MICROPROCESSOR AND INTERFACING

ASSIGNMENT-1

NAME: SIDDHARTH GAUTAM

REG NO: 19BCE0806

1.

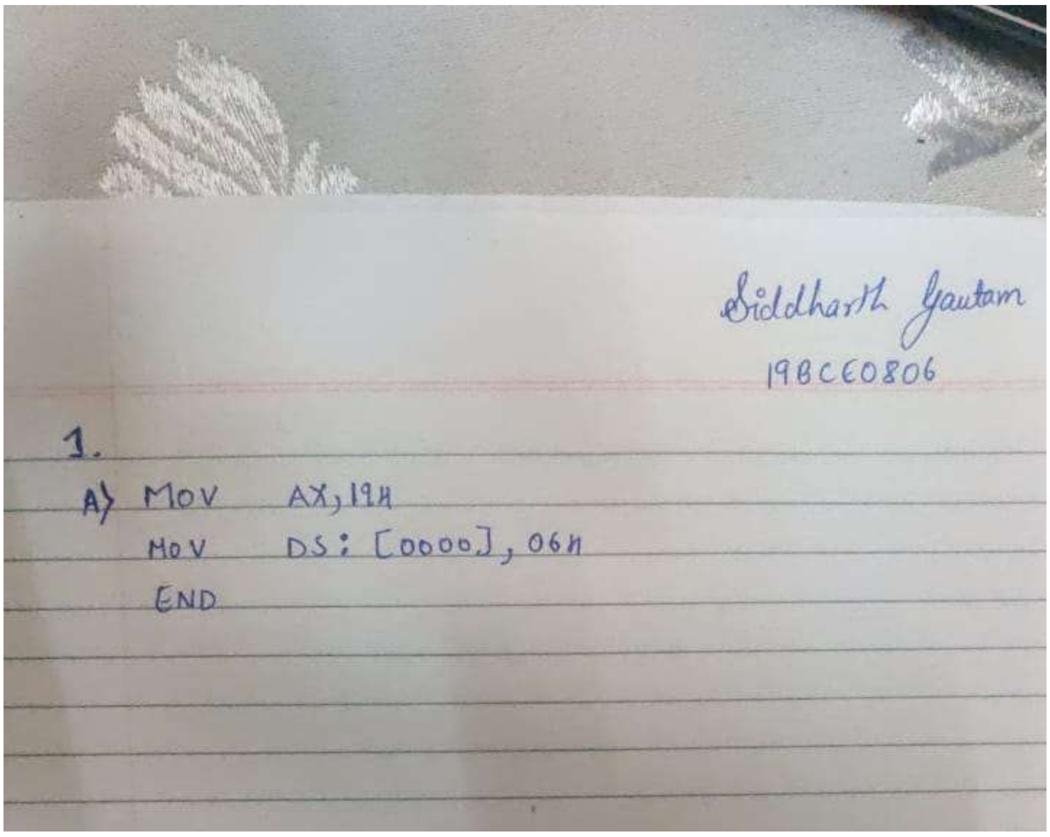
A)

Perform two data transfer function:

Data1 -General Purpose Registers

Data2 - Any memory location

CODE:

