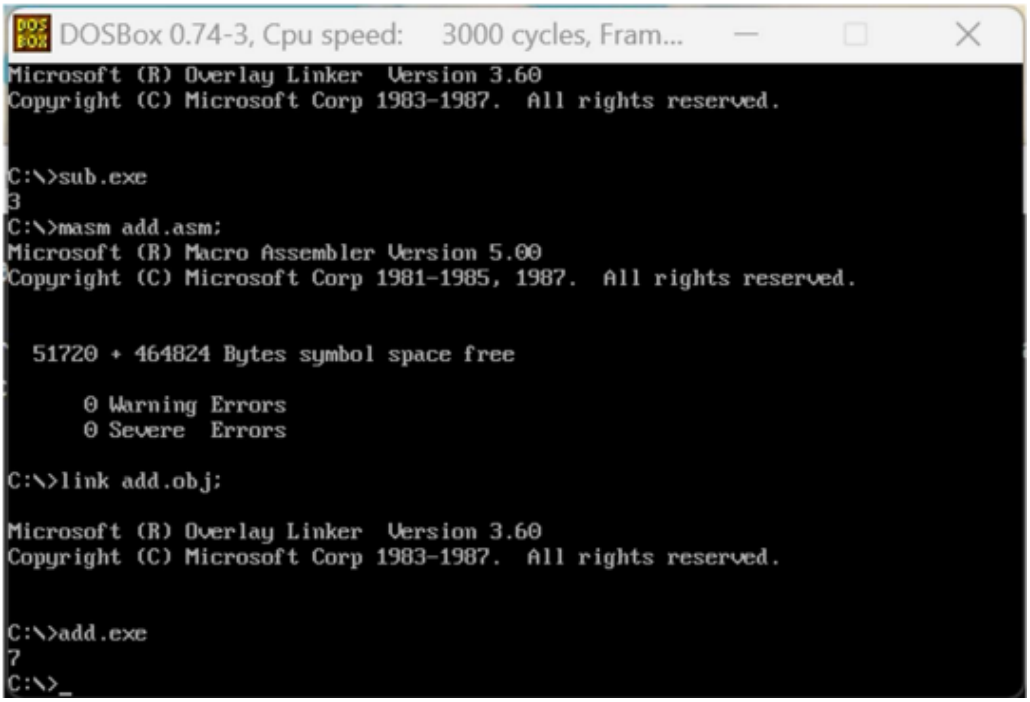
ASM Assignment 1

Submitted By: Kaji Manirul Islam
Roll: 0022-1100-1100
Dept: Information Technology
Class: 2nd Year 3rd Semester

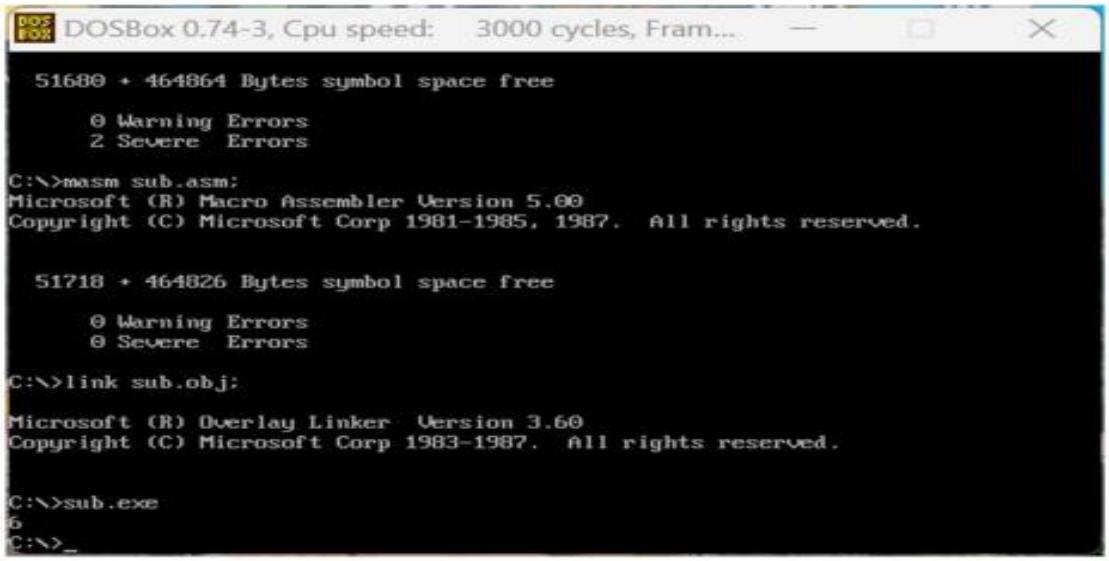
Q1. Write an Assembly Language Program to add two byte integers and store the result in DX register.

```
.model small
.stack 100h
.date
.code
main proc
        mov bx, 4
        mov dx, 3
        add dx, bx

        add dx, 48
        mov ah, 2
        int 21h
        mov ah, 4ch
        int21h
main endp
end main
```



