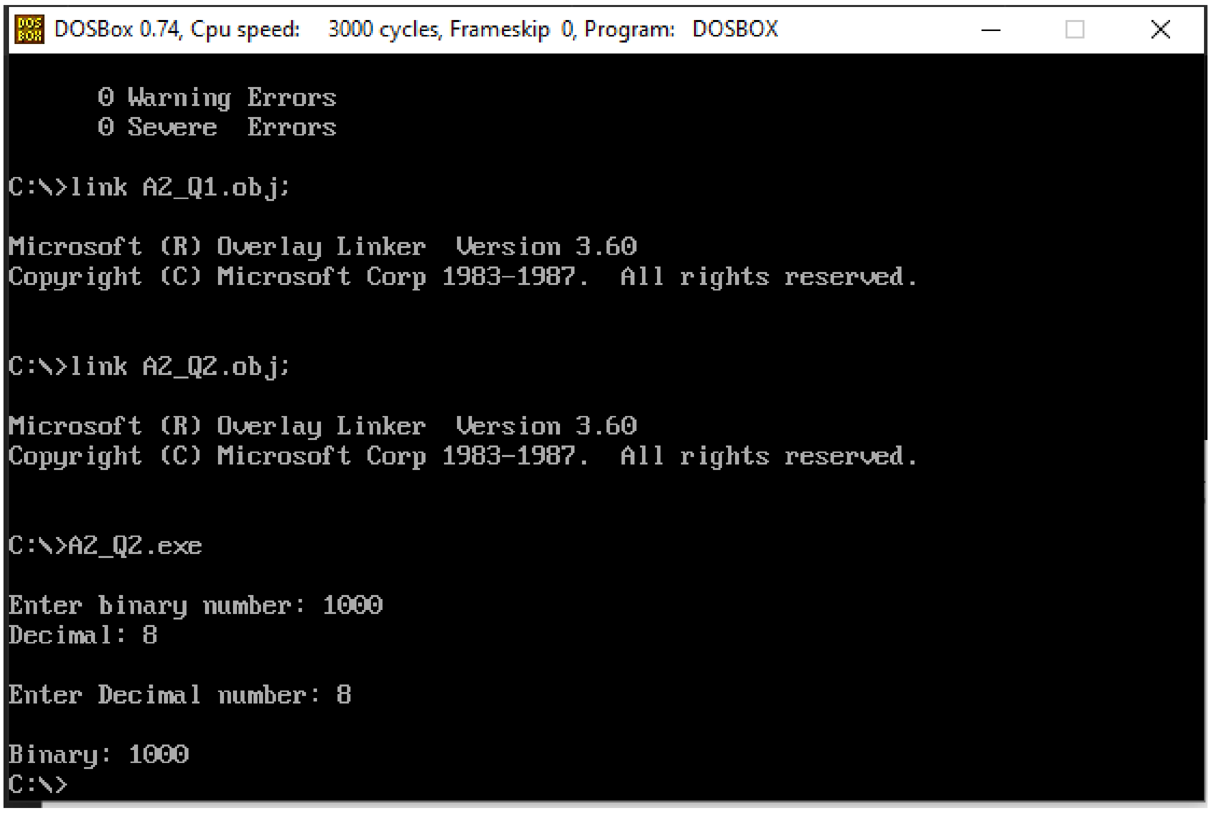
```
.model small
.stack 300h
.data
        msg1 db 0AH,0DH,'Enter binary number: $'
        msg2 db 0AH,0DH,'Decimal: $'
        msg3 db 0AH,0DH,'Enter Decimal number: $'
        msg4 db 0AH,0DH,'Binary: $'
        space db ' $'
        endl db 0AH,0DH,'$'

    binno db 17   ;MAX NO. OF CHARRRACTERS ALLOWED
     db ?         ;NO. OF CHARACTERS ENTERED BY USER
     db 17 dup(0) ;INITIALIZING
        str1 db 20 dup('$')
```

```
            str2 db 20 dup('$')
        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]     ;NUMBER OF CHARACTERS ENTERED BY USER
            mov ch, 00h
```

```
inc si          ;NOW SI POINTS TO THE FIRST CHARACTER OF BINNO

      mov ax,00h
      loop1:
       mov bl, [si]
       sub bl, '0'
       mov bh, 00h
       mov dx,02h
       mul dx       ; ax = ax * dx
       add ax, bx
       ;call writenum
       ;call 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  and remainder will store to the dx
              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:



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

```
      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q1.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>link A2_Q2.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q2.exe

Enter binary number: 1000
Decimal: 8

Enter Decimal number: 8

Binary: 1000
C:\>
```

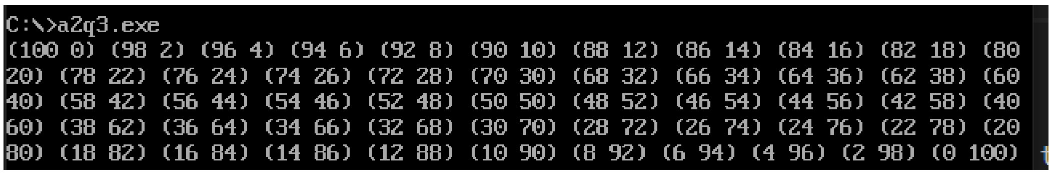
3. **Write and test a program to print pairs of even numbers where the summation of the numbers in each pair is 100.**

<u>Code:</u>

