BEGUM ROKEYA UNIVERSITY, RANGPUR

Department of Computer Science & Engineering

Course Title: Microprocessor and Assembly Language

Course Code: CSE 3205

Assignment On: Lab Report-01

Submitted By,	**Submitted To,**
Mst. Arafatun Jannat Tania ID: 1605040 3rd Year 2st Semester Session: 2016-17 Department of Computer Science and Engineering	Marzia Sultana Lecturer Department Of Computer Science & Engineering. Begum Rokeya University, Rangpur

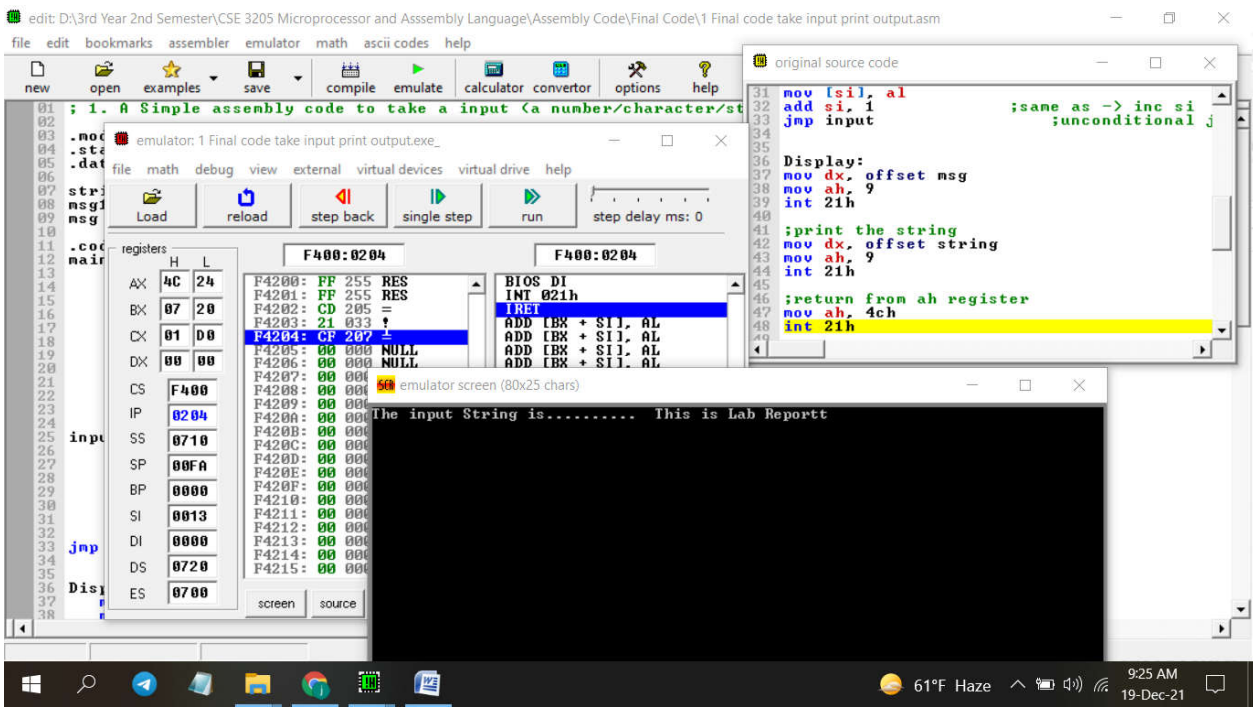
Submission Date: 19 December, 2021

; 1. A Simple assembly code to take a input (a number/character/string) from keyboard and print the input.

```
.model small              ;model directory <-- Specifies the total amount of memory the program would take
.stack 100h               ;stack segment directory <-- Specifies the storage for stack
.data                     ;data segment directory  <-- variables are defined here

string db 100 dup('$')
msg1   db 'Give the input String $'
msg    db 'The input String is.....  $'

.code                     ;code segment directory
main proc

   mov bx, @data
   mov ds, bx             ;initialize heap memory
   ;or .startup  it will load the data into DS memory

   ;print the msg1
```

```
    mov dx, offset msg1
    mov ah, 9
    int 21h

    lea si, string        ;same as -> mov si, offset string

input:
    mov ah, 1
    int 21h
    cmp al, 13            ;13 is the ASCII value of enter key

    je Display            ;conditional jump
    mov [si], al
    add si, 1             ;same as -> inc si
jmp input                 ;unconditional jump


Display:
   mov dx, offset msg
   mov ah, 9
   int 21h
```

```
    ;print the string
    mov dx, offset string
    mov ah, 9
    int 21h

    ;return from ah register
    mov ah, 4ch
    int 21h

main endp
end main
```



; 2.Perform addition of two numbers(2 Digit Number)

.model small ;model directory <-- Specifies the total amount of memory the program would take

.stack 100h ;stack segment directory <-- Specifies the storage for stack

.data ;data segment directory <-- variables are defined here

sum dw 0

```
    cnt     dw 3
    msg1    dw ' Enter First Number( 2 Digit ).... $'
    msg2    dw 13, 10, 'Enter Second Number( 2 Digit ).... $'

    endl    dw 13, 10, 13, 10, 'The sum is .... $'
    counter dw 0
    base    dw 10

.code                   ;code segment directory
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h

        mov cx, 0       ;so that cx contain a proper value
takingInput:
```

