BEGUM ROKEYA UNIVERSITY, RANGPUR

Department of Computer Science & Engineering

Course Title: Microprocessor and Assembly Language

Course Code: CSE 3206

Assignment On: Lab Report-02 On Assembly Codes

Submitted By,	**Submitted To,**

MD. Jubayer Hossen Jame ID: 1605006 Reg: 000008645 3rd Year 2st Semester Session: 2016-17 Department of Computer Science and Engineering	Marjia Sulttana Lecturer Department Of Computer Science & Engineering. Begum Rokeya University, Rangpur

Submission Date: 26 Februay, 2022

;1. Draw the following pattern (N.B. the length of the pyramid can be changed)

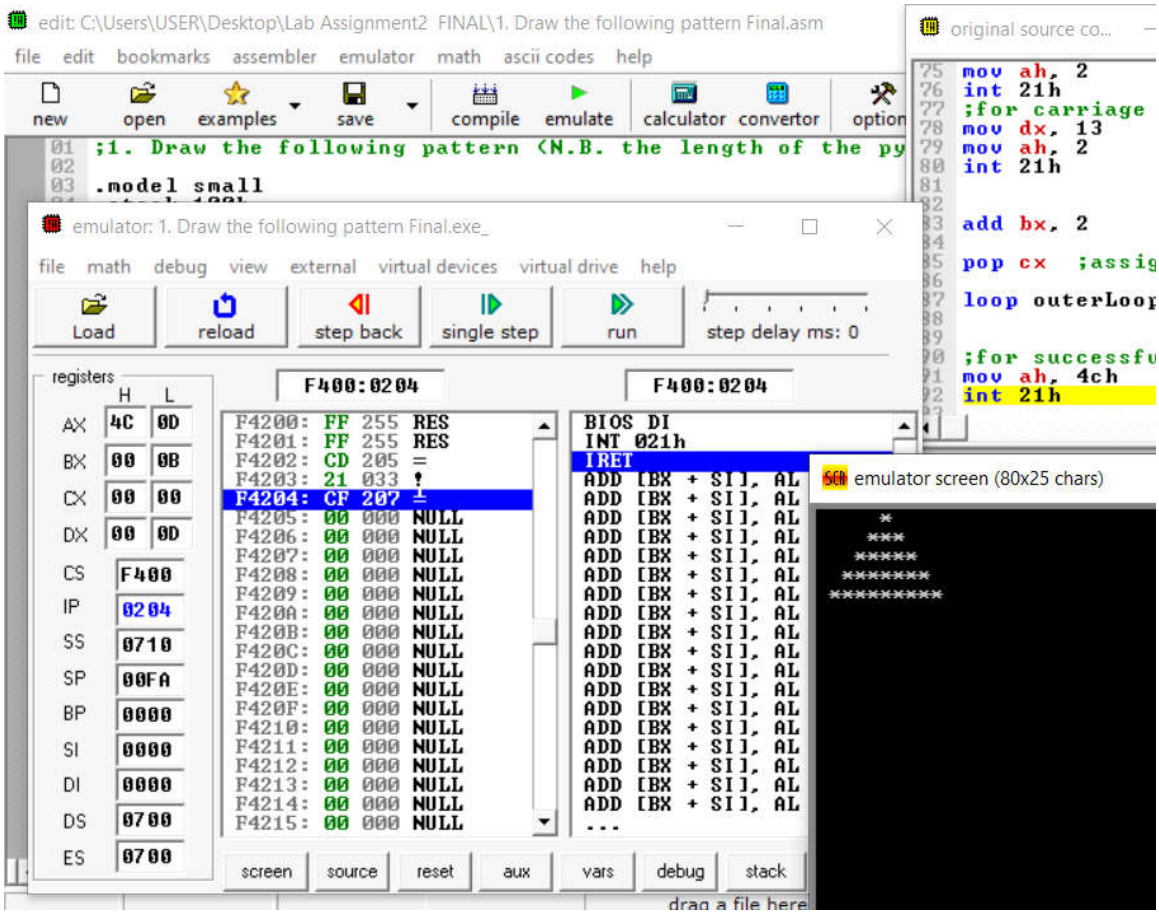
```
.model small
.stack 100h
```

```
.data
.code

main proc


   ;here we can input 2^8 - 1 = 255 (maximum)
;input number will store in bh register
mov bh, 0
mov bl, 10d


      INPUT:

      ;for input a single character
      mov ah, 1
      int 21h

      cmp al, 13d ;13d is the ASCII of enter key
      jne NUMBER

      jmp EXIT


      NUMBER:
      sub al, 30h ;zero ASCII 48d = 30h
```

```
    mov cl, al ;store the al value bcz after mul it will be corrupted
    mov al, bh

    mul bl     ;8 bits multiplication
            ;ax = al * 8-bits reg
    add al, cl
    mov bh, al

    JMP INPUT

EXIT:

     mov cx, 0 ;reset
     mov cl, bh

     mov bx, 0 ;reset
     mov bx, 1

outerLoop:

     push cx ;store the counter value

;for print the space
SPACE:
```

```
        mov dx, ' '
        mov ah, 2
        int 21h

loop SPACE

        mov cx, bx

        innerLoop:

                mov dx, '*'
                mov ah, 2
                int 21h

        loop innerLoop

        ;for new line
        mov dx, 10
        mov ah, 2
        int 21h
        ;for carriage return
        mov dx, 13
        mov ah, 2
        int 21h
```

```
        add bx, 2

        pop cx  ;assign counter of loop again

loop outerLoop


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

main endp
end main
```