```
.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

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:\>a2q3.exe
(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)
```

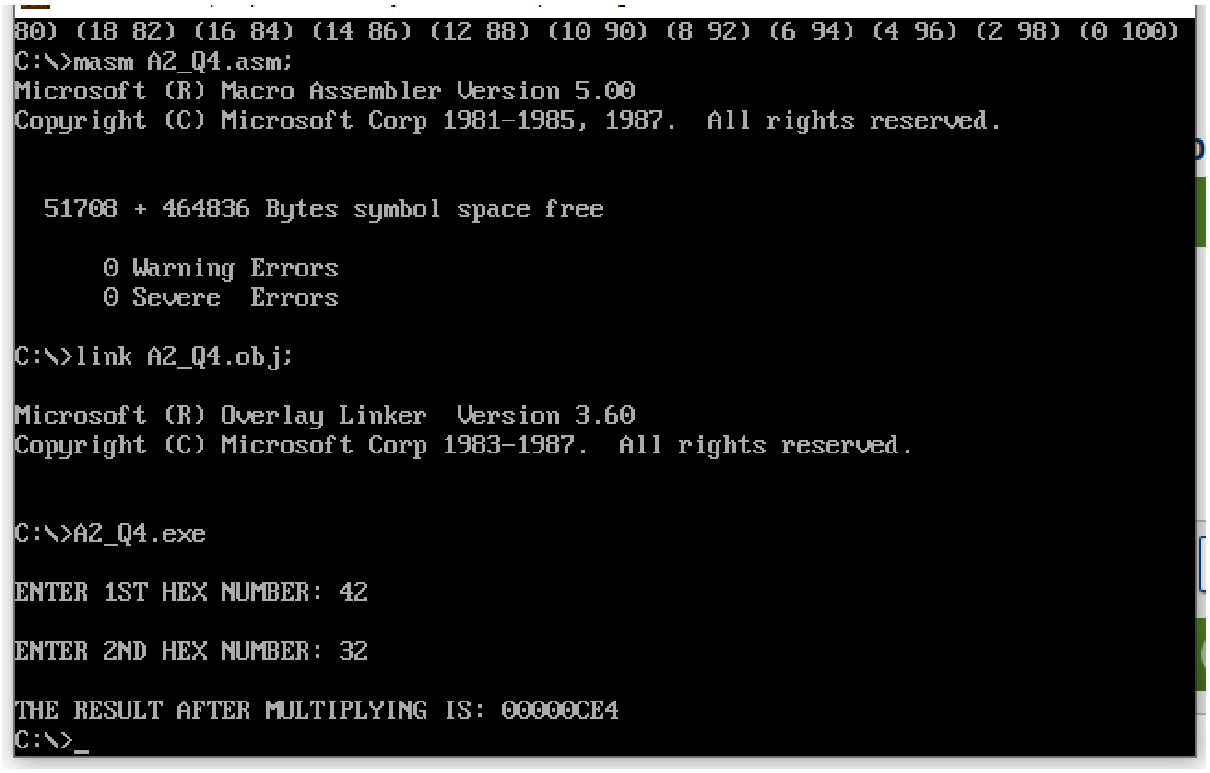
4. Write and test a MASM program to multiply two 32 bit numbers.

<u>Code:</u>

```
.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 whether it is letter or digit.39h is the ascii 9
         jg letter1
         and al,0fh ;if it is digit then convert it's ascii value to real value
         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:



```
80) (18 82) (16 84) (14 86) (12 88) (10 90) (8 92) (6 94) (4 96) (2 98) (0 100)
C:\>masm A2_Q4.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51708 + 464836 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q4.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q4.exe

ENTER 1ST HEX NUMBER: 42

ENTER 2ND HEX NUMBER: 32

THE RESULT AFTER MULTIPLYING IS: 00000CE4
C:\>_
```

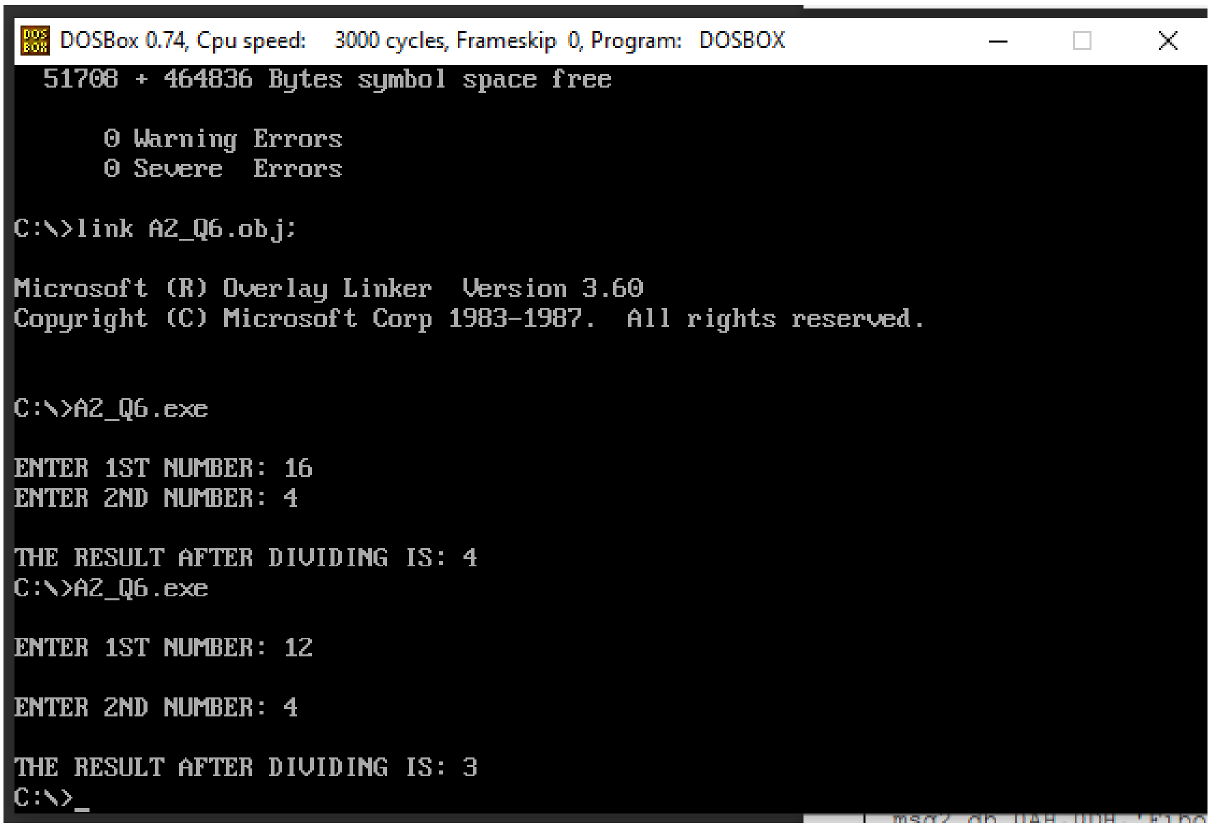
5. Write and test a MASM program to divide a 16 bit number by an 8 bit number.