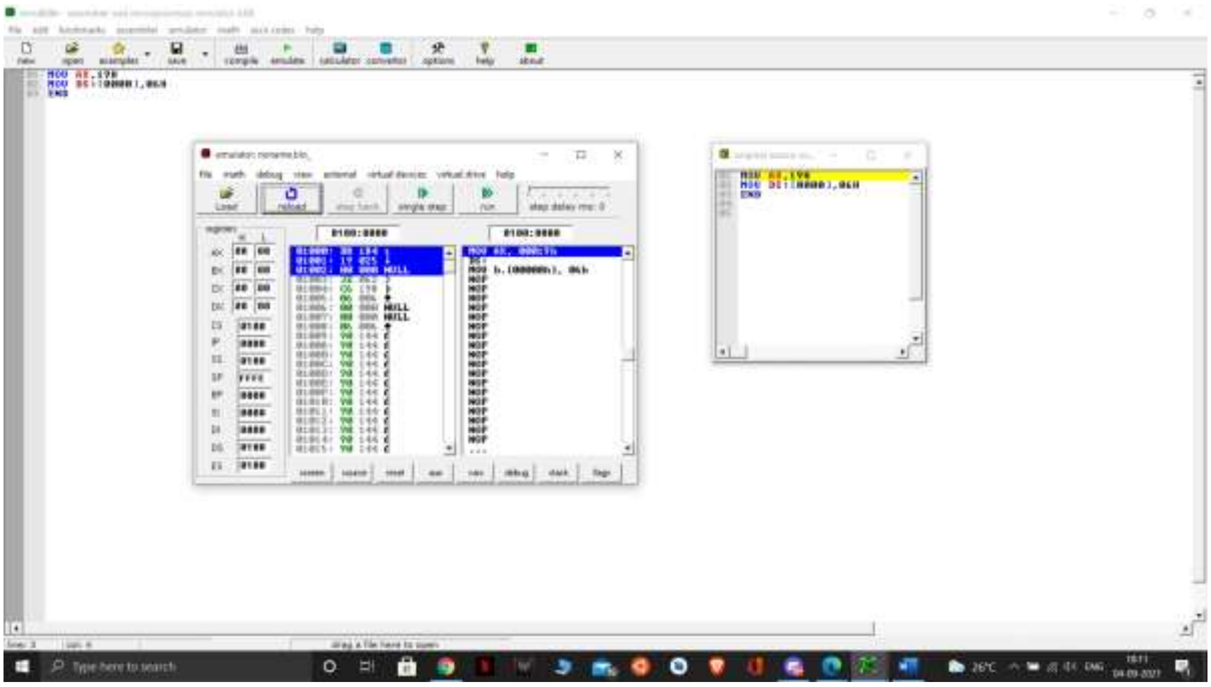


Siddharth Gautam
19BCE0806

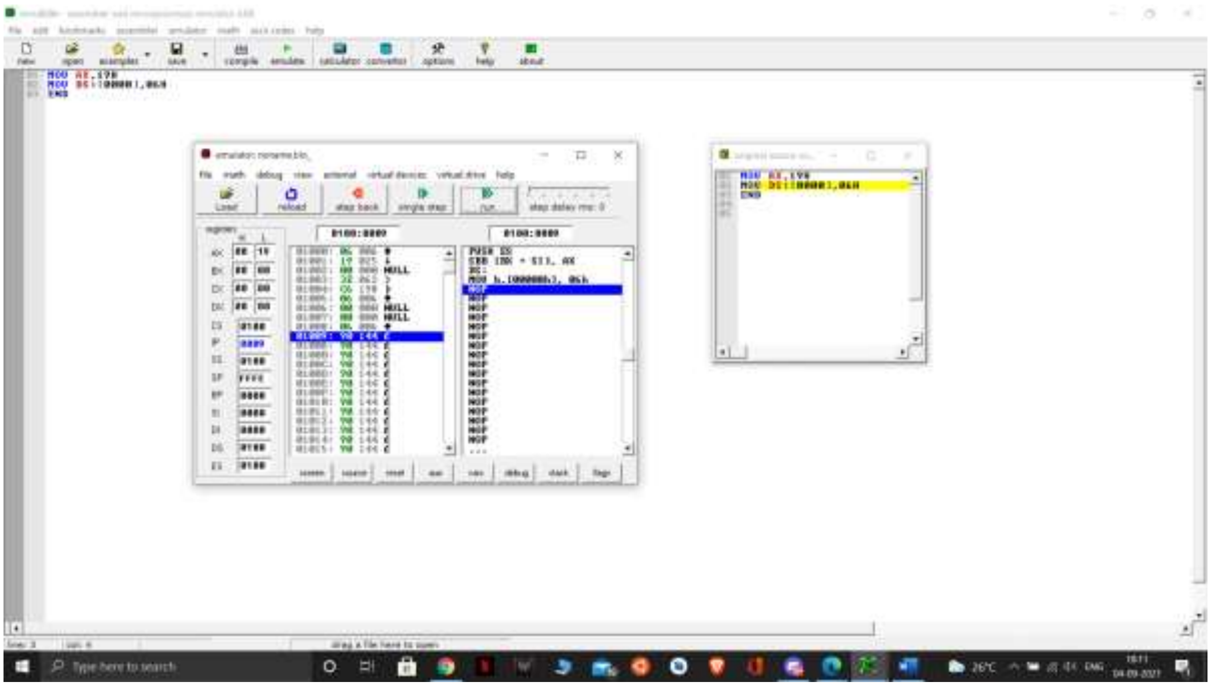
1.

```
A) MOV   AX,19H
   MOV   DS:[0000],06H