;2. 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.

; i. A=B-A

; ii. A= - (A+1)

; iii. C=A+B

; iv. B=3*B+7

; v. A=B-A-1

```
;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         db 0
    B         db 0
    opsn      db 0
    finalRslt db 0

    cnt     dw 3
    msg1    dw 'Enter the value of A and B = $'
    msg2    dw 13, 10, 13, 10, 'Choose option (problem number): $'

    endl    dw 13, 10, 13, 10, ' The result of $'
    end2    dw ' no. problem 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

;for input A
call inputNumber
mov A, bh

;for input B
call inputNumber
mov B, bh


;for print msg2
mov dx, offset msg2
mov ah, 9h
int 21h

;taking the option number
mov ah, 01h
int 21h
mov opsn, al ;store the value
sub al, 48
```

```
;****************************************  OPTION
****************************************

;****************************************
****************************************
    cmp al, 5
    je  case5

    cmp al, 4
    je  case4

    cmp al, 3
    je  case3

    cmp al, 2
    je  case2

    case1:
    ; i. A=B-A

        mov ax, 0 ;reset
        mov al, B
        sub al, A

        mov finalRslt, al
    call Display16bitsNumber
```

```
case2:
;   ii. A= - (A+1)

    mov ax, 0 ;reset
    mov al, A
    inc al
    neg al

    mov finalRslt, al
call Display16bitsNumber

case3:
;  iii. C=A+B

    mov ax, 0 ;reset
    mov al, A
    add al, B

    mov finalRslt, al
call Display16bitsNumber

case4:
;   iv. B=3*B+7

   mov ax, 0 ;reset
```

