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,**

Mst. Arafatun Jannat Tania ID: 1605040 Reg: 000008679 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

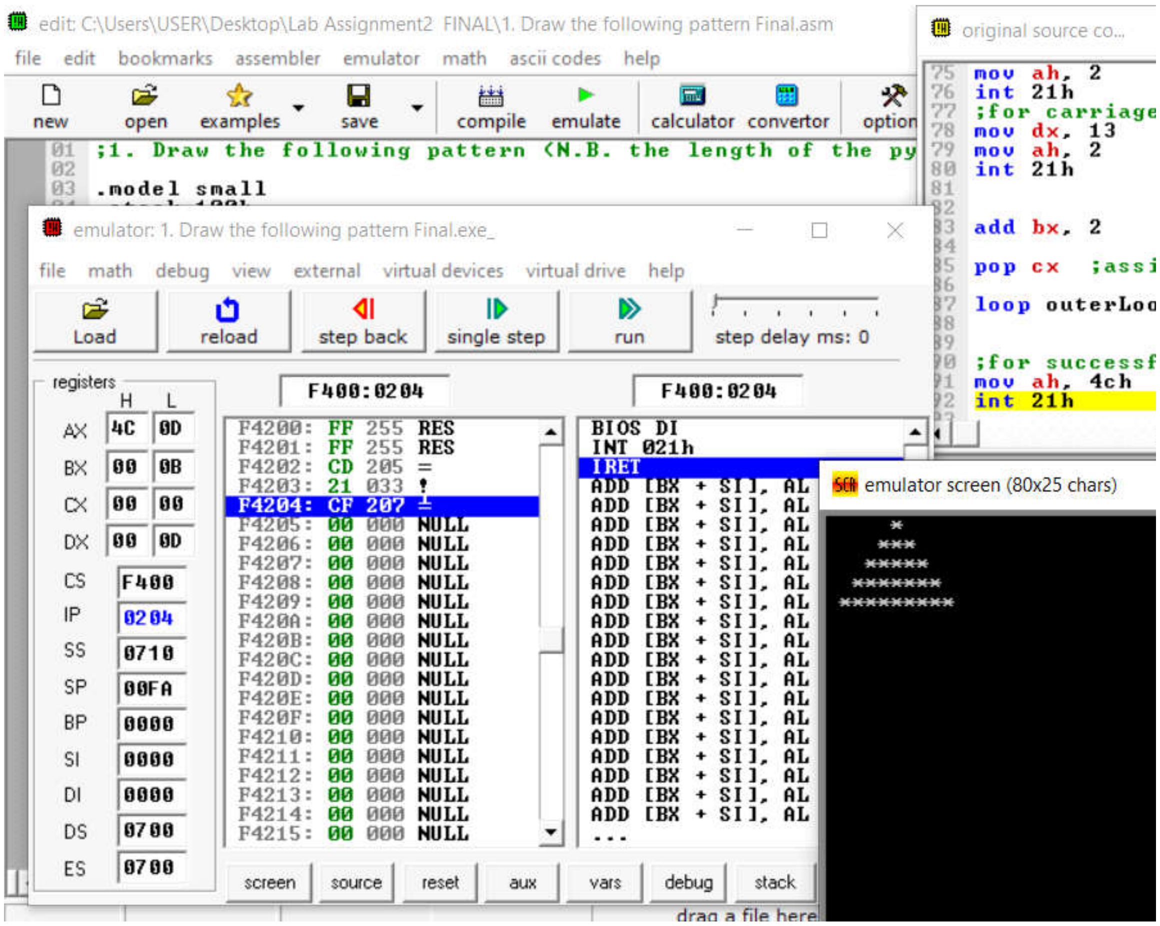
Submission Date: 26 Februay, 2022

;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
```


edit: C:\Users\USER\Desktop\Lab Assignment2 FINAL\1. Draw the following pattern Final.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
original source co...
emulator: 1. Draw the following pattern Final.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack
drag a file here
emulator screen (80x25 chars)