   END
```

OUTPUT:



```
MOV AX,17H
MOV DS:[0000],06H
END
```

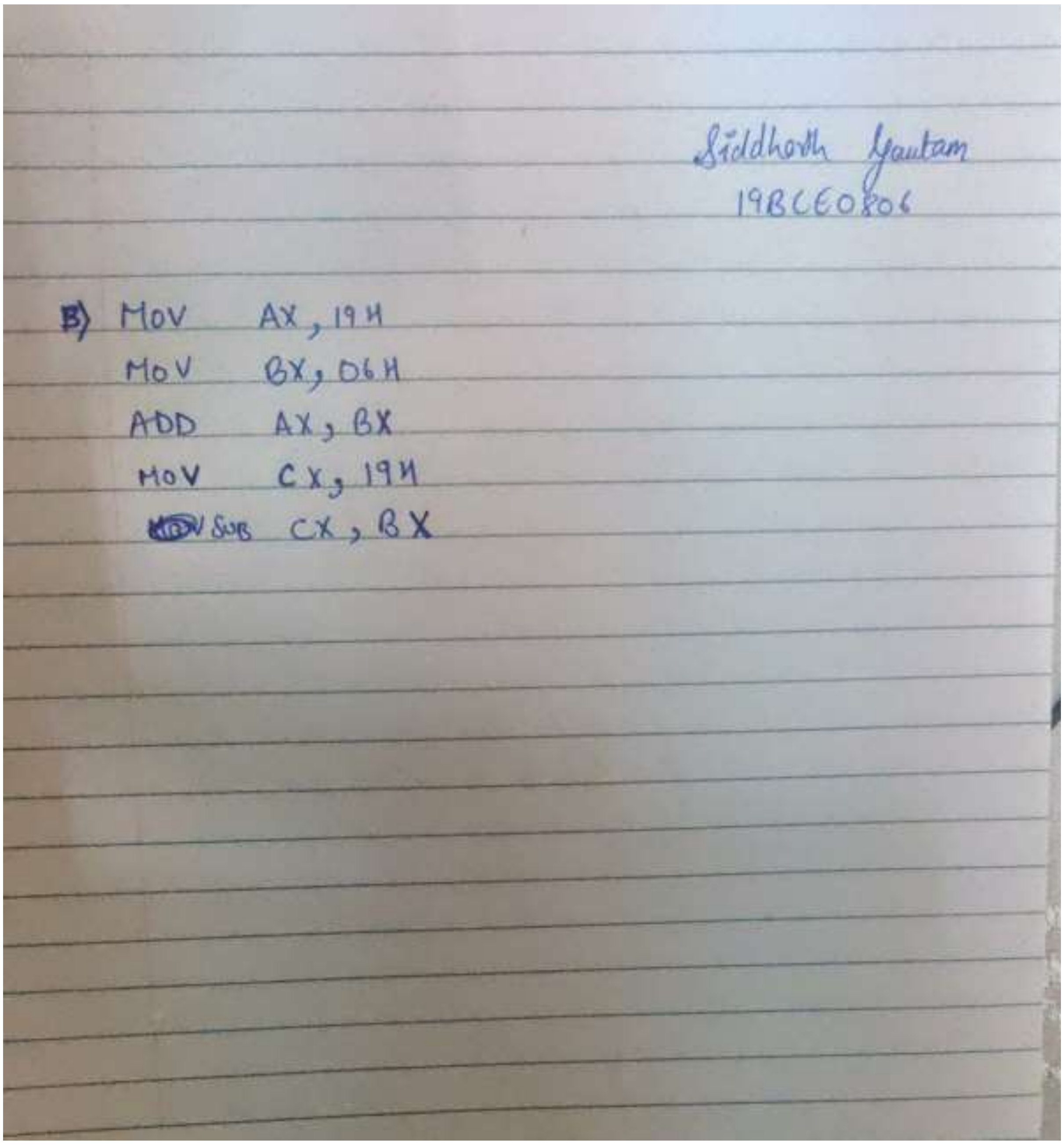


```
MOV AX,17H
MOV DS:[0000],06H
END
```