Q2. Write an 8086 Assembly Language Program to subtract two 8-bit signed integers. The numbers can be stored in the data segment.

```
.model small
.stack 100h
.data
        num1 db 9
        num2 db 3
        result db ?
.code
main proc
        mov ax, @data
        mov ds, ax
         mov al, num1
         sub al, num2
        mov result, al

        mov dl, result
        add dl, 48

        mov ah, 2
        int 21h

        mov ah, 4Ch
        int 21h
main endp
end main
```



Q3. Write an Assembly Language Program to print your name, which is stored in memory as a string.

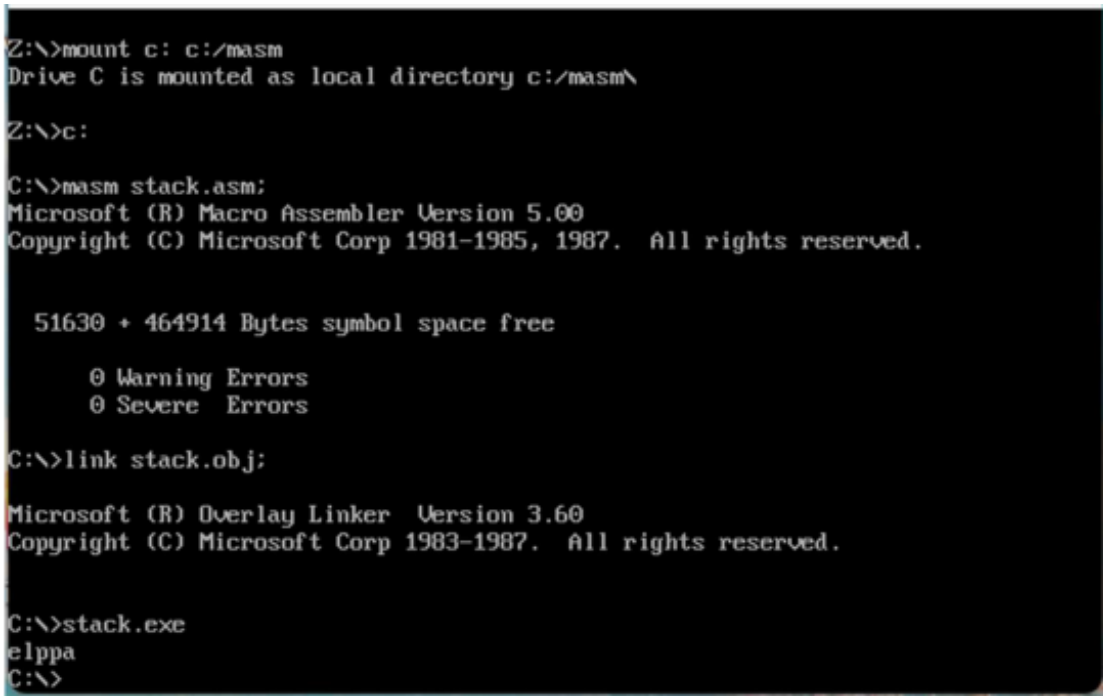
```
.model small
.stack 100h
.data
	name db "KMI$"
.code
main proc
	mov ax, @data
	mov ds, ax

	lea dx, name

	mov ah, 09h
	int 21h

	mov ah, 4Ch
	int 21h
main endp
end main
```

```
C:\>masm k.asm
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993.  All rights reserved.

 Invoking: ML.EXE /I. /Zm /c /Ta k.asm

Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

 Assembling: k.asm

C:\>link k.obj

Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Run File [k.exe]:
List File [nul.map]:
Libraries [.lib]:
Definitions File [nul.def]:

C:\>k.exe
KMI
C:\>
```