```
;1. Draw the following pattern (N.B. the length of the py
.model small
```

```
mov ah, 2
int 21h
;for carriage
mov dx, 13
mov ah, 2
int 21h

add bx, 2

pop cx   ;assi

loop outerLoo

;for successf
mov ah, 4ch
int 21h
```

	H	L
AX	4C	0D
BX	00	0B
CX	00	00
DX	00	0D
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
F400:0204
F4200:  FF 255 RES
F4201:  FF 255 RES
F4202:  CD 205 =
F4203:  21 033 !
F4204:  CF 207 ╧
F4205:  00 000 NULL
F4206:  00 000 NULL
F4207:  00 000 NULL
F4208:  00 000 NULL
F4209:  00 000 NULL
F420A:  00 000 NULL
F420B:  00 000 NULL
F420C:  00 000 NULL
F420D:  00 000 NULL
F420E:  00 000 NULL
F420F:  00 000 NULL
F4210:  00 000 NULL
F4211:  00 000 NULL
F4212:  00 000 NULL
F4213:  00 000 NULL
F4214:  00 000 NULL
F4215:  00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

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

;3. Even or odd check

;div (8-bit register)

;ax = ax / 8-bit register

;al = quotient, ah = remainder

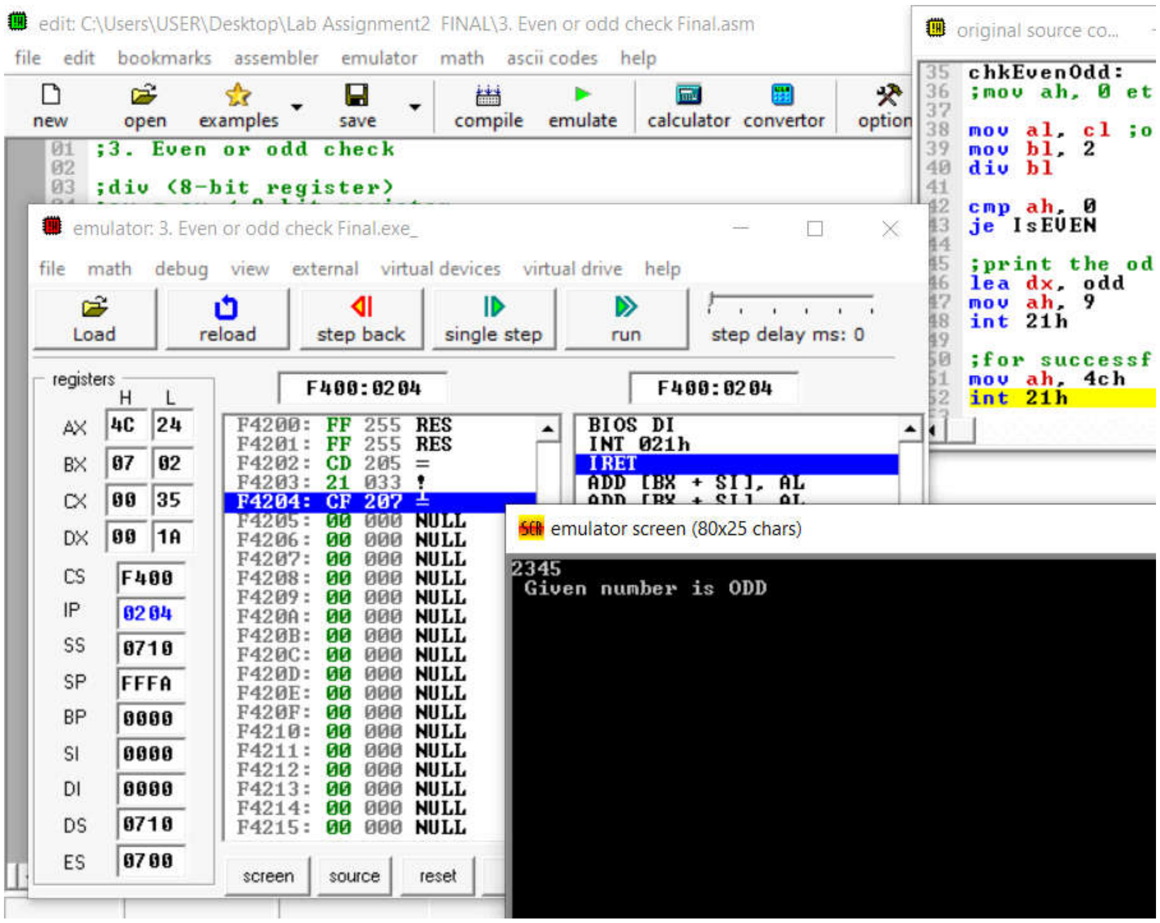
.model small

;.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
```