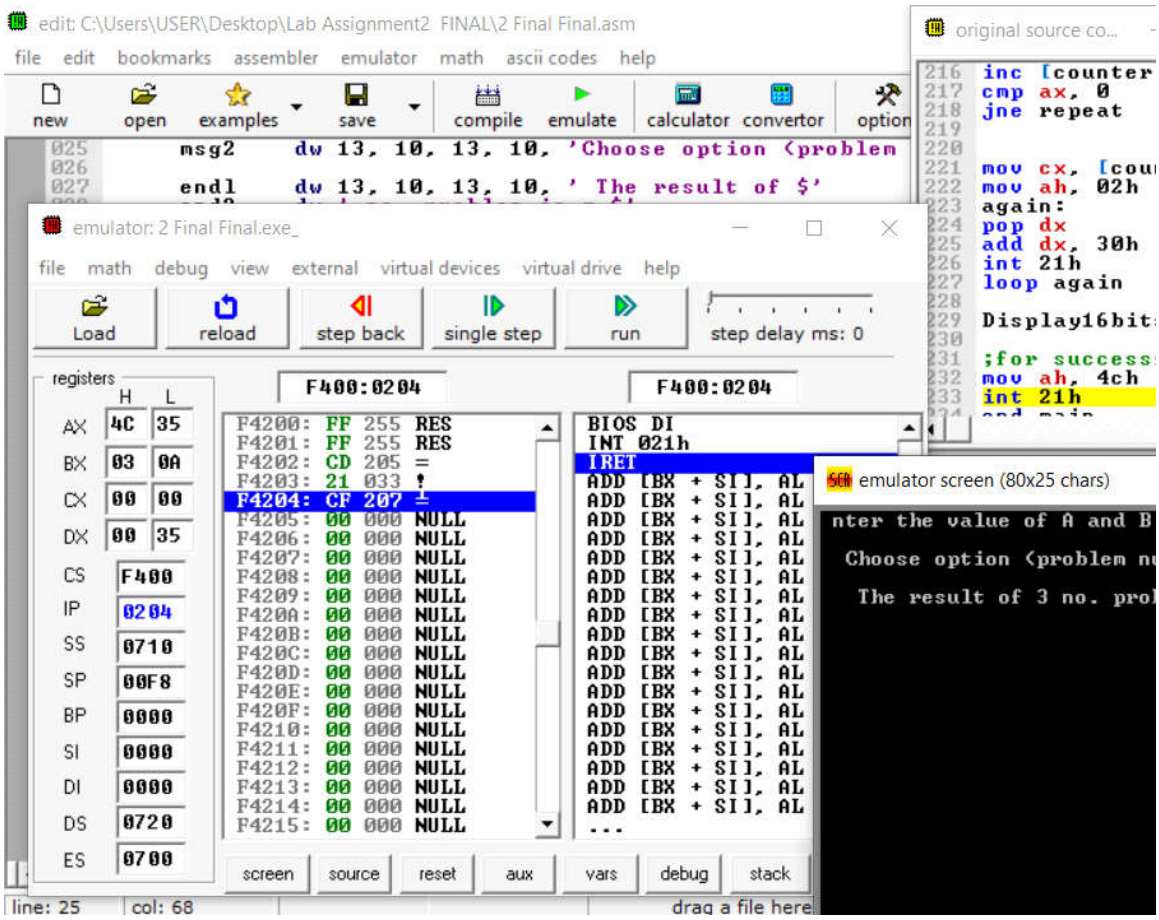
```
        mov al, B
        add al, B
        add al, B
        add al, 7

        mov finalRslt, al
    call Display16bitsNumber

    case5:
    ;   v. A=B-A-1

        mov ax, 0 ;reset
        mov al, B
        sub al, A
        sub al, 1

        mov finalRslt, al
    call Display16bitsNumber


main endp

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

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

```
inputNumber proc

    ;here we can input 2^8 - 1 = 255 maximum
;input number will store in bh register
mov bh, 0
mov bl, 10d


      INPUT:

      ;for input a single character
      mov ah, 1
      int 21h

      cmp al, 13d ;13d is the ASCII of enter key
      jne NUMBER

      jmp EXIT


      NUMBER:
      SUB AL,30H ;zero ASCII 48d = 30h

      mov cl, al ;store the al value bcz after mul it will be corrupted
      mov al, bh
```

```
    mul bl     ;8 bits multiplication
               ;ax = al * 8-bits reg
    add al, cl
    mov bh, al