Q4. Write an Assembly Language Program to reverse a string using stack and display the result.

```
.model small
.stack 100h
.data
        str db 'apple','$'
        rev db 50 dup('$')
        len dw 5
.code
main proc far
        mov ax,@data
        mov ds,ax
        lea dx,str
        mov ax,'#'
        push ax
        lea si,str
        lea di,rev
push_loop:
        mov al,[si]
        cmp al,'$'
        je pop_loop
        mov ah,0
        push ax
        inc si
        jmp push_loop
pop_loop:
        pop bx
        cmp bl,'#'
        je output
        mov [di],bl
        inc di
        jmp pop_loop
output:
        mov bl,'$'
        mov [di],bl
        lea dx,rev
        mov ah,09h
```

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

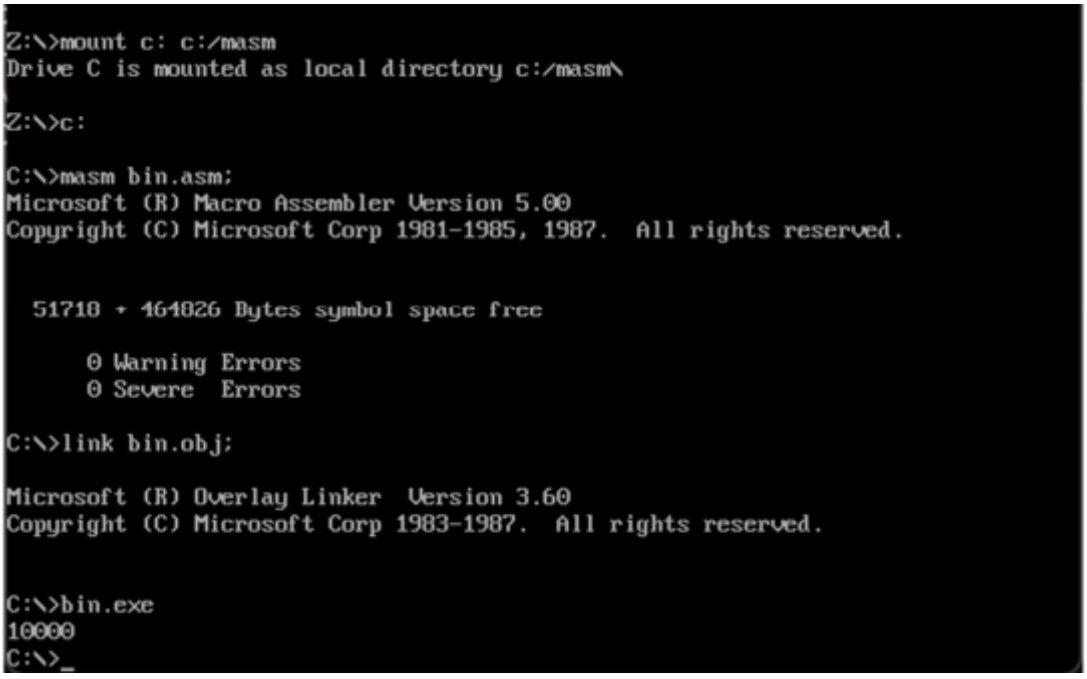


Q5. Write an 8086 Assembly Language Program which will ask for a number and the no. will be taken from keyboard. Print the number in decimal, binary and hexadecimal format.