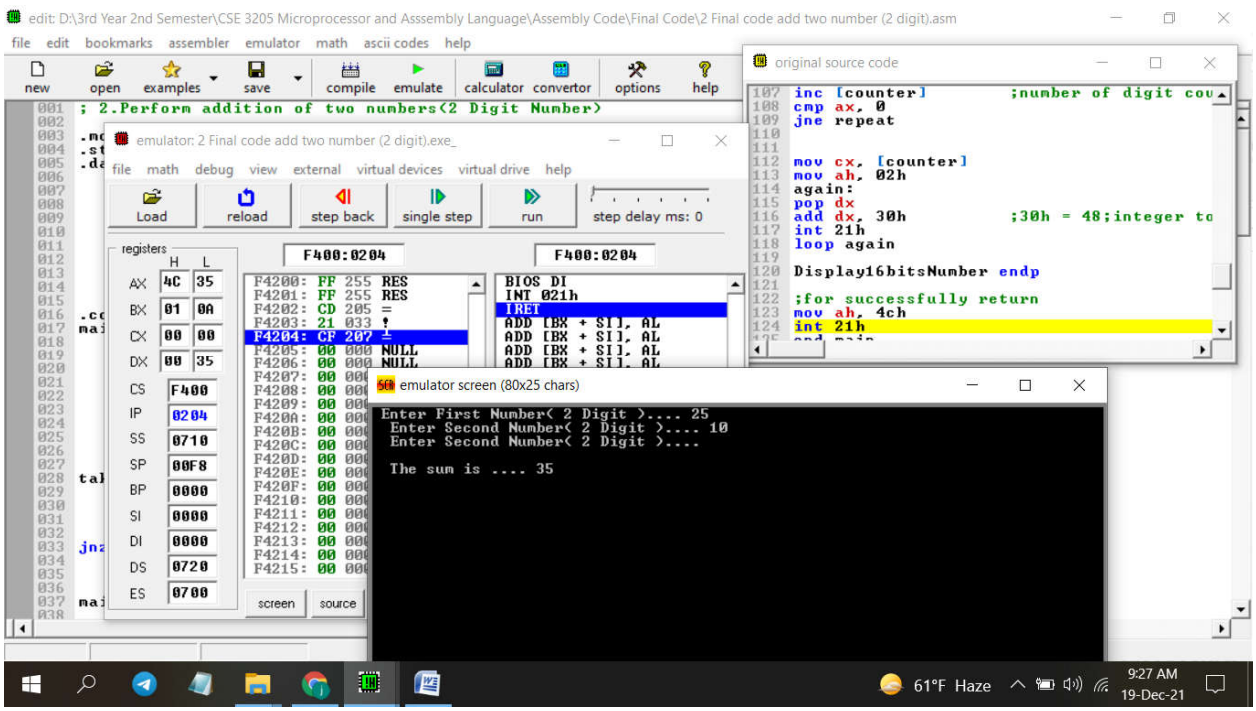
```
        add sum, cx
        dec cnt
        cmp cnt, 0
jnz input2DigitNumber

       call Display16bitsNumber

main endp

;************************************** PROCEDURE
**************************************

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

input2DigitNumber proc

        ;taking the first digit
        mov ah, 01h
        int 21h
        sub al, 30h      ; 30h = 48 ;ASCII to integer
```

```
mov bh, al       ; bx = bh bl

;taking the second digit
mov ah, 01h
int 21h
sub al, 30h



mov ch, al       ;store the second digit
mov al, bh       ;for multi
mov bl, 10       ;bx = bh bl
                 ;= 05 10
mul bl           ;8 bits multiplication
                 ;ax = al * 8-bits reg

add al, ch       ;al containx num * 10
                 ;al = al + ch
                 ;   = 50 + 2
                 ;   = 52

mov cx, ax       ;bcz in 71 line ax will corrupt
```

```
        ;print the msg2
        mov dx, offset msg2
        mov ah, 09h
        int 21h

jmp takingInput

input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

        ;print the endl
        mov dx, offset endl
        mov ah, 9
        int 21h


        mov ax, sum
```

```
    cmp ax, 0           ;ax < 0
    jge repeat          ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax             ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
    mov ah, 02h
    int 21h


    pop ax
    neg ax              ;again 2's compliment so that we can get the proper
value


repeat:
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ;number of digit count
    cmp ax, 0
```

```
jne repeat



    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
    add dx, 30h          ;30h = 48;integer to ASCII; character
    int 21h
loop again

Display16bitsNumber endp

    ;for successfully return
    mov ah, 4ch
    int 21h
end main
```



; 3.Perform subtraction of two numbers(2 Digit Number)

.model small ;model directory <-- Specifies the total amount of memory the program would take

.stack 100h ;stack segment directory <-- Specifies the storage for stack

.data ;data segment directory

sum dw 0

cnt dw 3

msg1 dw 'Enter First Number(2 Digit)..... $'

```
    msg2    dw 13, 10, 'Enter Second Number( 2 Digit ).... $'

    endl    dw 13, 10, 13, 10, 'The subtraction is (First - Second) .... $'
    counter dw 0
    base    dw 10

.code                   ;code segment directory
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h

        mov cx, 0       ;so that cx contain a proper value
takingInput:

        ;add sum, ax
```

```
        cmp cnt, 1
        je negativeValue

        add sum, cx
        dec cnt
        cmp cnt, 0
jnz input2DigitNumber

      negativeValue:
            sub sum, cx

      call Display16bitsNumber

main endp

;************************************** PROCEDURE
**************************************

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

input2DigitNumber proc
```

```
;taking the first digit
mov ah, 01h
int 21h
sub al, 30h      ; 30h = 48 ;ASCII to integer
mov bh, al       ; bx = bh bl

;taking the second digit
mov ah, 01h
int 21h
sub al, 30h


mov ch, al       ;store the second digit
mov al, bh       ;for multi
mov bl, 10       ;bx = bh bl
                 ;= 05 10
mul bl           ;8 bits multiplication
                ;ax = al * 8-bits reg

add al, ch       ;al containx num * 10
```

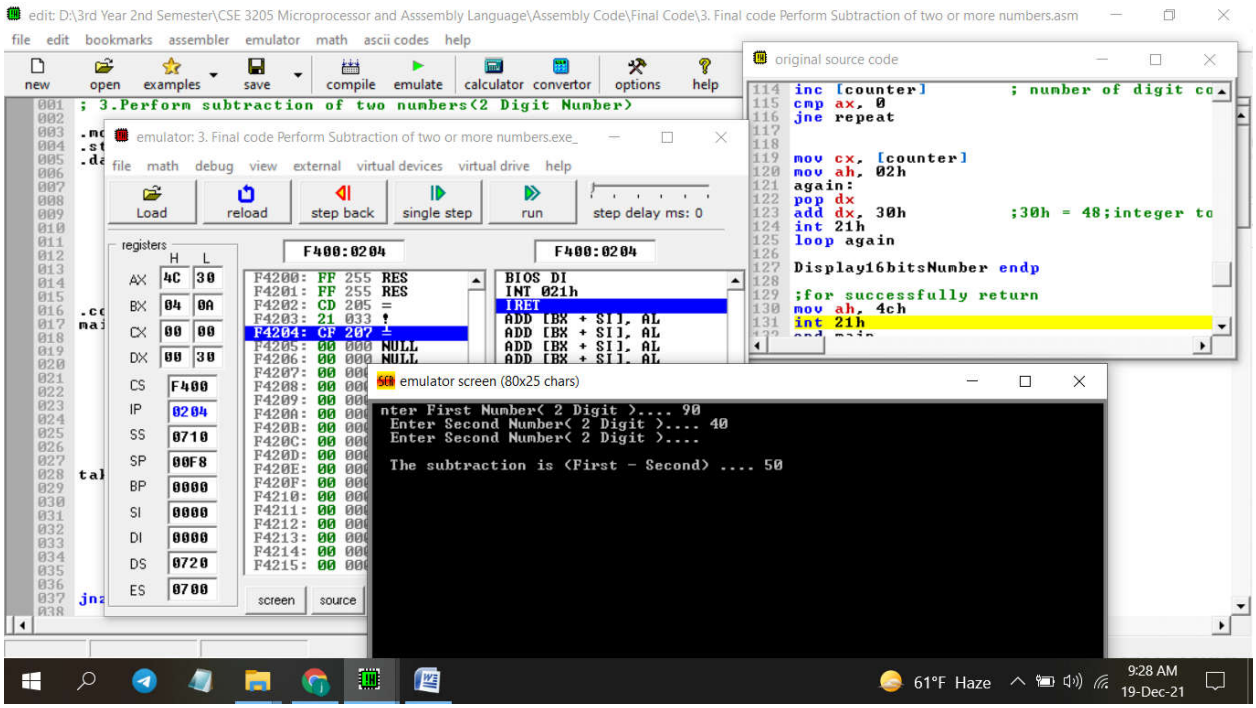
```
                ;al = al + ch
                ;   = 50 + 2
                ;   = 52

        mov cx, ax      ;bcz in 78 line ax will corrupt
        ;print the msg2
        mov dx, offset msg2
        mov ah, 09h
        int 21h

jmp takingInput

input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

        ;print the endl
        mov dx, offset endl
        mov ah, 9
```

```
    int 21h



    mov ax, sum


    cmp ax, 0          ;ax < 0
    jge repeat         ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax            ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
    mov ah, 02h
    int 21h


    pop ax
    neg ax             ;again 2's compliment so that we can get the proper
value


repeat:
```

```
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ; number of digit count
    cmp ax, 0
jne repeat



    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
    add dx, 30h         ;30h = 48;integer to ASCII; character
    int 21h
loop again

Display16bitsNumber endp

    ;for successfully return
    mov ah, 4ch
```