    JMP INPUT

EXIT: ret

inputNumber 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

    ;print the option number
    mov dl, opsn
    mov ah, 02h
    int 21h

    ;print the msg2
```

```
    mov dx, offset end2
    mov ah, 9
    int 21h



    mov ax, 0 ;reset
    mov al, finalRslt ;A is 8 bits and we

    ;here we work with 8 bits thats why ax will not allow
    cmp al, 0          ;al < 0
    jge repeat         ;if al >= 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 al             ;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. Even or odd check

;div (8-bit register)

;ax = ax / 8-bit register

;al = quotient, ah = remainder

.model small

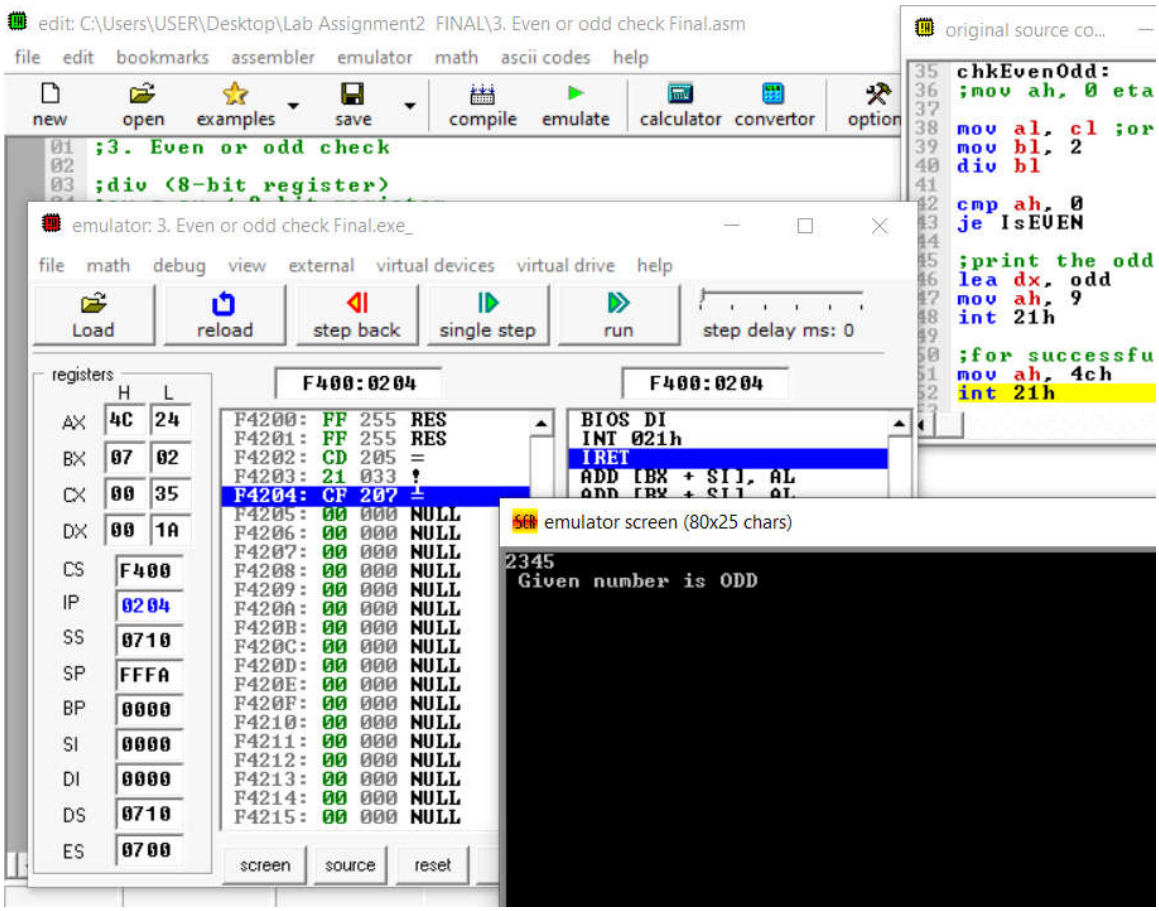
;.stack 100h

.data

```
    evn dw 10, 13, 'Given number is EVEN$' ;10, 13 for new line + carriage
return
    odd  dw 10, 13, 'Given number is ODD$'


.code