B)

Perform any two Arithmetic functions using Data 1 and 2: (ADD/SUB/MUL/DIV) Store the result in General Purpose Registers Check the status of Flag register

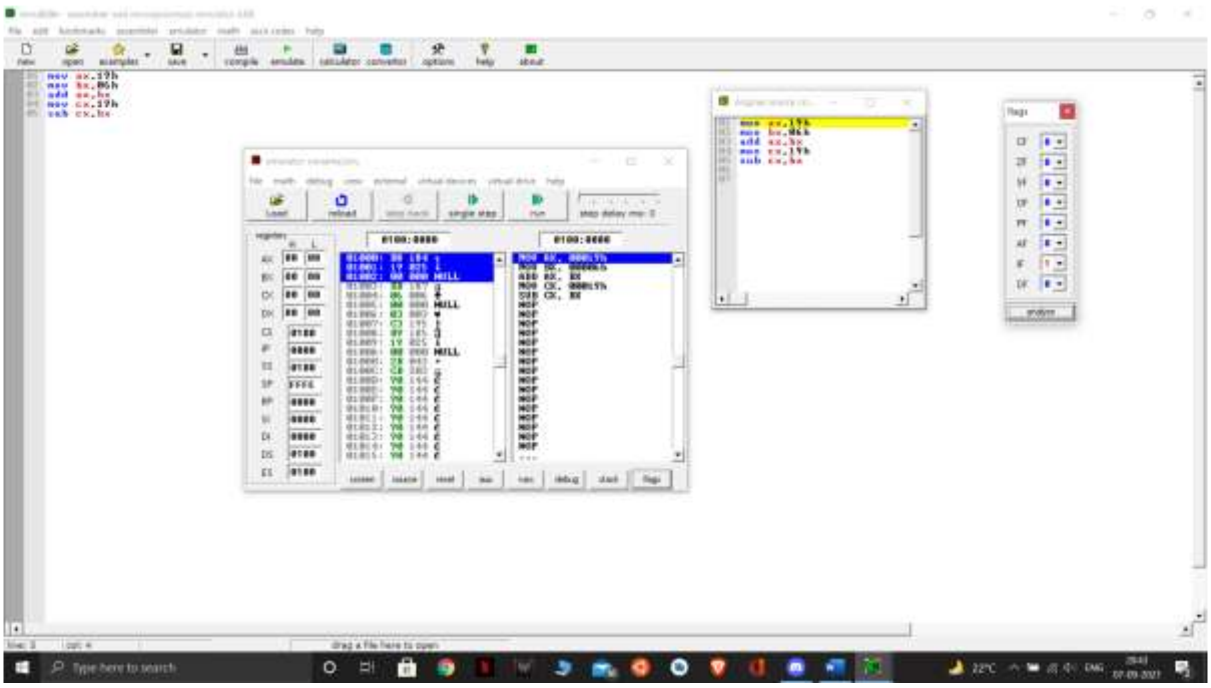
CODE:



Siddharth Gautam
19BCE0806

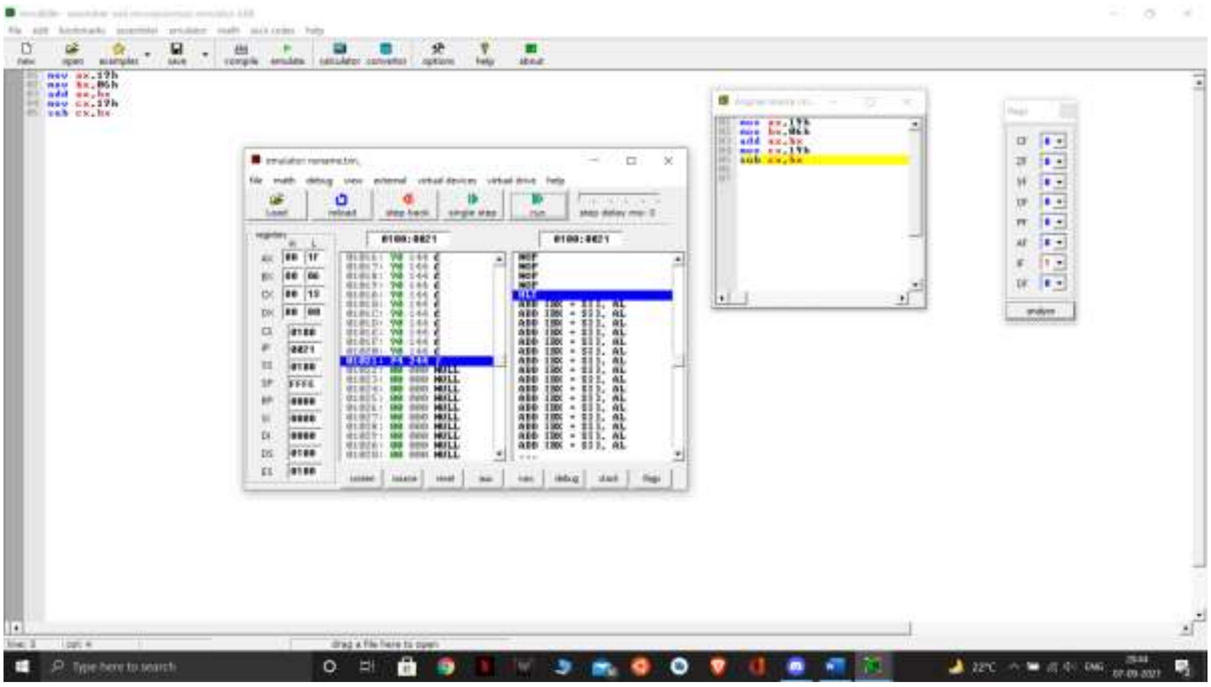
```
B) MOV   AX, 19H
   MOV   BX, 06H
   ADD   AX, BX
   MOV   CX, 19H
   SUB   CX, BX
```