int 21h

end main



;4. Case Conversion of a character/ a string

.model small ;model directory <-- Specifies the total amount of memory the program would take

.stack 100h ;stack segment directory <-- Specifies the storage for stack

.data ;data segment directory

```
string db 100 dup('$')
msg1   db 'Give the input String  .....  $'
msg2   db 13, 10, 'The Output is          .....  $'

.code                   ;code segment directory
conversion proc

        mov bx, @data
        mov ds, bx
        ;or .startup

        ;print the msg1
        mov dx, offset msg1
        mov ah, 9
        int 21h


        mov si, offset string
 takeInput:
        mov ah, 1
        int 21h
```

```
        cmp al, 13          ;13 is the ASCII value of enter key
        je Display          ;conditional jump

        cmp al, 32          ;32 is the ASCII value of space
        je space:

        cmp al, 96          ;96 is the ASCII value of (a)
        jg upper:
        jl lower:
jmp takeInput               ;unconditional jump

    space:
        mov [si], al
        inc si
jmp takeInput

    upper:
        sub al, 32
        mov [si], al
        inc si
jmp takeInput
```

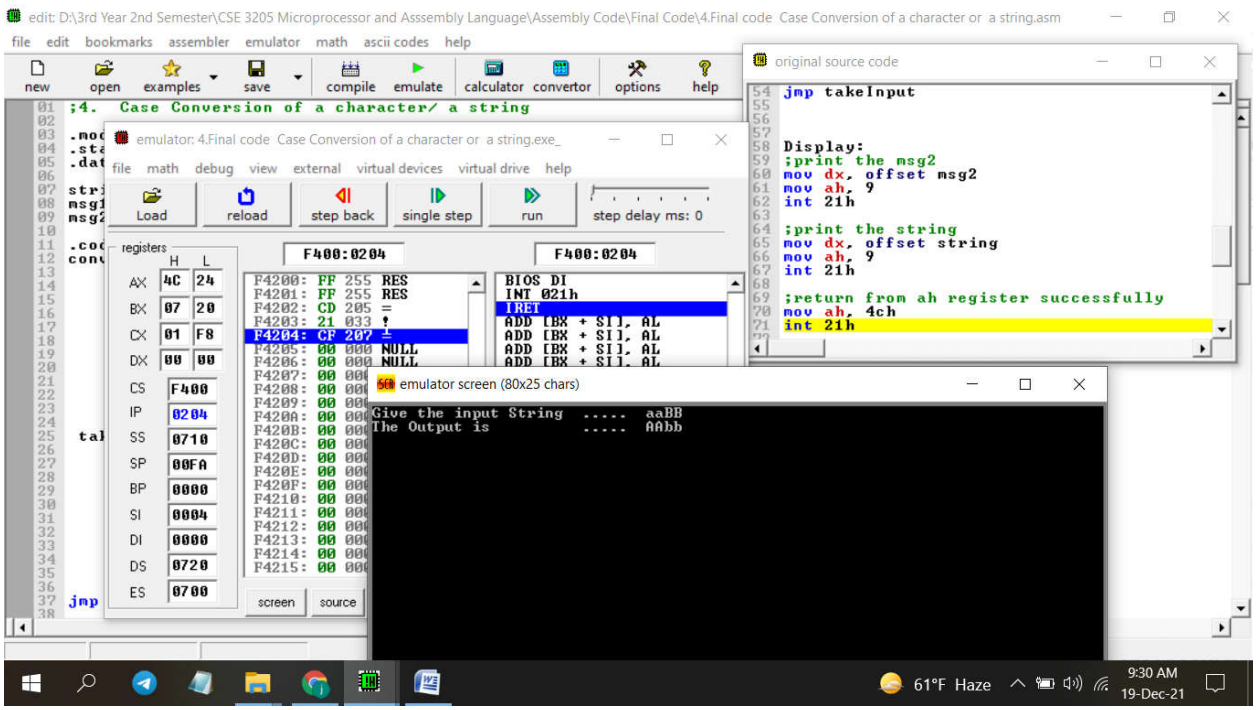
```
    lower:
        add al, 32
        mov [si], al
        inc si
jmp takeInput



    Display:
        ;print the msg2
        mov dx, offset msg2
        mov ah, 9
        int 21h

        ;print the string
        mov dx, offset string
        mov ah, 9
        int 21h

        ;return from ah register successfully
```

mov ah, 4ch

int 21h

conversion endp

end conversion



; 6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the

;following high-level language assignment statements Into assembly language.

;A, B, and C are word variables.

;a. A = B - A

;Assuming A and B are 2 Digit Numbers

```
.model small              ;model directory <-- Specifies the total amount of memory the program would take
.stack 100h               ;stack segment directory <-- Specifies the storage for stack
.data                     ;data segment directory  <-- variables are defined here

    A       dw 0
    B       dw 0
    cnt     dw 3
    msg1    dw 'Enter A ( 2 Digit ).... $'
    msg2    dw 13, 10, 'Enter B ( 2 Digit ).... $'

    endl    dw 13, 10, 13, 10, 'The result of A = B - A is .... $'
    counter dw 0
    base    dw 10
```

```
.code                   ;code segment directory
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h


        mov cx, 0       ;so that cx contain a proper value
takingInput:

        cmp cnt, 1
        je negative

        add A, cx
        dec cnt
```