Code:

```
.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:



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

```
  51708 + 464836 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q6.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q6.exe

ENTER 1ST NUMBER: 16
ENTER 2ND NUMBER: 4

THE RESULT AFTER DIVIDING IS: 4
C:\>A2_Q6.exe

ENTER 1ST NUMBER: 12

ENTER 2ND NUMBER: 4

THE RESULT AFTER DIVIDING IS: 3
C:\>_
```

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

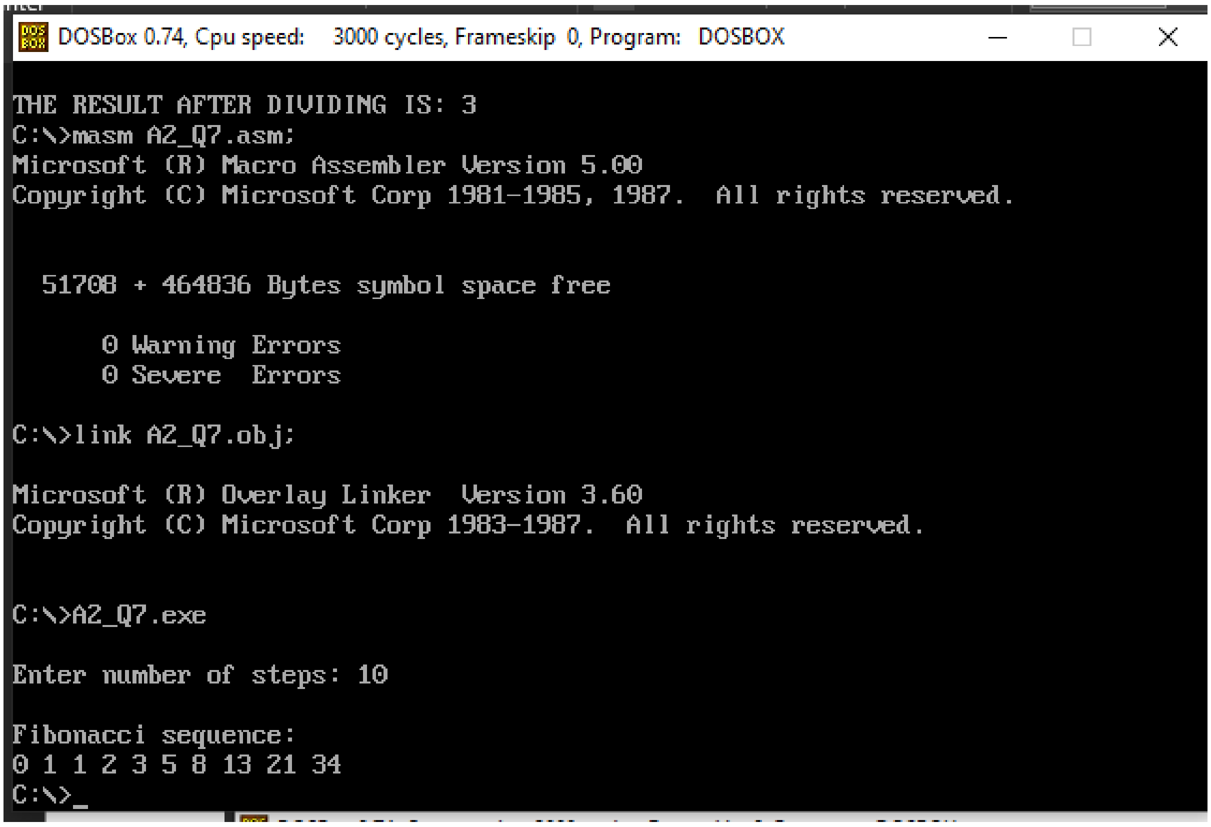
Code:

```
.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:



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

```
THE RESULT AFTER DIVIDING IS: 3
C:\>masm A2_Q7.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51708 + 464836 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q7.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q7.exe

Enter number of steps: 10

Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
C:\>_
```

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

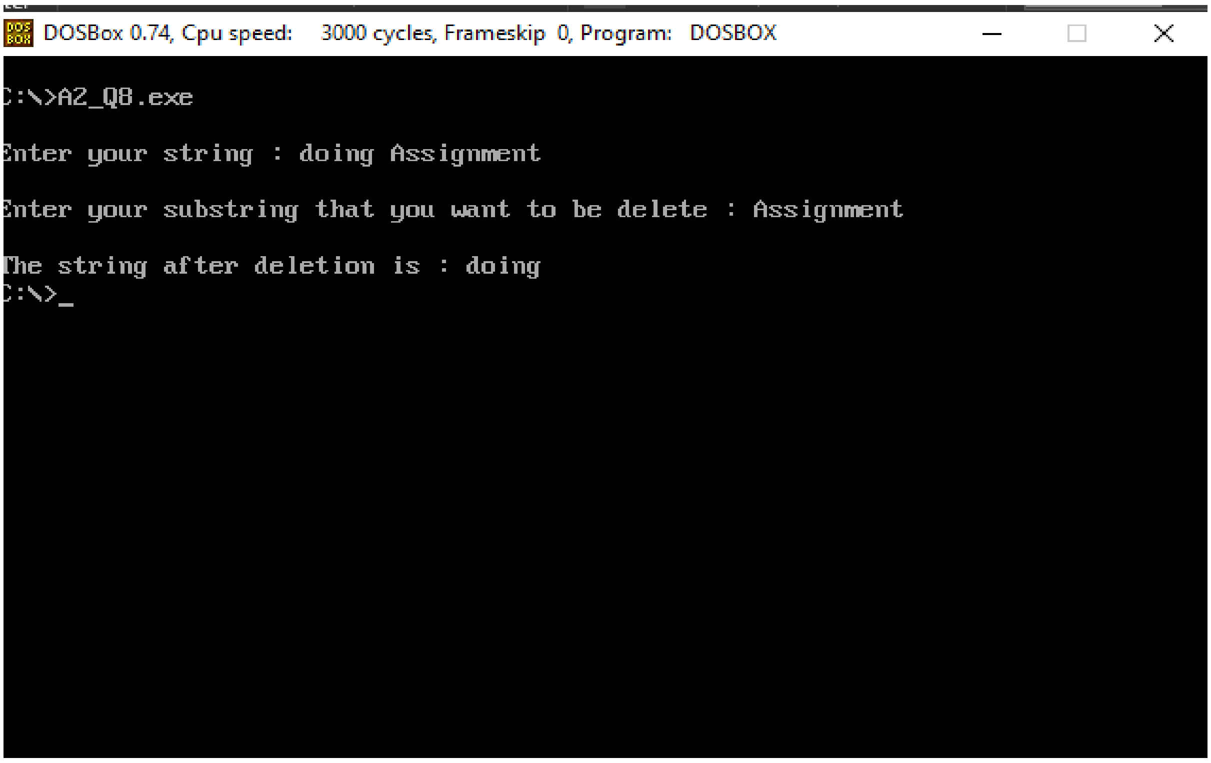
Code:

```
.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

END
```

Output:


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

```
C:\>A2_Q8.exe

Enter your string : doing Assignment

Enter your substring that you want to be delete : Assignment

The string after deletion is : doing
C:\>_
```

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

<u>Code:</u>

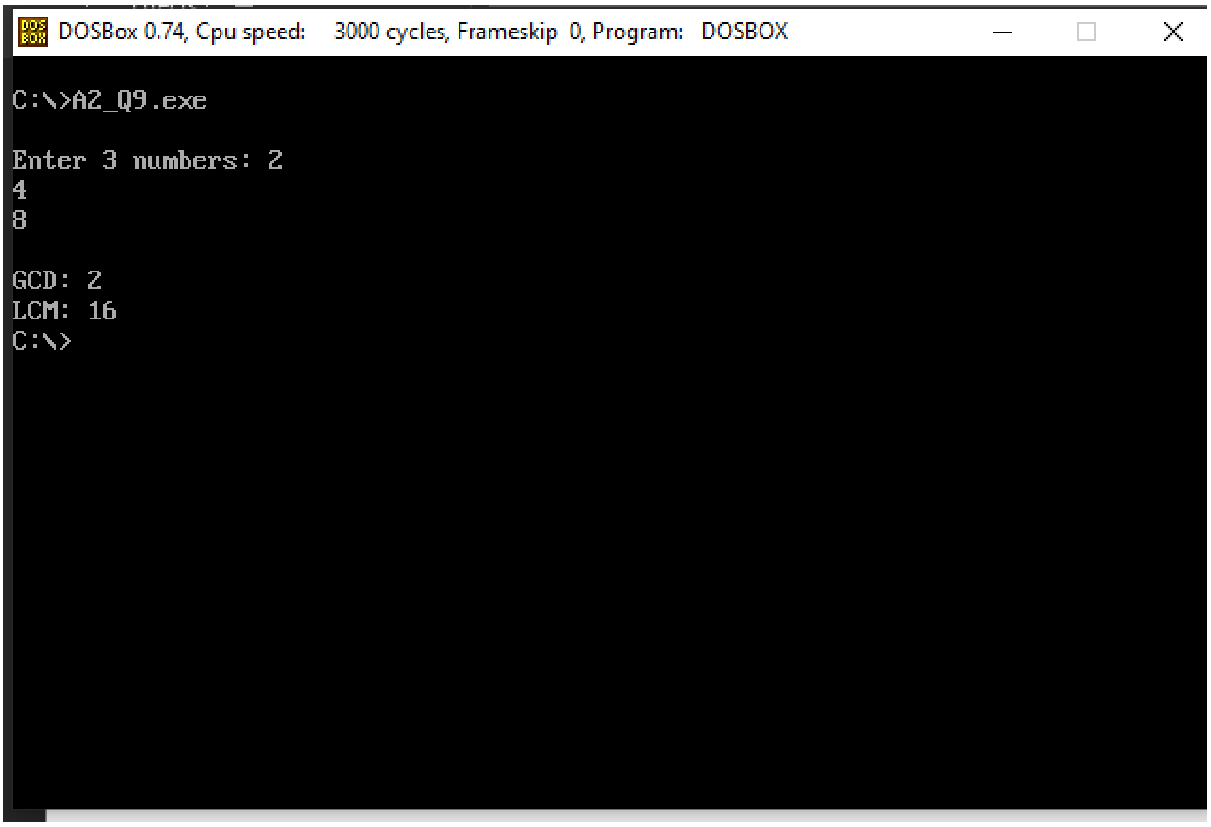
```
.model small
.stack 300h
.data
msg1 db 0AH,0DH,'Enter 3 numbers: $'
msg2 db 0AH,0DH,'GCD: $'
msg3 db 0AH,0DH,'LCM: $'
space db ' $'
endl db 0AH,0DH,'$'

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

.code
```

```
print macro msg
        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 val1, ax

        call readnum
        mov val2, ax

        call readnum
        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
                ;mov ax,bx
                ;call writenum
                ;mov ax,cx
                ;call writenum
                cmp cx, 0001h
        jnz loopgcd
        ans:
        mov num1, cx
        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
            call writenum

            mov ax, val1
            mov bx, val2
            mul bx
            mov bx, num1
            div bx

            mov bx, val3
            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:



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

```
C:\>A2_Q9.exe

Enter 3 numbers: 2
4
8

GCD: 2
LCM: 16
C:\>
```

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

<u>Code:</u>

```
.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 message
        JMP END1
        FO:
        PRINT MSG2 ; print 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:
```

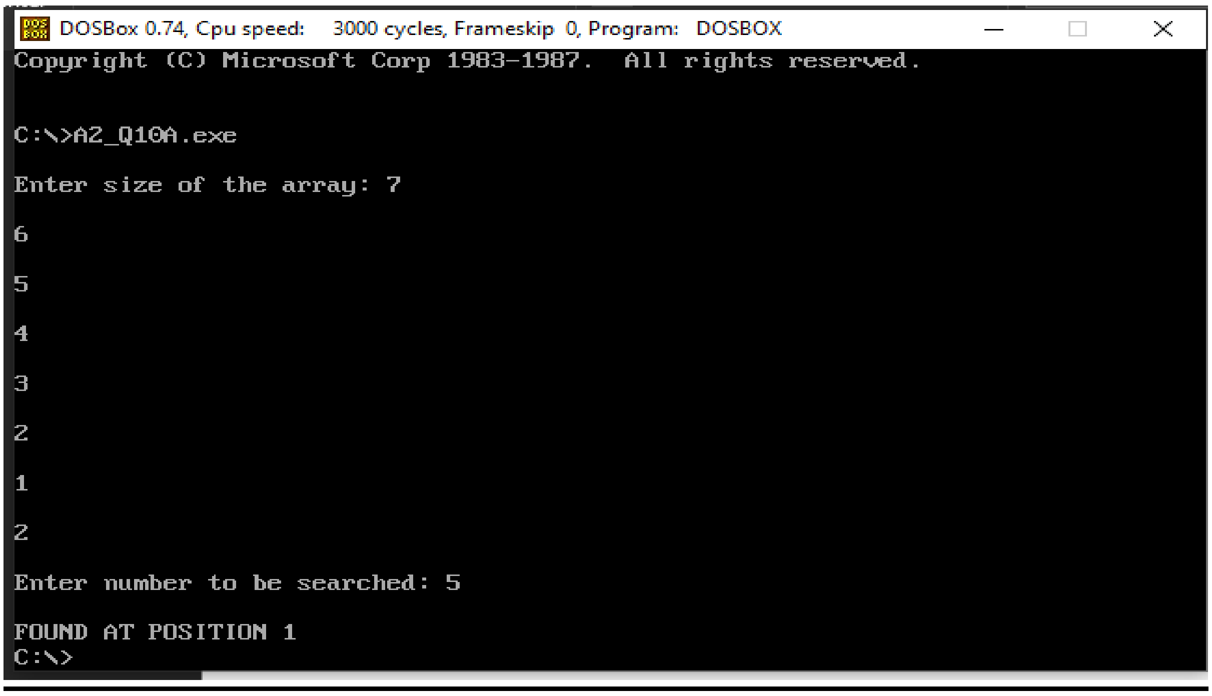
```
        mov ah,01h
        int 21h
        cmp al,'0'
        jb skip
        cmp al,'9'
        ja skip
        sub al,'0'
        push ax
        mov ax,bx
        mul cx
        mov bx,ax
        pop ax
        mov ah,00h
        add bx,ax
        jmp loopnum
        skip:
        mov ax,bx
 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

```
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
```