edit: C:\Users\USER\Desktop\Lab Assignment2 FINAL\3. Even or odd check Final.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
original source co...
emulator: 3. Even or odd check Final.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset
emulator screen (80x25 chars)

```
;3. Even or odd check

;div (8-bit register)
```

```
chkEvenOdd:
;mov ah, 0 et

mov al, cl ;o
mov bl, 2
div bl

cmp ah, 0
je IsEVEN

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

;for successf
mov ah, 4ch
int 21h
```

	H	L
AX	4C	24
BX	07	02
CX	00	35
DX	00	1A
CS	F400	
IP	0204	
SS	0710	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
```

```
2345
 Given number is ODD
```

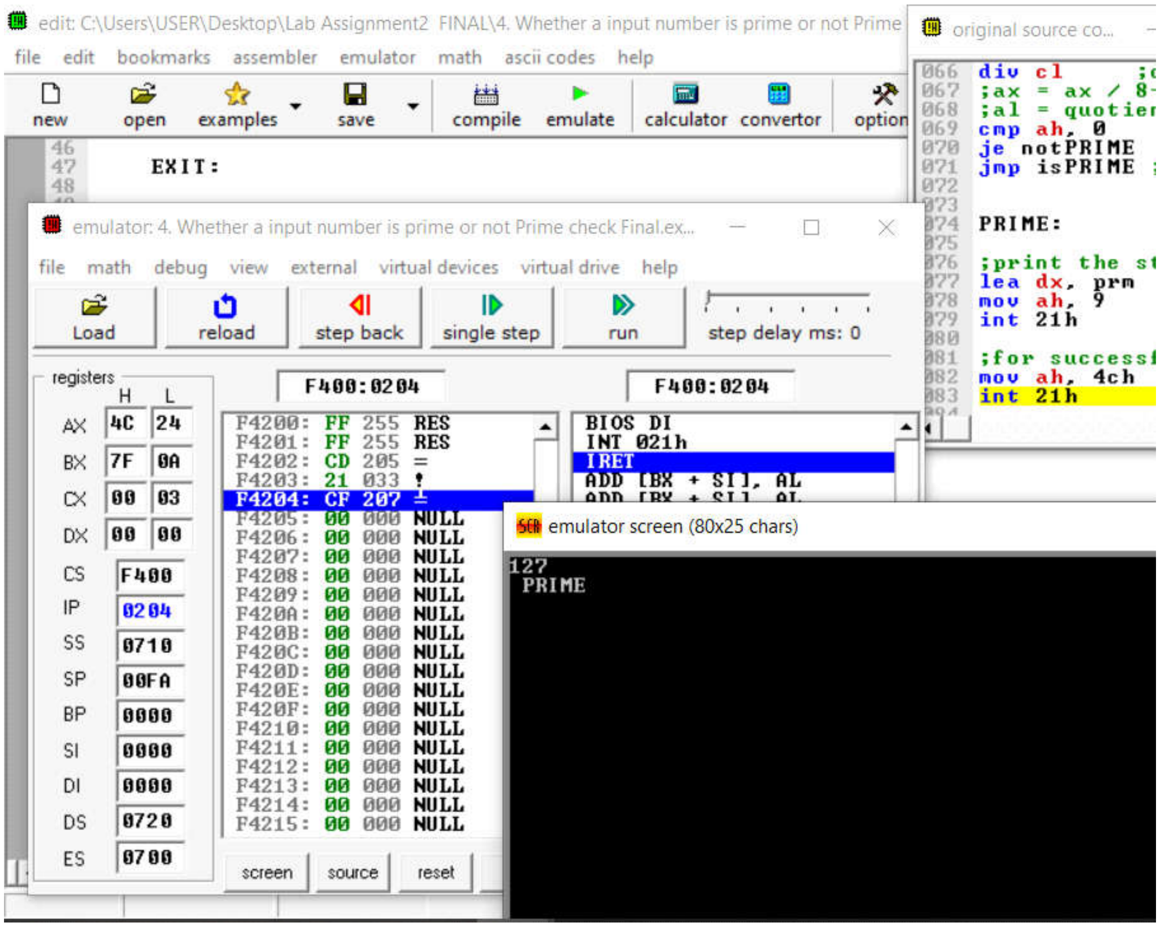
;4. Whether a input number is prime or not/ Prime check