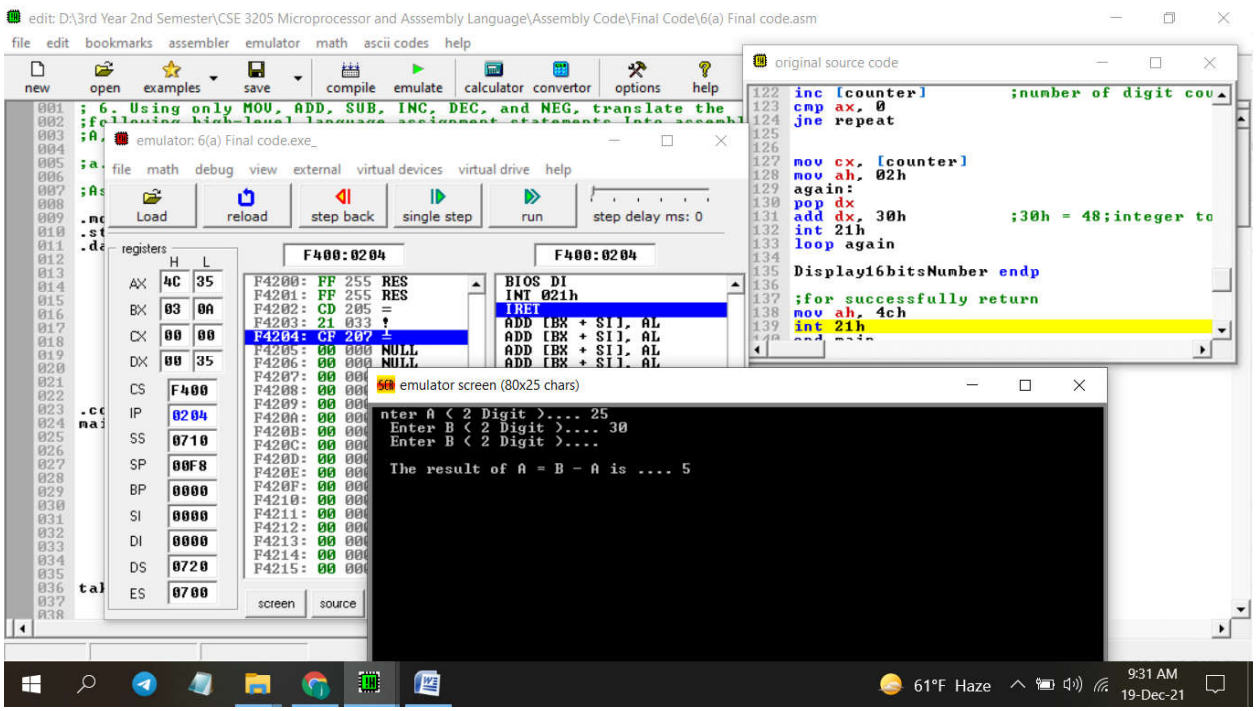
```
        cmp cnt, 0
jnz input2DigitNumber

        negative:
            sub cx, A  ;cx = B, sum = A so B = B - A
            mov A, cx  ;bcz in display cx will override

        call Display16bitsNumber

main endp

;************************************** PROCEDURE
**************************************

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

input2DigitNumber proc

        ;taking the first digit
        mov ah, 01h
        int 21h
```

```
sub al, 30h      ; 30h = 48 ;ASCII to integer
mov bh, al       ; bx = bh bl

;taking the second digit
mov ah, 01h
int 21h
sub al, 30h


mov ch, al       ;store the second digit
mov al, bh       ;for multi
mov bl, 10       ;bx = bh bl
                 ;= 05 10
mul bl           ;8 bits multiplication
                ;ax = al * 8-bits reg

add al, ch       ;al containx num * 10
                ;al = al + ch
                ;   = 50 + 2
                ;   = 52
```

```
        mov cx, ax
        ;print the msg2
        mov dx, offset msg2
        mov ah, 09h
        int 21h

jmp takingInput

input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

        ;print the endl
        mov dx, offset endl
        mov ah, 9
        int 21h


        mov ax, A
```

```
    cmp ax, 0           ;ax < 0
    jge repeat          ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax             ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
    mov ah, 02h
    int 21h


    pop ax
    neg ax              ;again 2's compliment so that we can get the proper
value


repeat:
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ;number of digit count
```

```
    cmp ax, 0
jne repeat



    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
    add dx, 30h         ;30h = 48;integer to ASCII; character
    int 21h
loop again

Display16bitsNumber endp

    ;for successfully return
    mov ah, 4ch
    int 21h
end main
```



; 6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the

;following high-level language assignment statements Into assembly language.

;A, B, and C are word variables.

;b. A = -(A+ 1)

;Assuming A is a 2 Digit Number

.model small

```
.stack 100h
.data

    A       dw 0
    cnt     dw 3
    msg1    dw ' Give the value of A (2 Digit) ... $'

    endl    dw 13, 10, 13, 10, 'The result of A = -(A + 1) is .... $'
    counter dw 0
    base    dw 10

.code
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h
```