TO BINARY:

```
.MODEL SMALL
.STACK 100H
.DATA
        d1 DW 16
.CODE
MAIN PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        ; Load the value stored in variable d1
         MOV AX, d1
        ; Convert the value to binary and print the value
        CALL PRINT
        ; Interrupt to exit
        MOV AH, 4CH
        INT 21H
MAIN ENDP
PRINT PROC
        ; Initialize count
        MOV CX, 0
        MOV DX, 0
label1:
        ; If AX is zero, jump to print1
        CMP AX, 0
        JE print1
        ; Initialize BX to 2
        MOV BX, 2
        ; Divide AX by BX to convert it to binary
         DIV BX
         ; Push the remainder (DX) in the stack
        PUSH DX
        ; Increment the count
        INC CX
        ; Set DX to 0
        XOR DX, DX
        JMP label1
```

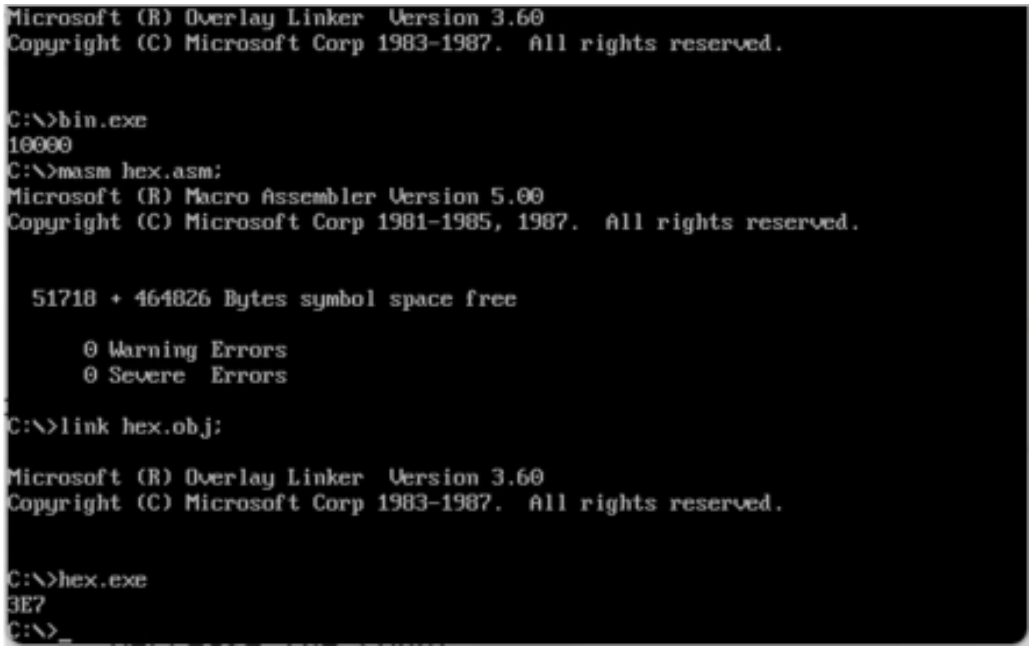
```
print1:
        ; Check if count is greater than zero
        CMP CX, 0
        JE exit
        ; Pop the top of the stack into DX
        POP DX
        ; Add 48 to DX so that it represents the ASCII value of digits
        ADD DX, 48
        ; Interrupt to print a character
         MOV AH, 02H
        INT 21H

        ; Decrease the count
        DEC CX
        JMP print1
exit:
        RET
PRINT ENDP
END MAIN
```