```
.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
```


edit: C:\Users\USER\Desktop\Lab Assignment2 FINAL\4. Whether a input number is prime or not Prime
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
original source co...
emulator: 4. Whether a input number is prime or not Prime check Final.ex...
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
emulator screen (80x25 chars)
screen source reset

```
        EXIT:
```

```
div cl          ;c
;ax = ax / 8-
;al = quotier
cmp ah, 0
je notPRIME
jmp isPRIME ;


PRIME:

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

;for successf
mov ah, 4ch
int 21h
```

```
registers
     H    L
AX  4C   24
BX  7F   0A
CX  00   03
DX  00   00
CS  F400
IP  0204
SS  0710
SP  00FA
BP  0000
SI  0000
DI  0000
DS  0720
ES  0700
```

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
```

```
127
 PRIME
```

;program to Reverse an input string.

.model small

.stack 100h

.data

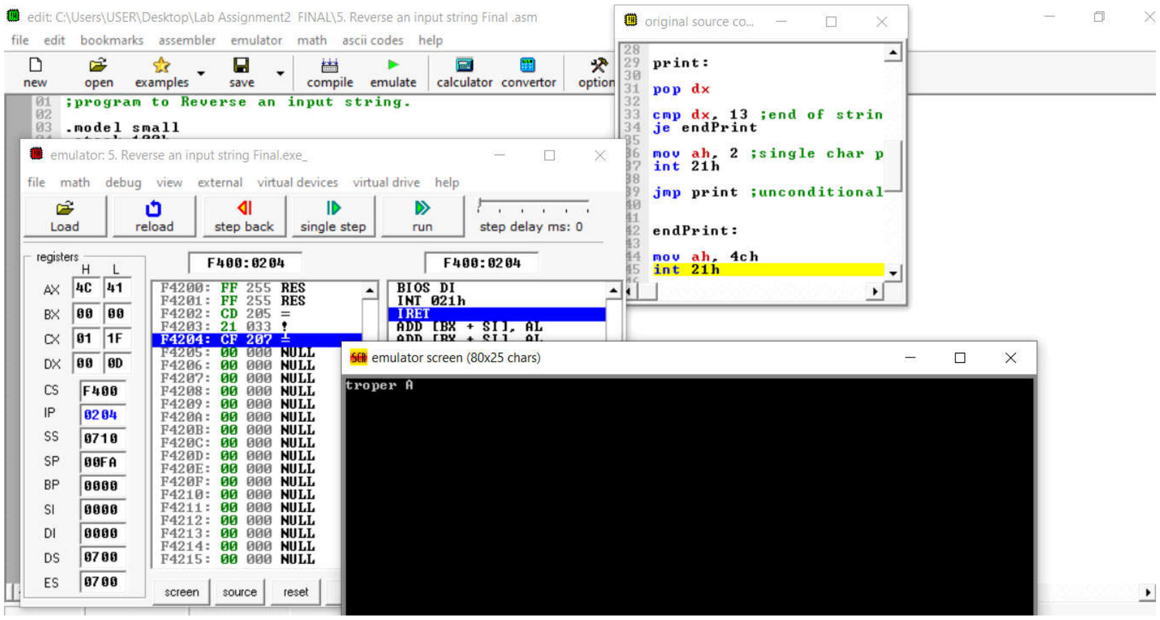
.code

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
```


edit: C:\Users\USER\Desktop\Lab Assignment2 FINAL\5. Reverse an input string Final .asm

```
;program to Reverse an input string.

.model small
```

original source co...