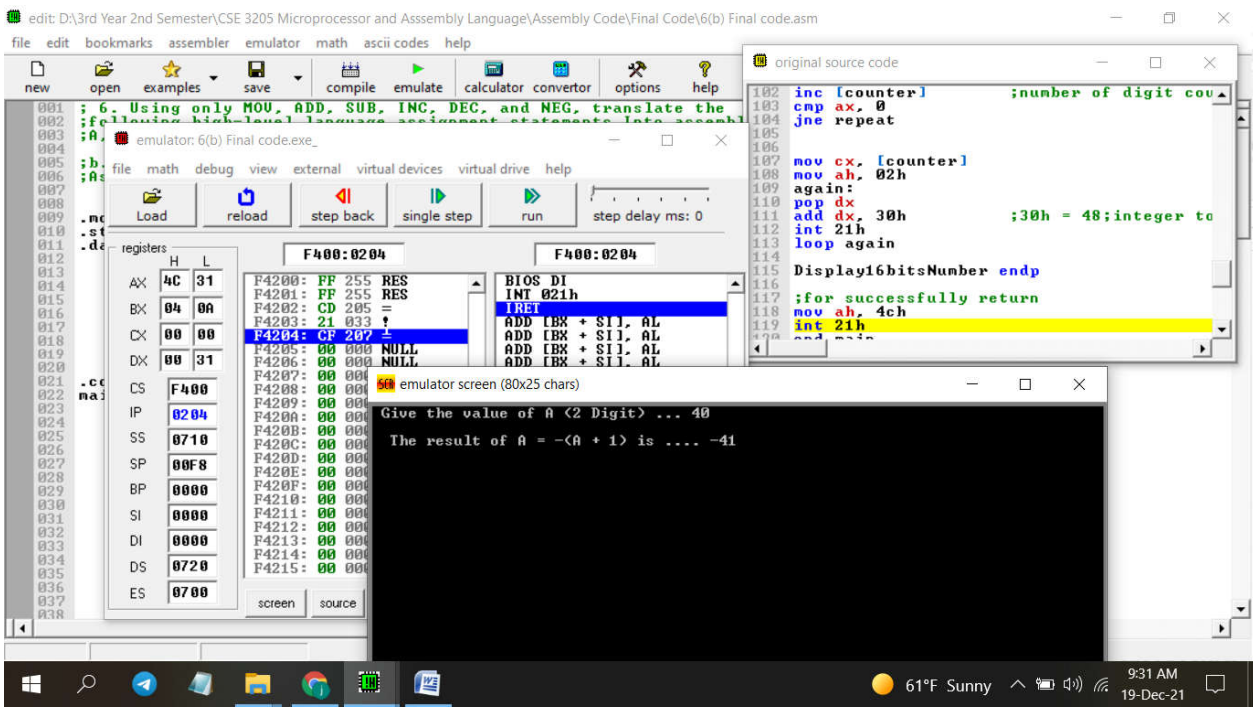
```
        mov ax, 0       ;remove @data address
        call input2DigitNumber
        add ax, 1
        neg ax
        mov A, ax       ;bcz in display ax will override

        call Display16bitsNumber

main endp

;************************************** PROCEDURE
**************************************

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

input2DigitNumber proc

        ;taking the first digit
        mov ah, 01h
        int 21h
```

```
sub al, 30h      ; 30h = 48 ;ASCII to integer
mov bh, al       ; bx = bh bl

;taking the second digit
mov ah, 01h
int 21h
sub al, 30h



mov ch, al       ;store the second digit
mov al, bh       ;for multi
mov bl, 10       ;bx = bh bl
                   ;= 05 10
mul bl           ;8 bits multiplication
                 ;ax = al * 8-bits reg

add al, ch       ;al containx num * 10
                 ;al = al + ch
                 ;   = 50 + 2
                 ;   = 52
ret              ;return the line 34
```

```
input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

    ;print the endl
    mov dx, offset endl
    mov ah, 9
    int 21h


    mov ax, A


    cmp ax, 0          ;ax < 0
    jge repeat         ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans


    ;if negative
    push ax            ;mov ah, 02h e value change hoye jabe
```

```
    mov dl, '-'
    mov ah, 02h
    int 21h

    pop ax
    neg ax              ;again 2's compliment so that we can get the proper
value

repeat:
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ;number of digit count
    cmp ax, 0
jne repeat

    mov cx, [counter]
    mov ah, 02h
again:
```

```
        pop dx
        add dx, 30h          ;30h = 48;integer to ASCII; character
        int 21h
loop again

Display16bitsNumber endp

        ;for successfully return
        mov ah, 4ch
        int 21h
end main
```



; 6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the

;following high-level language assignment statements Into assembly language.

;A, B, and C are word variables.

;c. C = A + B

;Assuming A and B are 2 Digit Numbers

```
.model small
.stack 100h
.data

    sum     dw 0
    cnt     dw 3

    endl    dw 13, 10, 13, 10, 'The sum of C = A + B is .... $'
    msg1    dw ' Give the value of A and B (2 Digit) ... $'
    counter dw 0
    base    dw 10
```

```
.code
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h

        mov ax, 0  ;for remove the 09h
takingInput:

        add sum, ax
        dec cnt
        cmp cnt, 0
jnz input2DigitNumber

       call Display16bitsNumber
```

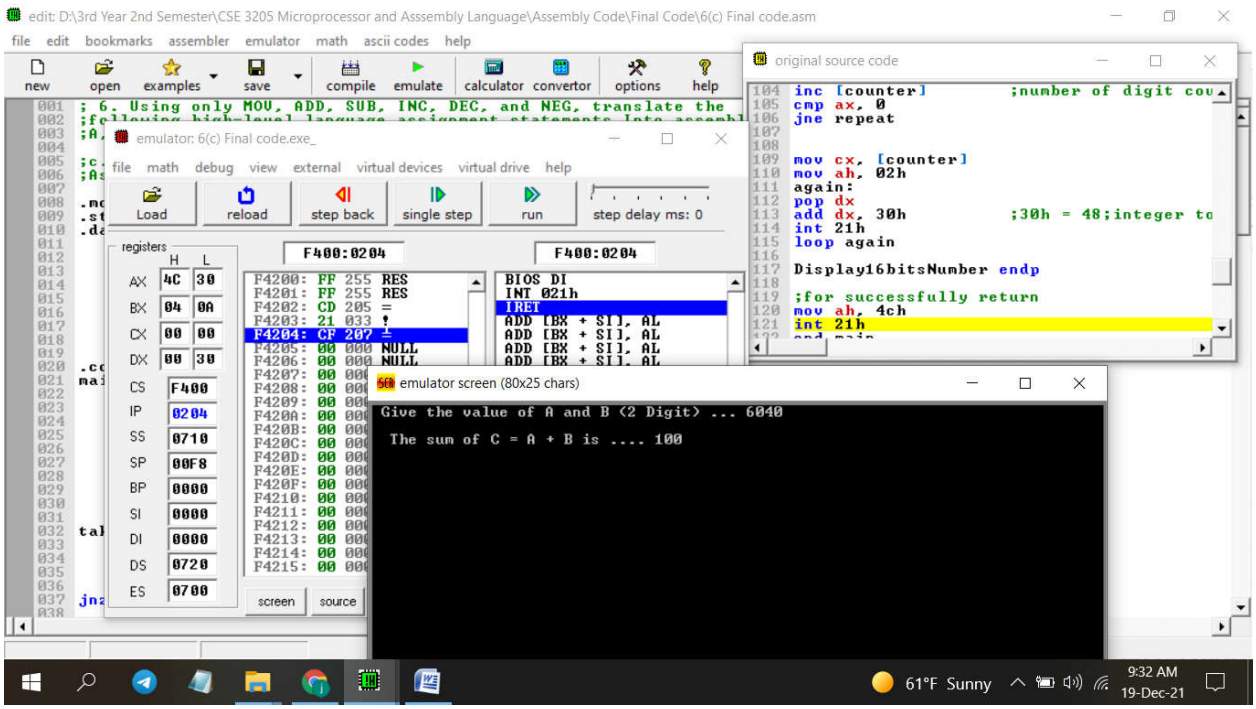
```
main endp

;****************************************** PROCEDURE
******************************************

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


input2DigitNumber proc

    ;taking the first digit
    mov ah, 01h
    int 21h
    sub al, 30h      ; 30h = 48 ;ASCII to integer
    mov bh, al       ; bx = bh bl

    ;taking the second digit
    mov ah, 01h
    int 21h
    sub al, 30h
```

```
        mov ch, al       ;store the second digit
        mov al, bh       ;for multi
        mov bl, 10       ;bx = bh bl
                         ;= 05 10
        mul bl           ;8 bits multiplication
                         ;ax = al * 8-bits reg

        add al, ch       ;al containx num * 10
                         ;al = al + ch
                         ;   = 50 + 2
                         ;   = 52
jmp takingInput

input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

      ;print the endl
```

```
    mov dx, offset endl
    mov ah, 9
    int 21h



    mov ax, sum


    cmp ax, 0          ;ax < 0
    jge repeat         ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax            ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
    mov ah, 02h
    int 21h


    pop ax
    neg ax             ;again 2's compliment so that we can get the proper
value
```

```
repeat:
    mov dx, 0          ; dx = dividend high (To avoid divide overflow
error)
    div base           ; ax = Quotient, dx = remainder
    push dx            ; push e always 16 bit dite hoy
    inc [counter]      ;number of digit count
    cmp ax, 0
jne repeat



    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
    add dx, 30h        ;30h = 48;integer to ASCII; character
    int 21h
loop again

Display16bitsNumber endp
```

```
    ;for successfully return
    mov ah, 4ch
    int 21h
end main
```



; 6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the

;following high-level language assignment statements Into assembly language.