OUTPUT:



```
mov ax,17h
mov bx,06h
add ax,bx
mov cx,17h
sub cx,bx
```

```
MOV AX, 00017h
MOV BX, 00006h
ADD AX, BX
MOV CX, 00017h
SUB CX, BX
```



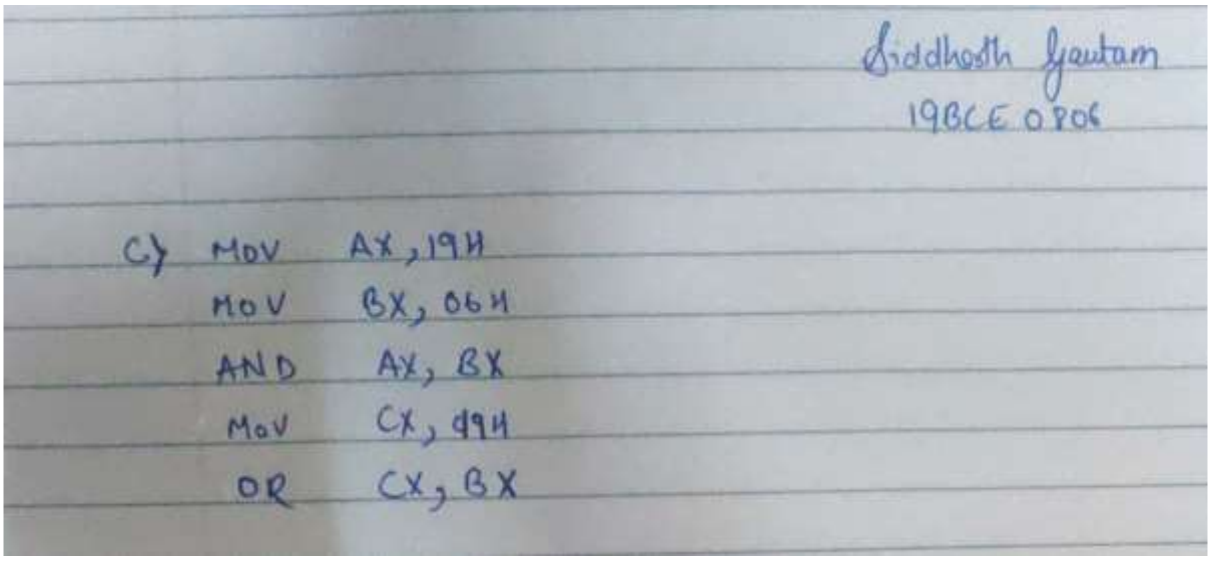
```
mov ax,17h
mov bx,06h
add ax,bx
mov cx,17h
sub cx,bx
```

```
AX 00 1D
BX 00 06
CX 00 11
DX 00 00
CS 0100
IP 0021
SS 0100
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
```

C)

Perform any two Logical functions using DATA 1 and DATA 2: (AND/OR/XOR) Store the result in General Purpose Registers Check the status of Flag register

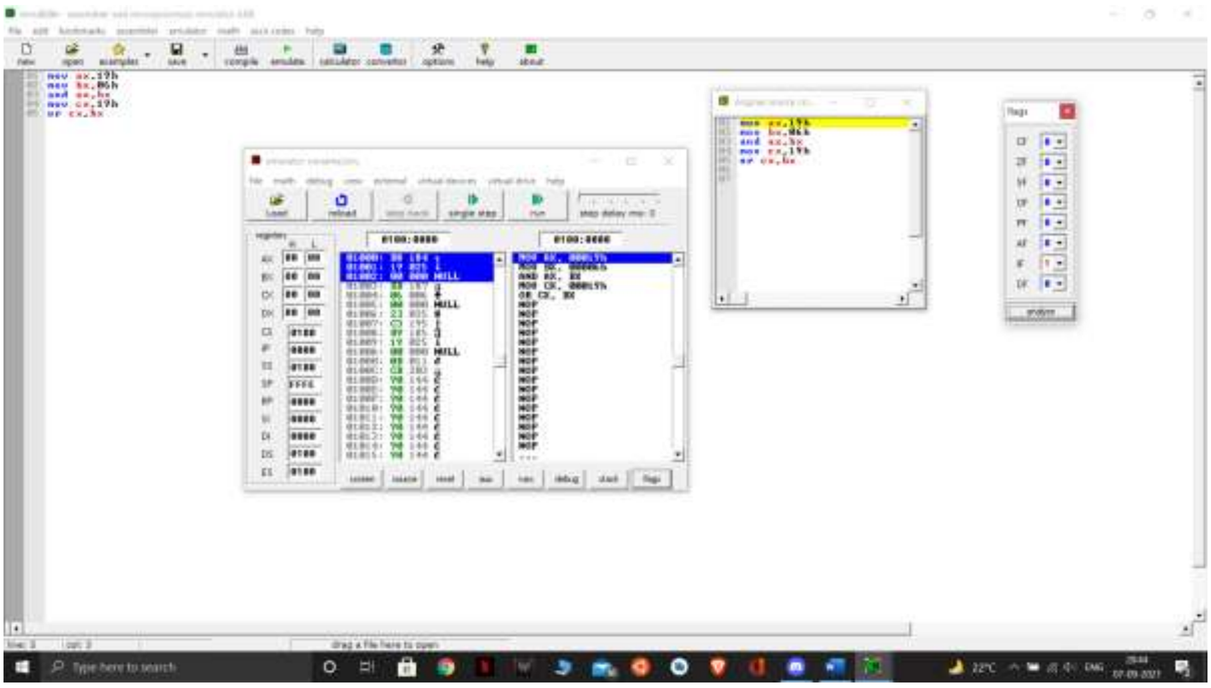
CODE:



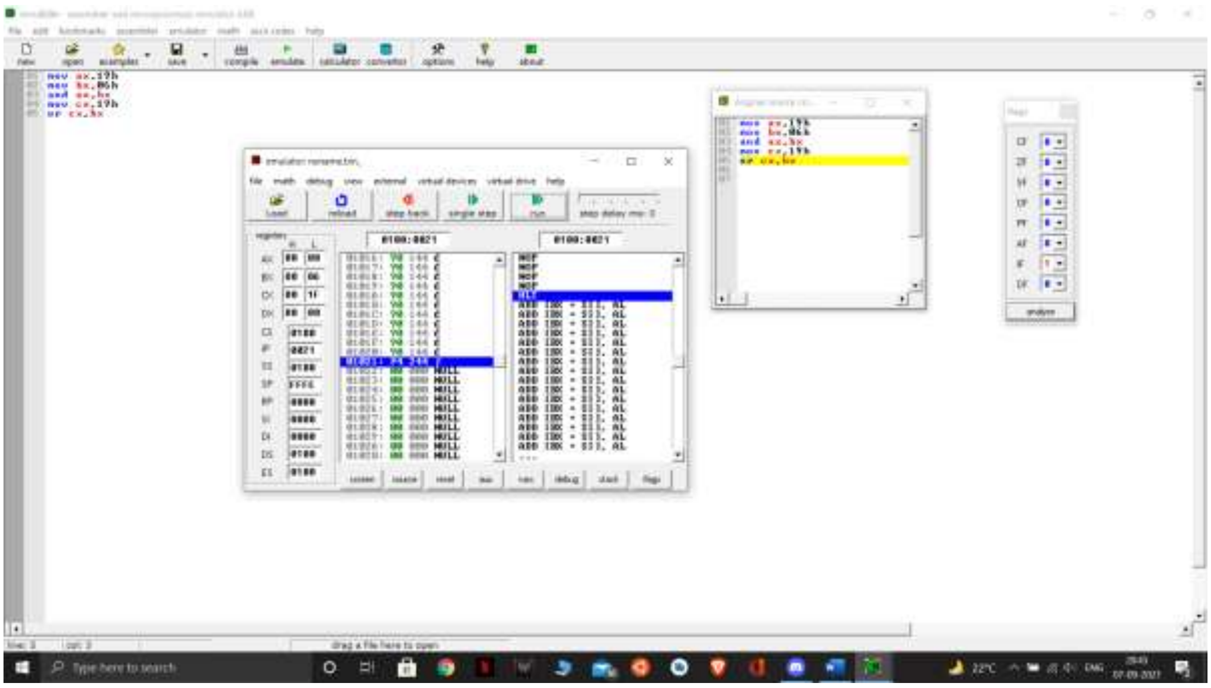
Siddharth Gautam
19BCE0806

```
C)  MOV   AX,19H
    MOV   BX,06H
    AND   AX,BX
    MOV   CX,49H
    OR    CX,BX
```

OUTPUT:



```
mov ax,19h
mov bx,06h
and ax,bx
mov cx,19h
or cx,bx
```



```
mov ax,19h
mov bx,06h
and ax,bx
mov cx,19h
or cx,bx
```

```
AX  00 00
BX  00 06
CX  00 1F
DX  00 00
CS  0100
IP  0021
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

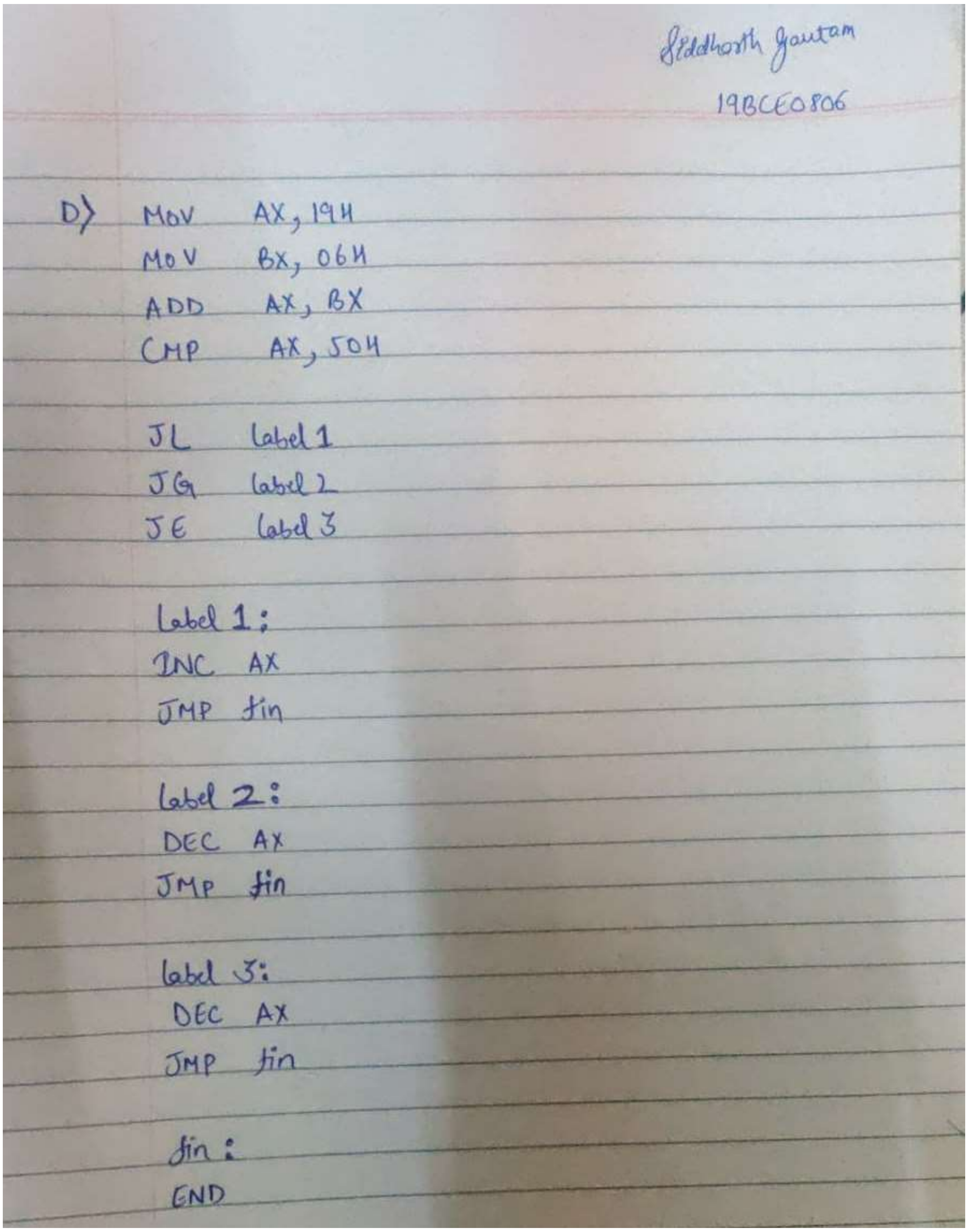
```
0100:0021
```