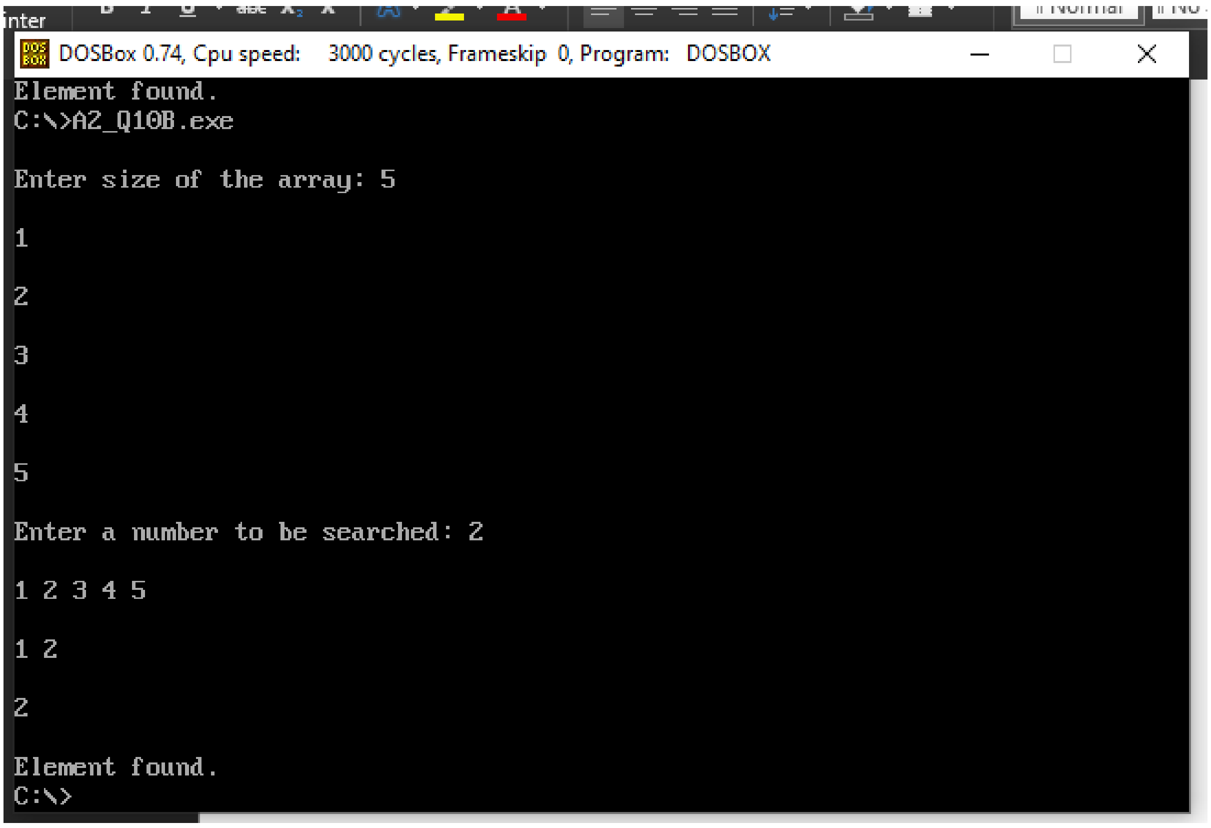
```
        mov ARRAY1[BX],AL
        inc BX
        loop rdnxt
        print msg2
        call readnum
        mov key, ax ;key to be searched
        mov dx, bx ;last index
        mov bx, 0 ;first index
        LEA SI, ARRAY1
        call binsearch ;calling proc to perform binary search
        mov 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:



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

```
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q10A.exe

Enter size of the array: 7

6

5

4

3

2

1

2

Enter number to be searched: 5

FOUND AT POSITION 1
C:\>
```


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