main proc

   mov bx, @data
   mov ds, bx



INPUT:
    mov ah, 1
    int 21h

    cmp al, 13 ;13 is ASCII of enter key
    je chkEvenOdd

    mov cl, al ;or mov cx, ax --- for store the last digit

jmp INPUT



chkEvenOdd:
        ;mov ah, 0 eta ekhane na dileo hobe bcz ax( 130 = 304d )
```

```
        mov al, cl ;or mov ax, cx
        mov bl, 2
        div bl

        cmp ah, 0
        je IsEVEN

        ;print the odd message
        lea dx, odd
        mov ah, 9
        int 21h

        ;for successfully terminate
        mov ah, 4ch
        int 21h

IsEVEN:

    ;print the even message
    lea dx, evn
    mov ah, 9
    int 21h

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

```
    int 21h


main endp
end main
```



;4. Whether a input number is prime or not/ Prime check

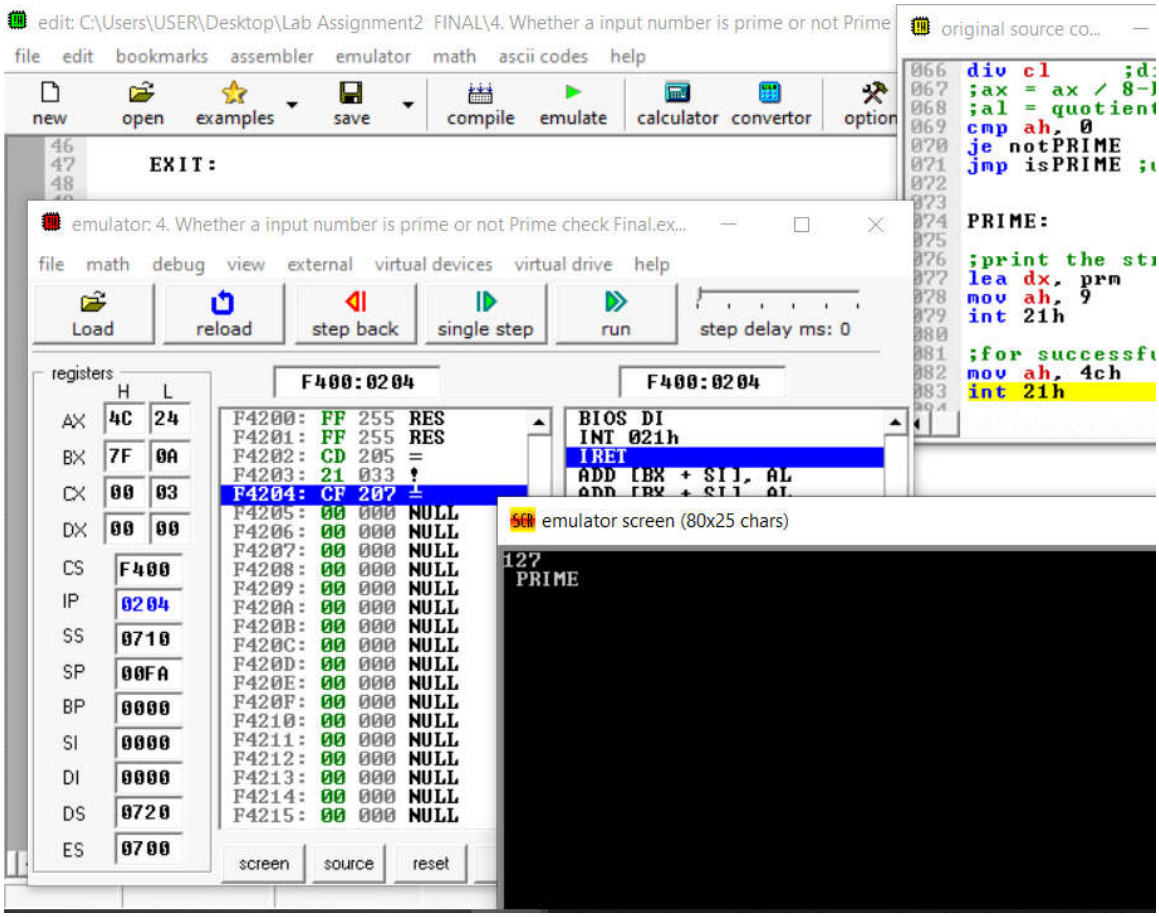
```
.model small
.stack 100h
.data