;A, B, and C are word variables.

;d. B = 3 * B + 7

;Assuming B is a 2 Digit Number

```
.model small
.stack 100h
.data

    B       dw 0
    cnt     dw 3
    msg1    dw ' Give the value of B (2 Digit) ... $'

    endl    dw 13, 10, 13, 10, 'The result of B = 3 * B + 7 is .... $'
    counter dw 0
    base    dw 10

.code
main proc

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

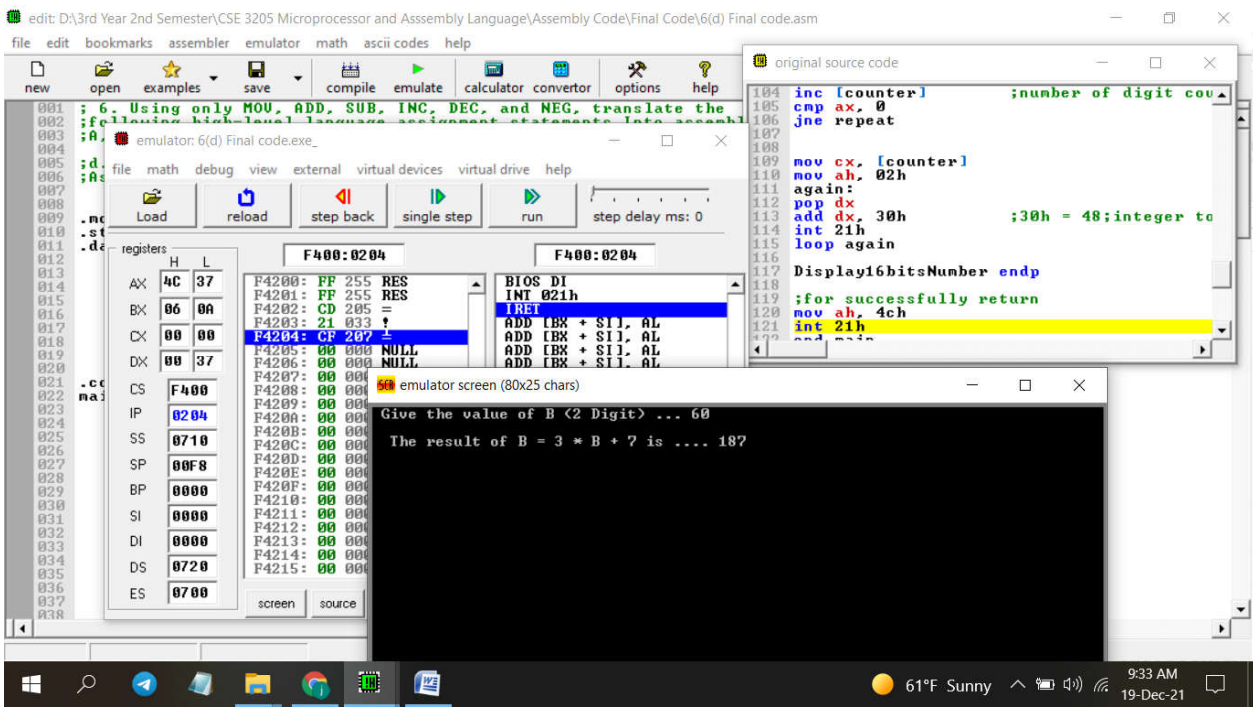
```
        ;print the msg1
        mov dx, offset msg1
        mov ah, 09h
        int 21h

        mov ax, 0       ;remove @data address
        call input2DigitNumber
        mov B, ax
        add ax, B
        add ax, B
        add ax, 7
        mov B, ax       ;bcz in display ax will override

        call Display16bitsNumber

main endp

;****************************************** PROCEDURE
******************************************

;******************************************
******************************************
```

```
input2DigitNumber proc

        ;taking the first digit
        mov ah, 01h
        int 21h
        sub al, 30h      ; 30h = 48 ;ASCII to integer
        mov bh, al       ; bx = bh bl

        ;taking the second digit
        mov ah, 01h
        int 21h
        sub al, 30h



        mov ch, al       ;store the second digit
        mov al, bh       ;for multi
        mov bl, 10       ;bx = bh bl
                         ;= 05 10
        mul bl           ;8 bits multiplication
                        ;ax = al * 8-bits reg
```

```
    add al, ch       ;al containx num * 10
                     ;al = al + ch
                     ;   = 50 + 2
                     ;   = 52
    ret              ;return the line 34
input2DigitNumber endp


;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

    ;print the endl
    mov dx, offset endl
    mov ah, 9
    int 21h


    mov ax, B

    cmp ax, 0            ;ax < 0
```

```
    jge repeat          ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax             ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
    mov ah, 02h
    int 21h


    pop ax
    neg ax              ;again 2's compliment so that we can get the proper
value


repeat:
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ;number of digit count
    cmp ax, 0
jne repeat
```

```
    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
    add dx, 30h          ;30h = 48;integer to ASCII; character
    int 21h
loop again

Display16bitsNumber endp

    ;for successfully return
    mov ah, 4ch
    int 21h
end main
```



; 6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the

;following high-level language assignment statements Into assembly language.

;A, B, and C are word variables.

;e. A = B - A - 1

;Assuming A and B are 2 Digit Numbers

.model small

```
.stack 100h
.data

    A       dw 0
    B       dw 0
    cnt     dw 3
    msg1    dw ' Give the value of A (2 Digit) ... $'
    msg2    dw 13, 10, 'Give the value of B (2 Digit) ... $'

    endl    dw 13, 10, 13, 10, 'The result of A = B - A - 1 is .... $'
    counter dw 0
    base    dw 10