```
Element found.
C:\>A2_Q10B.exe

Enter size of the array: 5

1

2

3

4

5

Enter a number to be searched: 2

1 2 3 4 5

1 2

2

Element found.
C:\>
```

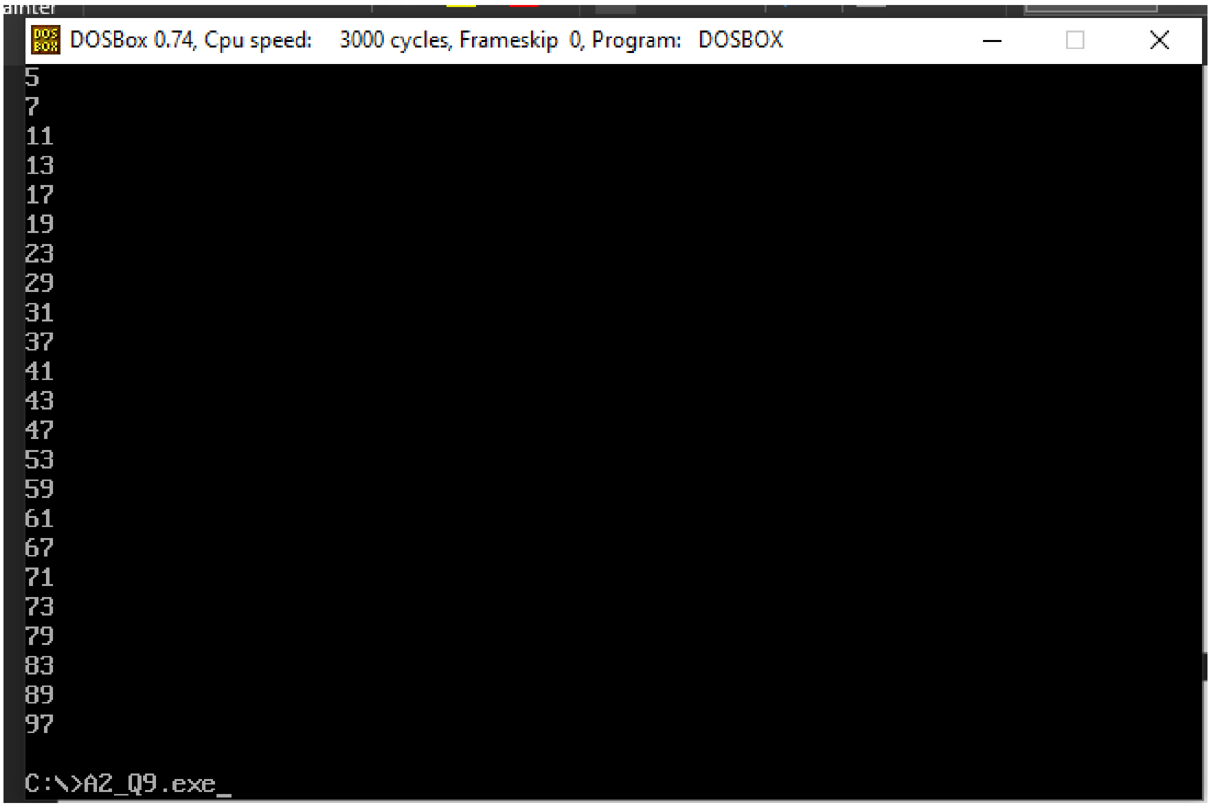
10. Write and test a MASM program to print prime numbers between 1 to 100.

<u>Code:</u>

```
.model small
.stack 100h

.data
x db 0ah, 0dh, '$'

.code
main proc
mov ax, @data
mov ds, ax
mov cl, 2
mov ch, 00h
l1: mov bl, 1
mov bh, 0
l2: mov ax, cx
div bl
cmp ah, 0
jne l3
```

```
inc bh
l3: inc bl
cmp bl, cl
jne l2

cmp bh, 1
jg l4

mov ax, cx
call displayNumber

l4: inc cl
cmp cl, 100
jne l1

mov ah, 4ch
int 21h

main endp
displayNumber proc
mov bl, 10
mov bh, 00h
l5: mov ah, 00h
div bl
push ax
inc bh
cmp al, 0
jne l5
l6: pop dx
mov dl, dh
mov dh, 0
add dl, 48
mov ah, 02h
int 21h
dec bh
cmp bh, 0
jne l6

lea dx, x
mov ah, 09h
int 21h
ret

displayNumber endp
end
```

Output:


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

```
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97

C:\>A2_Q9.exe_
```

11. Write and test a MASM program perform selection and insertion sort.

<u>Code:</u>

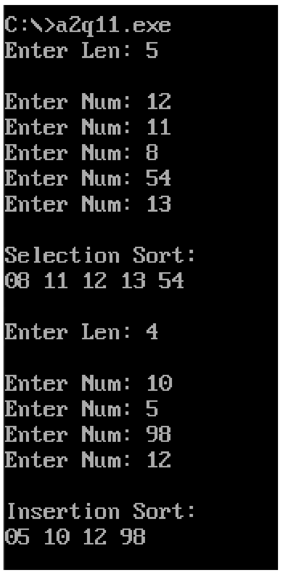
```
.model tiny
.stack 100
.data
linefeed   db 13, 10, "$"
prompt1    db "Enter Len: $"
prompt2    db "Enter Num: $"
msg1       db "Array is: $"
msg2       db "Selection Sort:$"
msg3       db "Insertion Sort:$"
len        db ?
nums       db 10 DUP(?), "$"
dec_out    db 2 DUP(?), "$"

.code                   ; code segment
call main
mov  ax, 4c00h          ; terminate properly
int  21h

main proc
   mov ax, @data
```

```
    mov ds, ax

    call get_arr_inp
    call ins_linefeed

    call selection_sort

    mov dx, offset msg2
    call show_msg
    call ins_linefeed
    call disp_arr_output
    call ins_linefeed
    call ins_linefeed

    call get_arr_inp
    call ins_linefeed

    call insertion_sort
    mov dx, offset msg3
    call show_msg
    call ins_linefeed
    call disp_arr_output
    call ins_linefeed

    ret
main endp

; insertion sort
insertion_sort proc
    push ax
    push bx
    push cx
    push dx

    mov cl, 1
    mov bx, offset nums

ins_outer:
    mov ch, 0
    mov di, cx
    mov dl, nums[di]
    mov si, di
    dec si

ins_inner:
    cmp si, 0
    jl ins_outer_update
    cmp nums[si], dl
    jbe ins_outer_update

    mov ch, nums[si]
    mov nums[di], ch
    dec di
    dec si
    jmp ins_inner


ins_outer_update:
    mov nums[si+1], dl
    inc cl
    cmp cl, len
    jl ins_outer

    pop dx
    pop cx
    pop bx
    pop ax
    ret
insertion_sort endp

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

    mov cl, len
    mov bx, offset nums
sel_outer:
    ; call disp_arr_output
    ; call ins_linefeed

    mov ch, 0
    inc ch
    mov dh, cl
    mov dl, [bx]
sel_inner:
    push cx
    xchg cl, ch
    mov ch, 0
    add bx, cx
    mov al, [bx]
    cmp dl, al
    jbe sel_inner_upd
    mov dl, al
    mov dh, cl

sel_inner_upd:
    sub bx, cx
    pop cx
    inc ch
    cmp ch, cl
    jl sel_inner

sel_done_inner:
    mov ah, [bx]
    push bx
    add bl, dh
    adc bh, 0
    mov [bx], ah
    pop bx

    mov [bx], dl

    inc bx
    dec cl
    cmp cl, 1
    jg sel_outer

    pop dx
    pop cx
    pop bx
    pop ax
    ret
selection_sort endp

; get array as input
get_arr_inp proc
    push ax
    push bx
    push cx
    push dx

    mov dx, offset prompt1
    call show_msg
    call get_dec_val
    mov len, al

    call ins_linefeed

    mov cx, 0
get_arr_elems_loop:
    mov bx, offset nums
    add bx, cx

    mov dx, offset prompt2
    call show_msg
    call get_dec_val
    mov [bx], al

    inc cl
    cmp cl, len
    jl get_arr_elems_loop

done_get_arr_elems:

    pop dx
    pop cx
    pop bx
    pop ax
    ret
get_arr_inp endp

disp_arr_output proc
    push ax
    push bx
    push cx
    push dx

    mov cl, 0
    mov bx, offset nums
disp_arr_output_loop:
    mov al, [bx]
    mov ah, 0

    call disp_dec_val
    mov al, 32
    call show_char

    inc bx
    inc cl
    cmp cl, len
    jl disp_arr_output_loop

    pop dx
    pop cx
    pop bx
    pop ax
    ret
disp_arr_output endp
```

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

   mov dx, 0
get_characters:
   call get_char
   cmp al, 13 ; cmp w/ [enter]
   je done

   sub al, 48

   mov bx, dx
   mov cl, 3
   shl bx, cl
   shl dx, 1
   add dx, bx
   add dl, al
   jnc get_characters
   add dh, 1
   jmp get_characters

done:
   mov ax, dx

   pop dx
   pop cx
   pop bx
   ret
get_dec_val endp

; display ax value in decimal
disp_dec_val proc
   push ax
   push bx
   push cx
   push dx

   mov cl, 2
disp_dec_val_loop:
   dec cl
   cmp cl, 0
   jl disp_dec_val_loop_done
   mov bx, offset dec_out
   push cx
   mov ch, 0
   add bx, cx
   pop cx

   mov ch, 10
   div ch
   push ax
   add ah, 48
   mov [bx], ah
   pop ax

   mov ah, 0
   jmp disp_dec_val_loop

disp_dec_val_loop_done:
   mov dx, offset dec_out
   call show_msg

   pop dx
   pop cx
   pop bx
   pop ax
   ret
disp_dec_val endp

; show character, ascii value in al
show_char proc
   push ax
   push dx

   mov dl, al
   mov ah, 2
   int 21h

   pop dx
   pop ax
   ret
show_char endp

; show message, location in dx
show_msg proc
   push ax
   mov ah, 9
   int 21h
   pop ax
   ret
show_msg endp
```

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

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

end
```

Output:



```
C:\>a2q11.exe
Enter Len: 5

Enter Num: 12
Enter Num: 11
Enter Num: 8
Enter Num: 54
Enter Num: 13

Selection Sort:
08 11 12 13 54

Enter Len: 4

Enter Num: 10
Enter Num: 5
Enter Num: 98
Enter Num: 12

Insertion Sort:
05 10 12 98
```

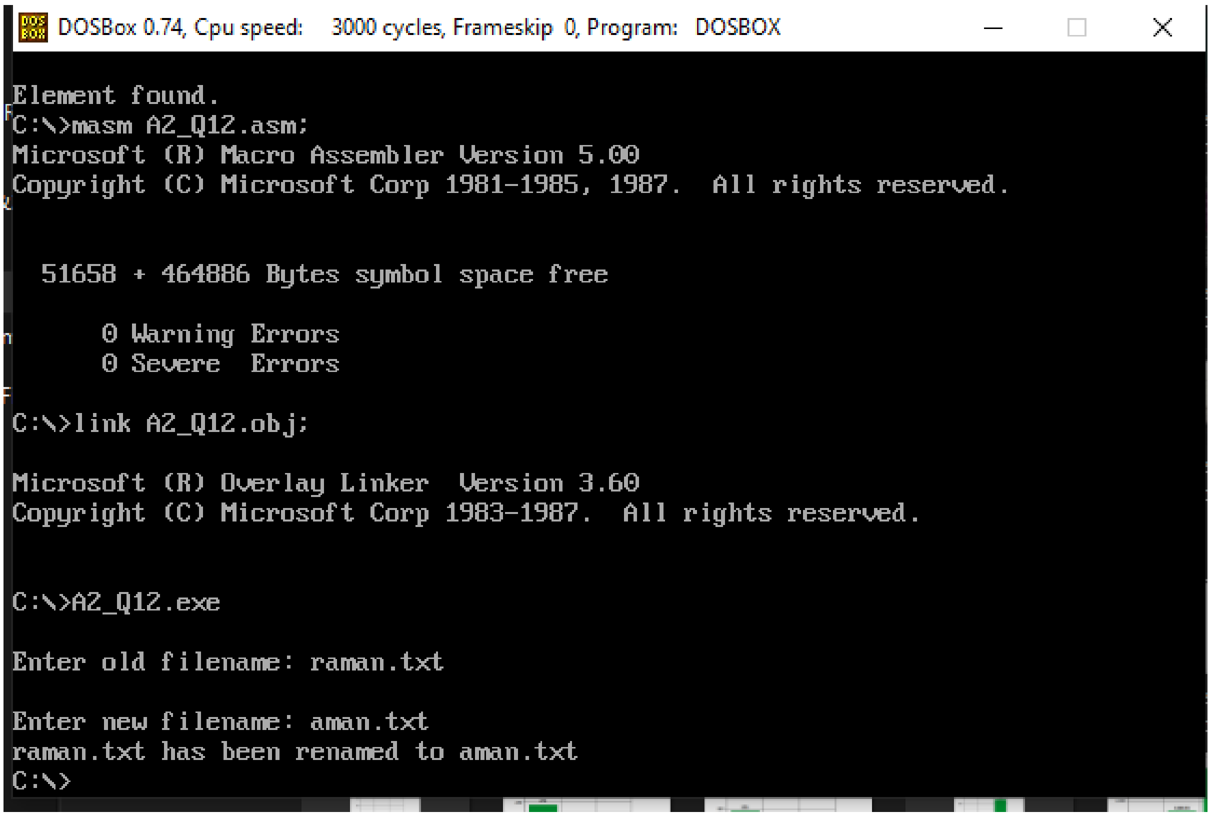
12. Write and test a MASM program to rename a file.

Code:

```
.model small
.stack 64
.data
        msg1 db 0AH,0DH,'Enter old filename: $'
        msg2 db 0AH,0DH,'Enter new filename: $'
        ;old1 db 'ABC.TXT',0
        old db 80 dup('$')
        ;new1 db 'DEF.TXT',0
        new db 80 dup('$')
        sucmsg db 'has been renamed to $'
    failmsg db 'not found. ERROR!!!$'

.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
    mov es,ax

        print msg1
        lea SI, old
        call readstring
```

```
            print msg2
            lea SI, new
            call readstring


            mov ax,@data
            mov ds,ax
            mov es,ax
        lea dx,old  ;ds:dx points to the ASCIIZ string old,0
        lea di,new  ;es:di points to the ASCIIZ string new,0
        mov ah,56h  ;DOS function 56h is used for renaming
        int 21h
        jc error    ;if there is an error carry flag is set
        print old
            print sucmsg
            print new
        jmp exit
error:
            print old
        print failmsg