```
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
```

D)

Add Data 1 and 2, if the result is greater than 50H then answer is result – 1 else the answer is result + 1. (use branching instructions)

CODE:

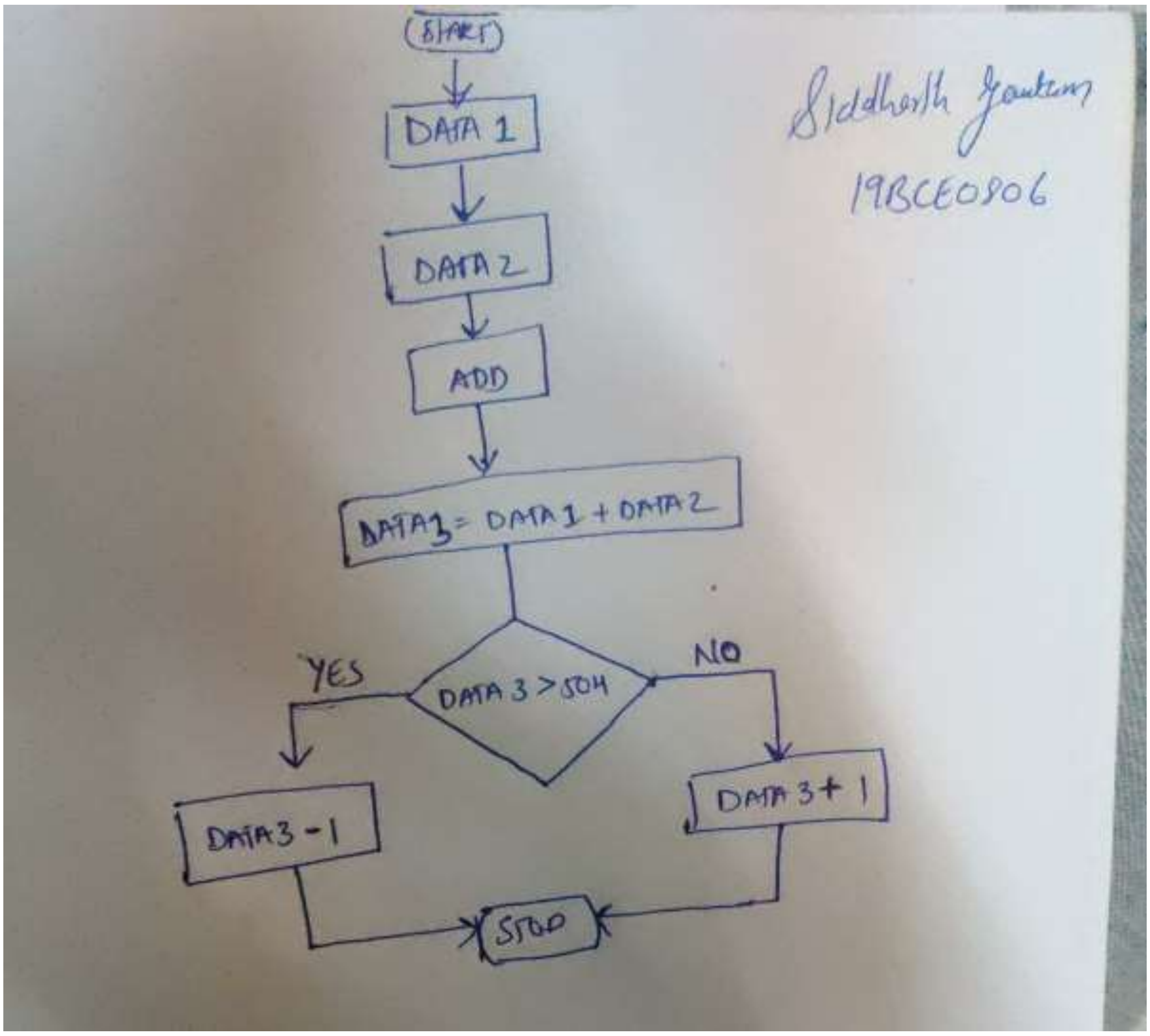


Siddharth Gautam

19BCE0806

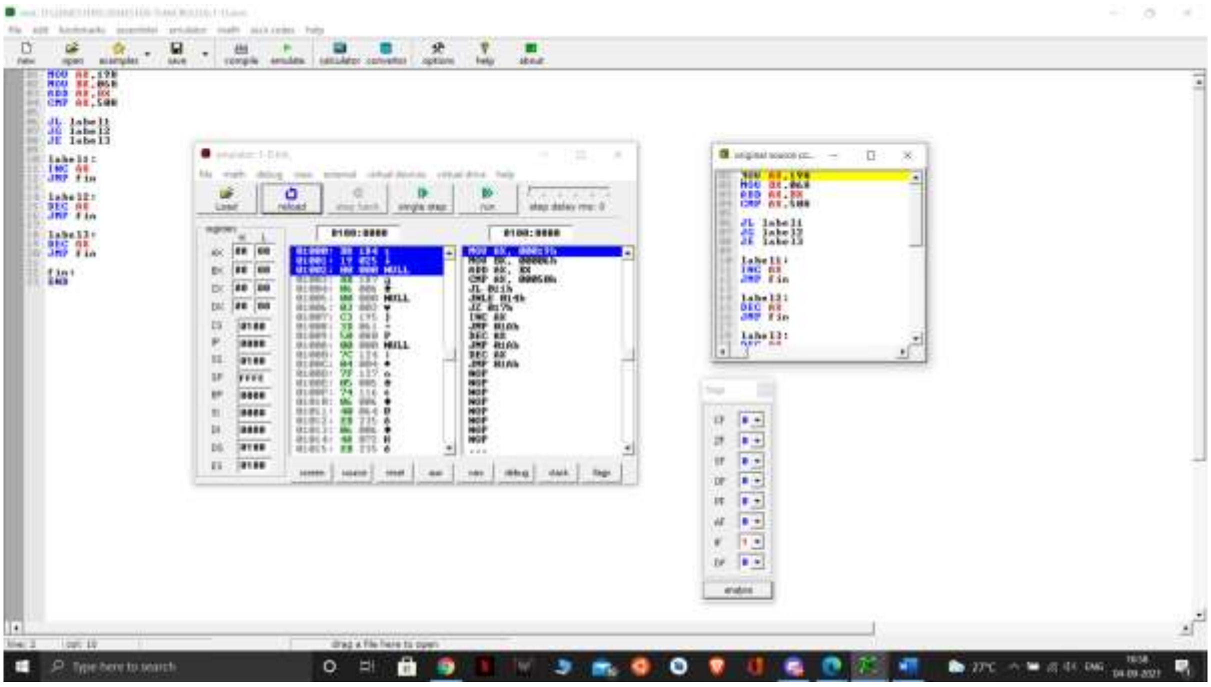
```
D)  MOV   AX, 19H
    MOV   BX, 06H
    ADD   AX, BX
    CMP   AX, 50H