.code
main proc

        mov ax, @data
        mov ds, ax

        ;print the msg1
        mov dx, offset msg1
```

```
        mov ah, 09h
        int 21h

        mov cx, 0       ;for remove garbage value
takingInput:

        cmp cnt, 1
        je inputB

        add A, cx
        dec cnt
        cmp cnt, 0
jnz input2DigitNumber

        inputB:
            sub cx, A  ;cx = B, sum = A so B = B - A
            mov A, cx  ;bcz in display I will use A
            dec A

        call Display16bitsNumber
```

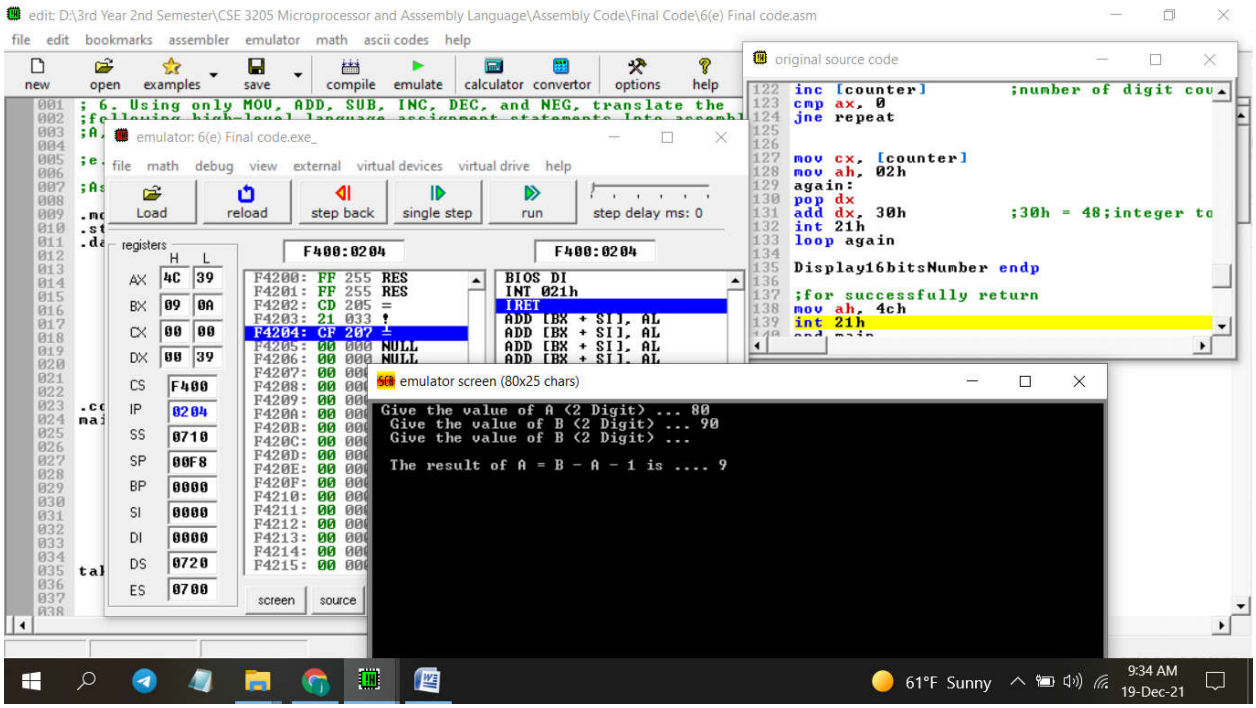
```
main endp

;****************************************** PROCEDURE
******************************************

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


input2DigitNumber proc


        ;taking the first digit
        mov ah, 01h
        int 21h
        sub al, 30h      ; 30h = 48 ;ASCII to integer
        mov bh, al       ; bx = bh bl


        ;taking the second digit
        mov ah, 01h
        int 21h
        sub al, 30h
```

```
        mov ch, al       ;store the second digit
        mov al, bh       ;for multi
        mov bl, 10       ;bx = bh bl
                         ;= 05 10
        mul bl           ;8 bits multiplication
                         ;ax = al * 8-bits reg

        add al, ch       ;al containx num * 10
                         ;al = al + ch
                         ;   = 50 + 2
                         ;   = 52

        mov cx, ax
        ;print the msg2
        mov dx, offset msg2
        mov ah, 09h
        int 21h

jmp takingInput

input2DigitNumber endp
```

```
;Printing 16 bit number using stack in 8086 Assembly language
Display16bitsNumber proc

    ;print the endl
    mov dx, offset endl
    mov ah, 9
    int 21h



    mov ax, A


    cmp ax, 0          ;ax < 0
    jge repeat         ;if ax >= 0 ;for jg 0 result will -0 that is not right
ans



    ;if negative
    push ax            ;mov ah, 02h e value change hoye jabe
    mov dl, '-'
```

```
    mov ah, 02h
    int 21h

    pop ax
    neg ax              ;again 2's compliment so that we can get the proper
value

repeat:
    mov dx, 0           ; dx = dividend high (To avoid divide overflow
error)
    div base            ; ax = Quotient, dx = remainder
    push dx             ; push e always 16 bit dite hoy
    inc [counter]       ;number of digit count
    cmp ax, 0
jne repeat

    mov cx, [counter]
    mov ah, 02h
again:
    pop dx
```

```
        add dx, 30h          ;30h = 48;integer to ASCII; character
        int 21h
loop again

Display16bitsNumber endp

        ;for successfully return
        mov ah, 4ch
        int 21h
end main
```

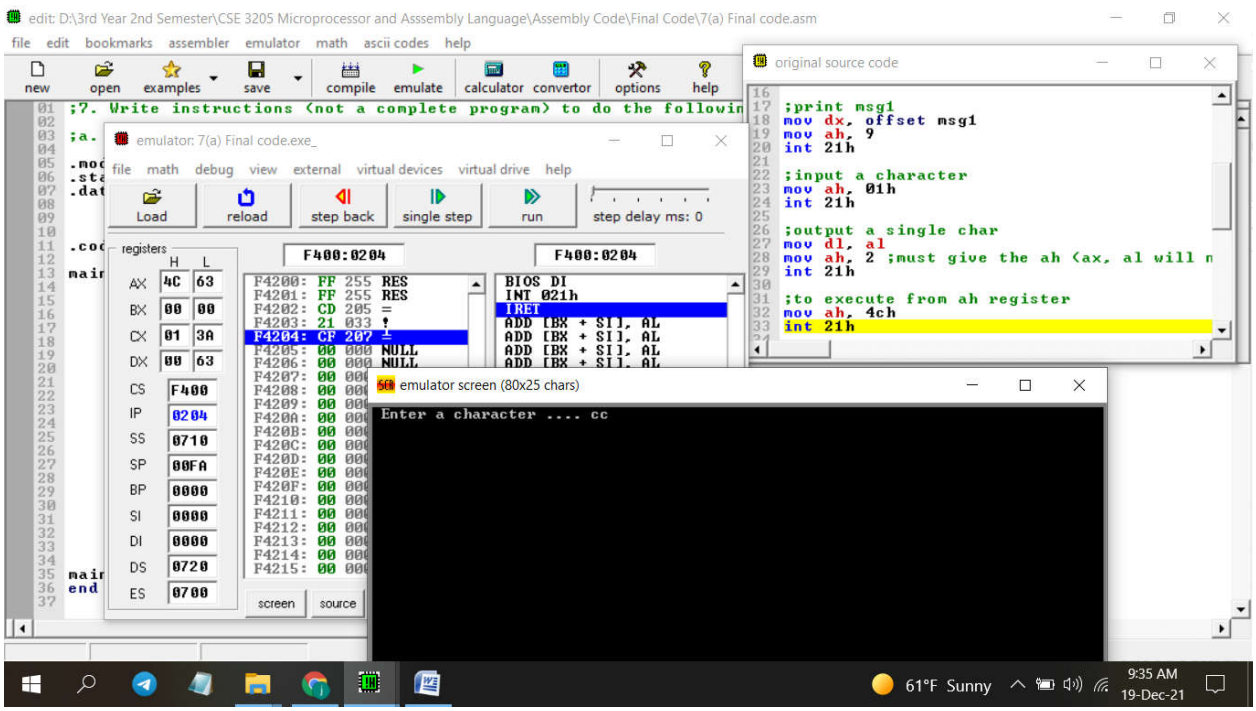


;7. Write instructions (not a complete program) to do the following.

;a. Read a character, and display it at the next position on the same line