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


readstring proc near
read:
            mov ah, 01h
            int 21h
            cmp al, 13
            je done

            mov [SI],al

            inc SI

            jmp read


done:

        mov al, 0

        mov [SI],al

            ret

readstring endp


end main
```

<u>Output:</u>



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

```
Element found.
C:\>masm A2_Q12.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51658 + 464886 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q12.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>A2_Q12.exe

Enter old filename: raman.txt

Enter new filename: aman.txt
raman.txt has been renamed to aman.txt
C:\>
```

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

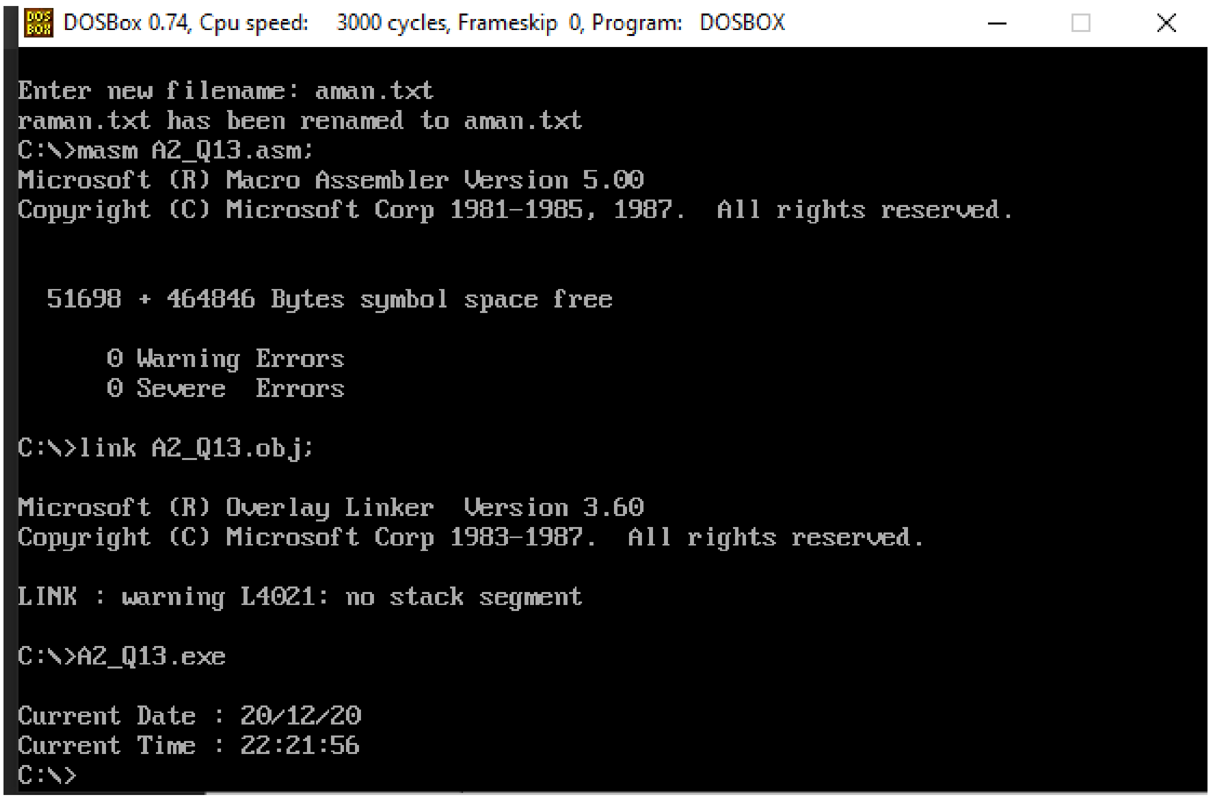
<u>Code:</u>

```
;Used INTERRUPTS
;AH=2AH  //Gets the system date
;AH=02h // Displays the ascii value in DOS Prompt
;For 2AH
; Day is in DL
; Month is in DH
; Year is in CX


;Declaration Part
.MODEL SMALL
.DATA
        msg1 db 0AH,0DH,'Current Date : $'
        msg2 db 0AH,0DH,'Current Time : $'
        endl db 0AH,0DH,'$'

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

main proc
START: MOV AX,@DATA
MOV DS,AX

print msg1

;Day Part
DAY:
MOV AH,2AH    ; To get System Date
INT 21H
MOV AL,DL     ; Day is in DL
AAM
MOV BX,AX
CALL DISP
```

```
MOV DL,'/'
MOV AH,02H    ; To Print / in DOS
INT 21H

;Month Part
MONTH:
MOV AH,2AH    ; To get System Date
INT 21H
MOV AL,DH     ; Month is in DH
AAM
MOV BX,AX
CALL DISP

MOV DL,'/'    ; To Print / in DOS
MOV AH,02H
INT 21H

;Year Part
YEAR:
MOV AH,2AH    ; To get System Date
INT 21H
ADD CX,0F830H ; To negate the effects of 16bit value,
MOV AX,CX     ; since AAM is applicable only for AL (YYYY -> YY)
AAM
MOV BX,AX
CALL DISP

print msg2
;Hour Part
HOUR:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,CH     ; Hour is in CH
AAM
MOV BX,AX
CALL DISP

MOV DL,':'
MOV AH,02H    ; To Print : in DOS
INT 21H

;Minutes Part
MINUTES:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,CL     ; Minutes is in CL
AAM
```

```
MOV BX,AX
CALL DISP

MOV DL,':'    ; To Print : in DOS
MOV AH,02H
INT 21H

;Seconds Part
Seconds:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,DH     ; Seconds is in DH
AAM
MOV BX,AX
CALL DISP



;To terminate the Program

MOV AH,4CH     ; To Terminate the Program
INT 21H
main endp

;Display Part
DISP PROC
MOV DL,BH      ; Since the values are in BX, BH Part
ADD DL,30H     ; ASCII Adjustment
MOV AH,02H     ; To Print in DOS
INT 21H
MOV DL,BL      ; BL Part
ADD DL,30H     ; ASCII Adjustment
MOV AH,02H     ; To Print in DOS
INT 21H
RET
DISP ENDP      ; End Disp Procedure


end main      ; End of MAIN
```

Output:


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

```
Enter new filename: aman.txt
raman.txt has been renamed to aman.txt
C:\>masm A2_Q13.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51698 + 464846 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link A2_Q13.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\>A2_Q13.exe

Current Date : 20/12/20
Current Time : 22:21:56
C:\>
```