TO HEXADECIMAL:

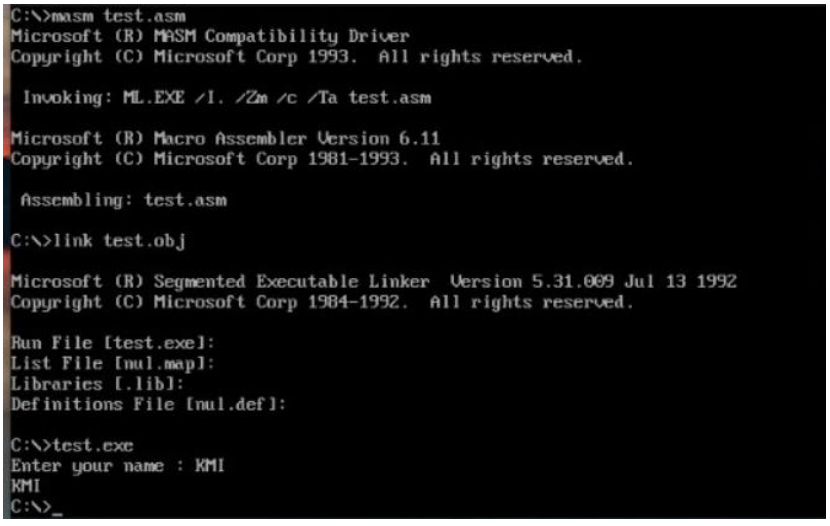
```
.MODEL SMALL
.STACK 100H
.DATA
        d1 DW 999
.CODE
MAIN PROC FAR
        MOV AX, @DATA
        MOV DS, AX
        ; Load the value stored in variable d1
        MOV AX, d1
        ; Convert the value to hexadecimal and print the value
        CALL PRINT
        ; Interrupt to exit
        MOV AH, 4CH
        INT 21H
        MAIN ENDP
        PRINT PROC
        ; Initialize count
        MOV CX, 0
        MOV DX, 0
label1:
        ; If AX is zero, jump to print1
        CMP AX, 0
        JE print1
        ; Initialize BX to 16
        MOV BX, 16
        ; Divide AX by BX to convert it to hexadecimal
        DIV BX
        ; Push the remainder (DX) in the stack
        PUSH DX
        ; Increment the count
        INC CX
        ;Set DX to 0
        XOR DX, DX
        JMP label1
print1:
        ; Check if count is greater than zero
        CMP CX, 0
        JE exit
```

```
        ; Pop the top of the stack into DX
        POP DX
        ; Compare the value with 9
        CMP DX, 9
        JLE continue
        ; If value is greater than 9, add 7 so that after adding 48 it
        represents A
        ; For example, 10 + 7 + 48 = 65 which is ASCII value of A
        ADD DX, 7
continue:
        ; Add 48 so that it represents the ASCII value of digits
        ADD DX, 48
        ; Interrupt to print a character
        MOV AH, 02H
        INT 21H
        ; Decrease the count
        DEC CX
        JMP print1
exit:
        RET
PRINT ENDP
END MAIN
```