```
.model small
.stack 100h
.data

    msg1  dw  ' Enter a character .... $'

.code

main proc
    mov ax, @data
    mov ds, ax

    ;print msg1
    mov dx, offset msg1
```

```
    mov ah, 9
    int 21h

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

    ;output a single char
    mov dl, al
    mov ah, 2 ;must give the ah (ax, al will not work)
    int 21h

    ;to execute from ah register
    mov ah, 4ch
    int 21h

main endp
end main
```



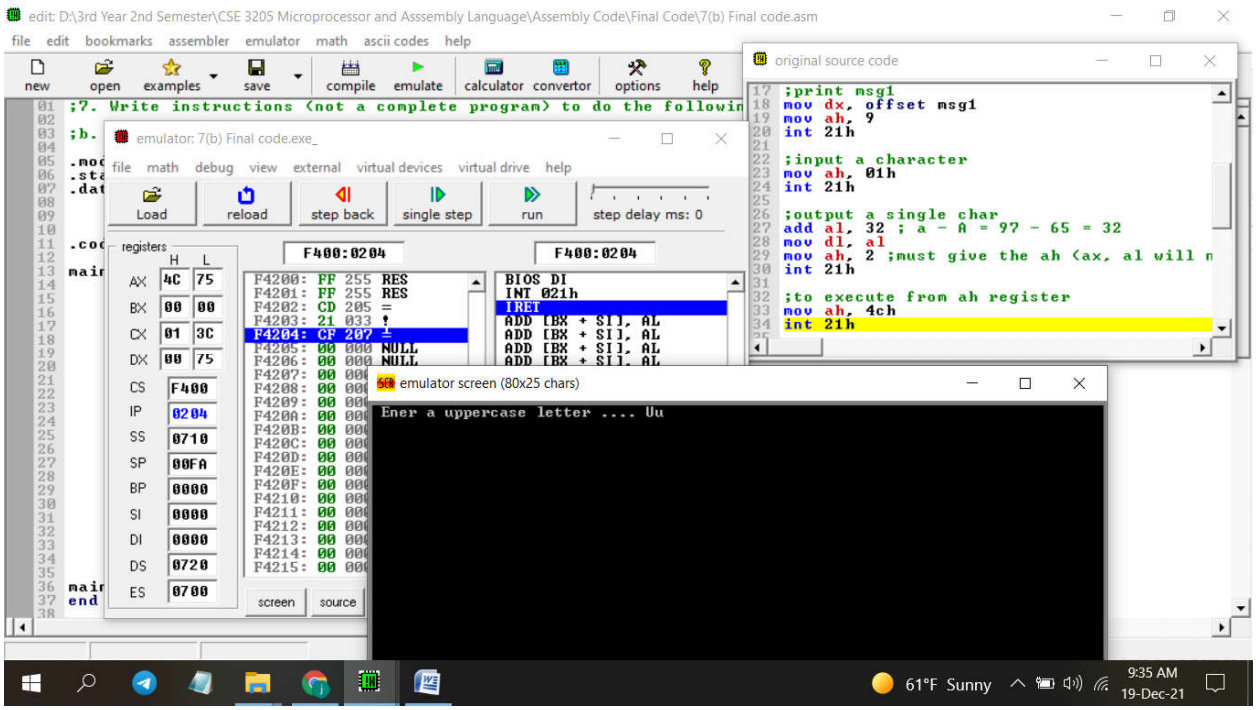
;7. Write instructions (not a complete program) to do the following.

;b. Read an uppercase letter (omit error checking), and display it at the next position on the same line in lower case.

```
.model small

.stack 100h

.data


    msg1  dw  ' Ener a uppercase letter .... $'
```

```
.code

main proc
    mov ax, @data
    mov ds, ax

    ;print msg1
    mov dx, offset msg1
    mov ah, 9
    int 21h

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

    ;output a single char
    add al, 32 ; a - A = 97 - 65 = 32
    mov dl, al
    mov ah, 2 ;must give the ah (ax, al will not work)
    int 21h
```

```
    ;to execute from ah register
    mov ah, 4ch
    int 21h

main endp
end main
```



;8.Write a program to (a) display a "?", (b) read two decimal digits

;whose sum "is less than 10, (c) display them and their sum on the

;next line, with an appropriate message.

;Sample execution:

;?27

;THE SUM OF 2 AND 7 IS 9

```
.model small
.stack 100h
.data

  msg1 dw ' ?$'
  msg2 dw 13, 10, 'THE SUM OF $'
  msg3 dw ' AND $'
  msg4 dw ' IS $'

  sum  db 0
```

```
.code

main proc
    ;for access data directly from Data Segment
    mov ax, @data
    mov ds, ax

    ;print the msg1
    mov dx, offset msg1    ;same as -- lea dx, msg1
    mov ah, 9
    int 21h

    ;********************* INPUT
*******************************
    ;input first digit
    mov ah, 1
    int 21h
    mov ch, al    ;cx = ch cl ;store the first digit
                  ;= 02
    add sum, al
```

```
    ;input second digit
    mov ah, 1
    int 21h
    mov cl, al    ;cx = ch cl ;store the second digit
                  ;= 02 07
    add sum, al

    ;********************* OUTPUT
*******************************
    ;print the msg2
    mov dx, offset msg2
    mov ah, 9
    int 21h

    ;print the first digit
    mov dl, ch
    mov ah, 2
    int 21h

    ;print the msg3
```

```
mov dx, offset msg3
mov ah, 9
int 21h

;print the second digit
mov dl, cl
mov ah, 2
int 21h

;print the msg4
mov dx, offset msg4
mov ah, 9
int 21h

;print the sum
sub sum, 48   ;maintain the ASCII value
mov dl, sum
mov ah, 2
int 21h

;to execute from ah register
```

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