```
print:

pop dx

cmp dx, 13 ;end of strin
je endPrint

mov ah, 2 ;single char p
int 21h

jmp print ;unconditional


endPrint:

mov ah, 4ch
int 21h
```

emulator: 5. Reverse an input string Final.exe_

registers

	H	L
AX	4C	41
BX	00	00
CX	01	1F
DX	00	0D
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
```

emulator screen (80x25 chars)

```
troper A
```

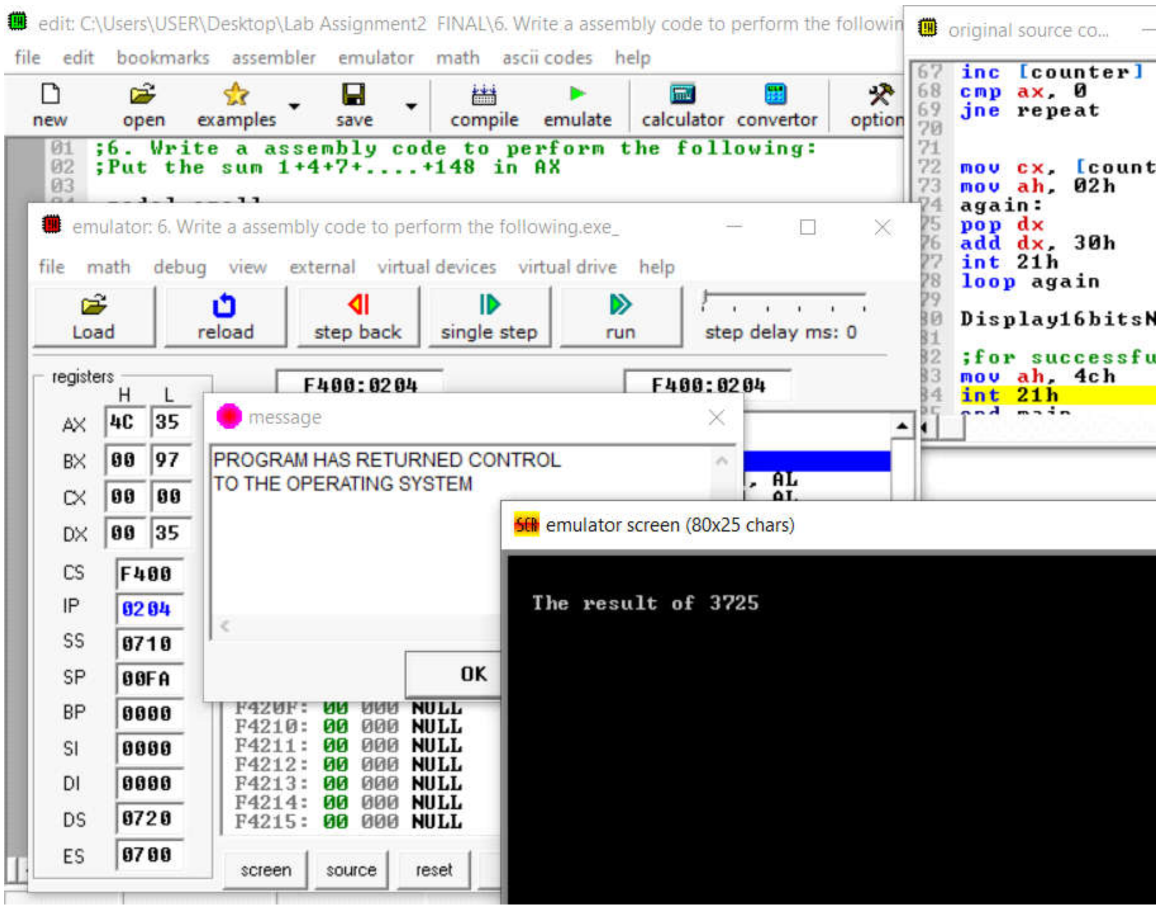
;6. Write a assembly code to perform the following:

```
;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
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
original source co...
emulator screen (80x25 chars)
screen source reset

```
;6. Write a assembly code to perform the following:
;Put the sum 1+4+7+....+148 in AX
```

```
inc [counter]
cmp ax, 0
jne repeat

mov cx, [count
mov ah, 02h
again:
pop dx
add dx, 30h
int 21h
loop again

Display16bitsN

;for successfu
mov ah, 4ch
int 21h
```

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
```

```
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

message

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

OK

```
The result of 3725
```