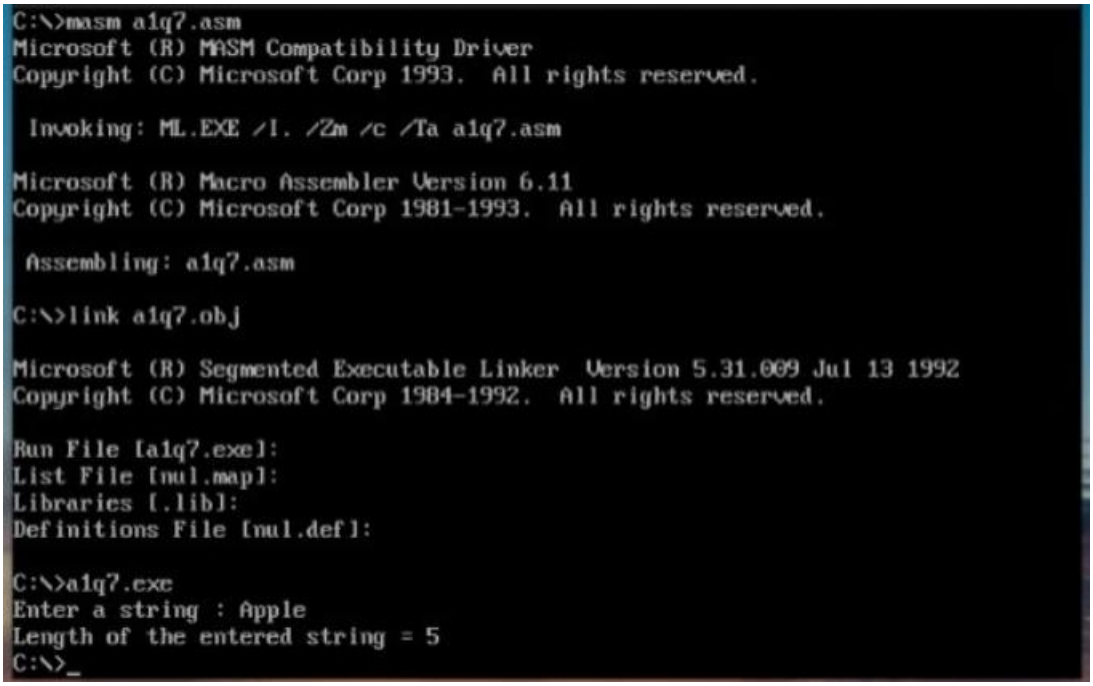
Q6. Now modify the program in Q2 so that it will ask for your name and takes the input from keyboard.

```
.model small
.stack 100h
.data
        arr db 20 DUP('$')
        msg db 'Enter your name : $'
.code
main proc
        mov ax,@data
        mov ds,ax
        mov ax,offset msg
        mov dx,ax
        mov ah,9
        int 21h
        mov si, offset arr
lp:
        mov ah,1
        int 21h
        cmp al,13
        je progend
        mov [si],al
        inc si
        jmp lp
progend :
        mov dx, offset arr
        mov ah,9
        int 21h
        mov ah,4ch
        int 21h
main endp
```



Q7. Write an Assembly Language Program to check the length of a given string.

```
.model small
.stack 100h
.data
        arr db 20 DUP('$')
        msg db 'Enter a string : $'
        msgg db 'Length of the entered string = $'
.code
main proc
        mov ax,@data
        mov ds,ax
        mov ax,offset msg
        mov dx,ax
        mov ah,9
        Int 21h
        mov cx,0
        mov si, offset arr
lp:
        mov ah,1
        int 21h
        cmp al,13
        je progend
        mov [si],al
        inc cx
        inc si
        jmp lp
progend:
        mov dx, offset msgg
        mov ah,9
        int 21h
        mov bx, cx ; Store the length in BX register for later conversion
        add bx, 48 ; Convert length to ASCII
        mov dx, bx ; Load DX with the ASCII value of length
        mov ah, 2
        int 21h
        mov ah,4ch
        int 21h
main endp
end mai
```


C:\>masm a1q7.asm
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993. All rights reserved.
Invoking: ML.EXE /I. /Zm /c /Ta a1q7.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.
Assembling: a1q7.asm
C:\>link a1q7.obj
Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.
Run File [a1q7.exe]:
List File [nul.map]:
Libraries [.lib]:
Definitions File [nul.def]:
C:\>a1q7.exe
Enter a string : Apple
Length of the entered string = 5
C:\>_