    prm  dw 10, 13, 'PRIME$'        ;10, 13 for new line + carriage return
    nprm dw 10, 13, 'NOT PRIME$'

.code
main proc
```

```
mov bx, @data
   mov ds, bx     ;initialize heap memory

;here we can input 2^8 - 1 = 255 (maximum)
;input number will store in bh register
mov bh, 0
mov bl, 10d


     INPUT:

     ;for input a single character
     mov ah, 1
     int 21h

     cmp al, 13d ;13d is the ASCII of enter key
     jne NUMBER

     jmp EXIT


     NUMBER:
     sub al, 30h ;zero ASCII 48d = 30h

     mov cl, al ;store the al value bcz after mul it will be corrupted
```

```
    mov al, bh

    mul bl     ;8 bits multiplication
               ;ax = al * 8-bits reg
    add al, cl
    mov bh, al

    JMP INPUT

EXIT:


 cmp bh, 1
   jle notPRIME

   mov cx, 0  ;reset
   mov cl, bh

   isPRIME:

       ;prepare for div operation
       mov ax, 0 ;reset
       mov al, bh

       dec cl    ;we will check value till n-1
       cmp cl, 3
```

```
        jle PRIME

        div cl     ;div (8-bit register)
                ;ax = ax / 8-bit register
                ;al = quotient, ah = remainder
        cmp ah, 0
        je notPRIME
        jmp isPRIME ;unconditional jump


PRIME:

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

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


notPRIME:

    ;print the string
    lea dx, nprm
```

```
    mov ah, 9
    int 21h

    ;for successsfully return
    mov ah, 4ch
    int 21h

main endp
end main
```

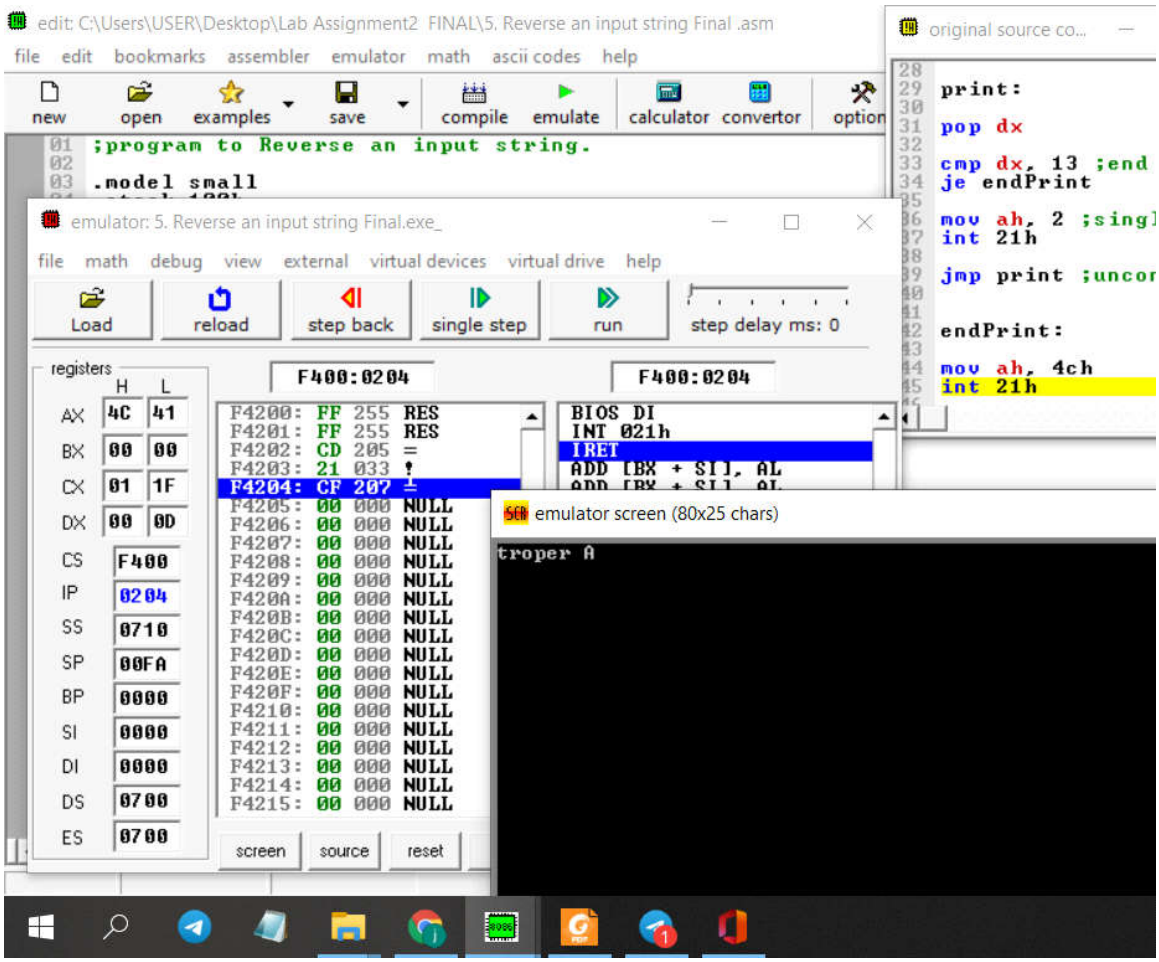


;program to Reverse an input string.

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

main proc

    ;put ASCII 13 to mark end of string
    mov ax, 13
```

```
        push ax

INPUT:

        mov ah, 1
        int 21h

        cmp al, 13 ;13 is ASCII of Enter key
        je reversePrint
        push ax

jmp INPUT ;unconditional jump



reversePrint:

      print:

          pop dx

          cmp dx, 13 ;end of string
          je endPrint

          mov ah, 2 ;single char print tai vul astese na ; ekhane bug ase
          int 21h
```

```
    jmp print ;unconditional jump


endPrint:

    mov ah, 4ch
    int 21h


main endp

end main
```



;6. Write a assembly code to perform the following:

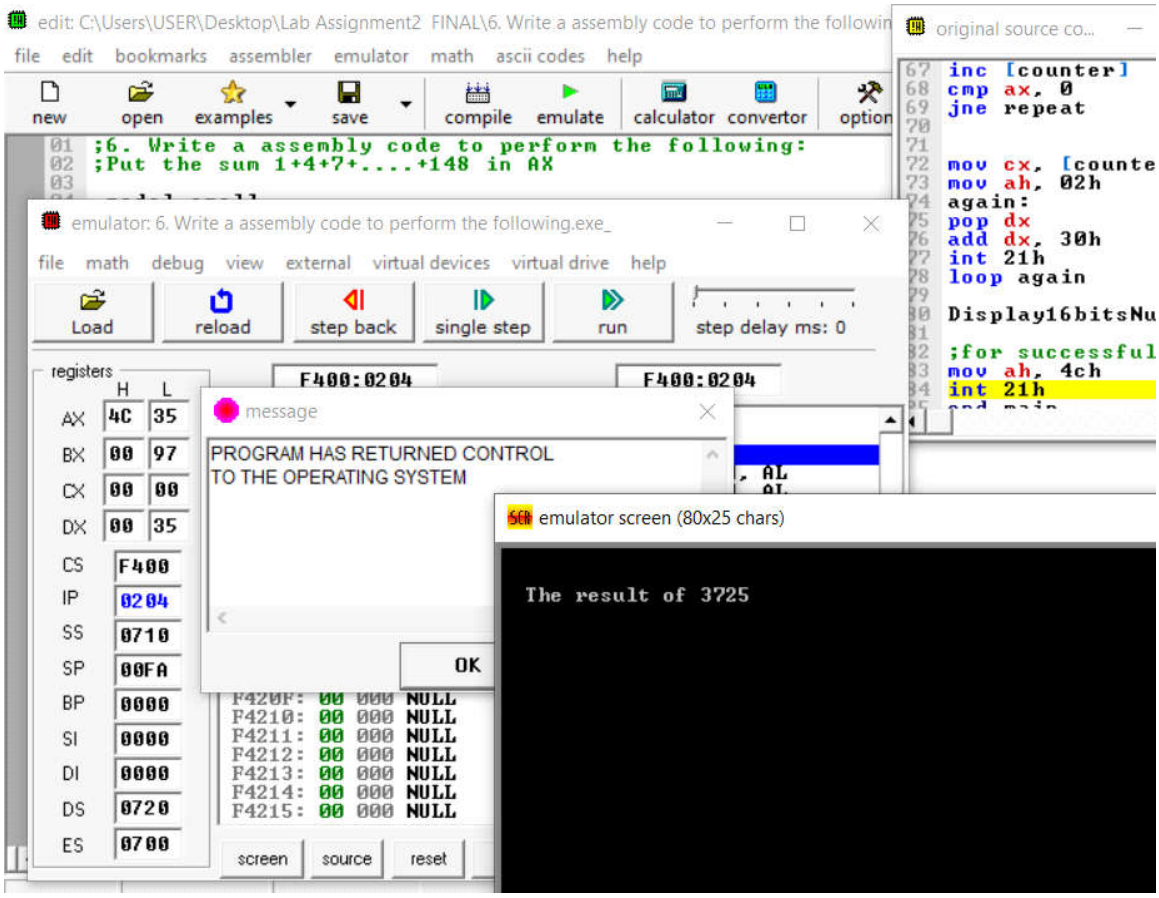
;Put the sum 1+4+7+....+148 in AX

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

    base    dw 10
    endl    dw 13, 10, 13, 10, 'The result of $'
    counter dw 0
```

```
   sum     dw 0

.code

main proc

        mov ax, @data
        mov ds, ax

        mov bx, 1
        mov ax, 0
repet:

        cmp bx, 148
        jg Display16bitsNumber

        add ax, bx
        add bx, 3
jmp repet ;unconditional jump


main endp


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

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

```
****************************************

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

    mov sum, ax
    ;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
```


edit: C:\Users\USER\Desktop\Lab Assignment2 FINAL\6. Write a assembly code to perform the followin
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
;6. Write a assembly code to perform the following:
;Put the sum 1+4+7+....+148 in AX
original source co...
inc [counter]
cmp ax, 0
jne repeat
mov cx, [counte
mov ah, 02h
again:
pop dx
add dx, 30h
int 21h
loop again
Display16bitsNu
;for successful
mov ah, 4ch
int 21h
emulator: 6. Write a assembly code to perform the following.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 4C 35
BX 00 97
CX 00 00
DX 00 35
CS F400
IP 0204
SS 0710
SP 00FA
BP 0000
SI 0000
DI 0000
DS 0720
ES 0700
F400:0204
F400:0204
message
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
OK
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
screen source reset
emulator screen (80x25 chars)
The result of 3725