    JL    label 1
    JG    label 2
    JE    label 3

    label 1:
    INC  AX
    JMP  fin

    label 2:
    DEC  AX
    JMP  fin

    label 3:
     DEC  AX
     JMP  fin

    fin :
    END
```

FLOWCHART:

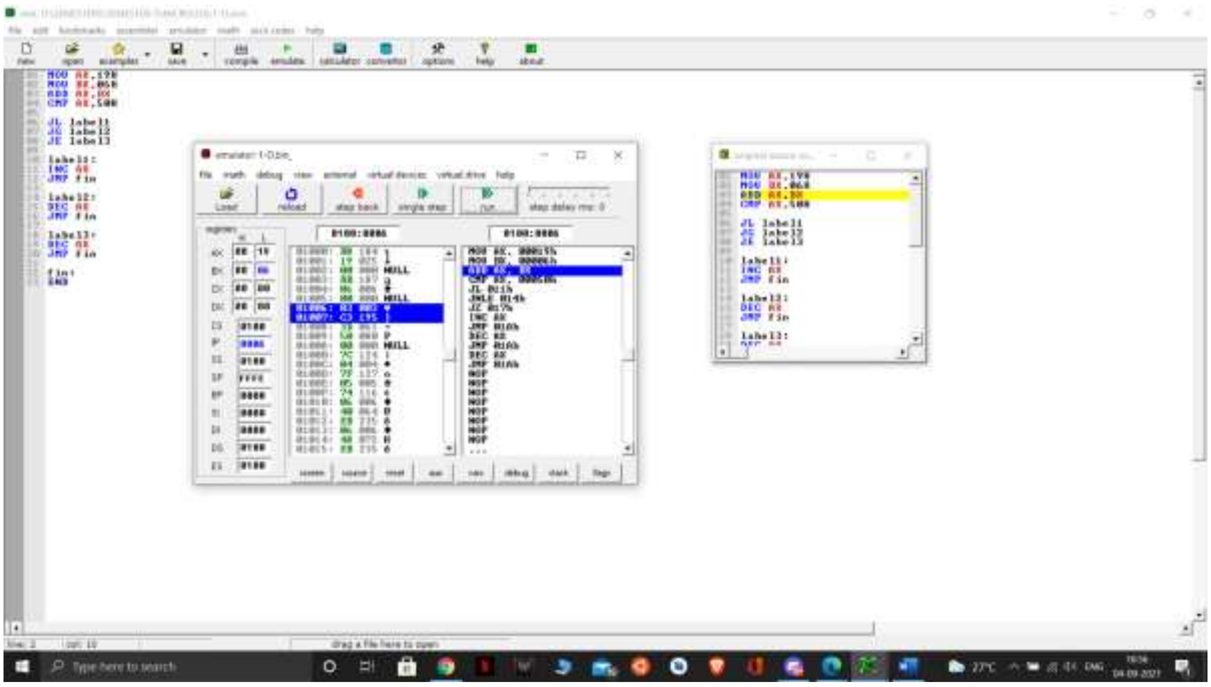


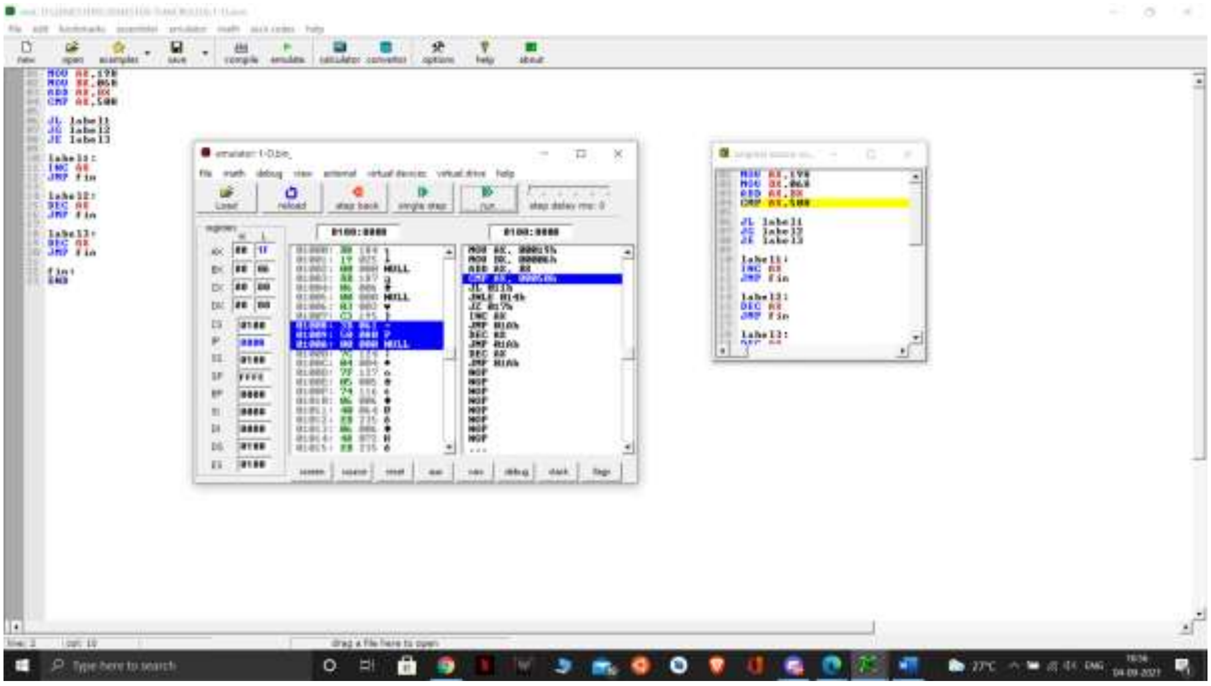
START
DATA 1
DATA 2
ADD
DATA3 = DATA 1 + DATA 2
DATA 3 > 504
YES
NO
DATA3 - 1
DATA 3 + 1
STOP
Siddharth Gautam
19BCE0806

OUTPUT:



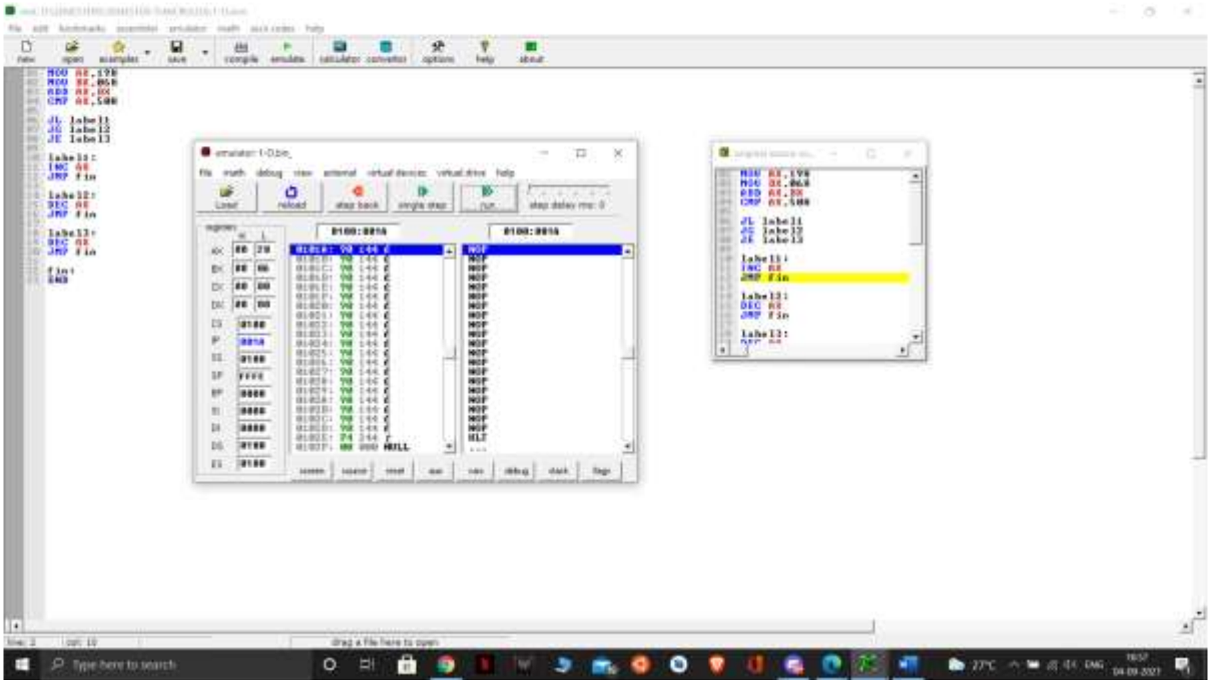
```
MOV AX,19H
MOV BX,06H
ADD AX,BX
CMP AX,50H

JL label1
JG label2
JE label3

label1:
INC AX
JMP fin

label2:
DEC AX
JMP fin

label3:
DEC AX
JMP fin

fin:
END
```



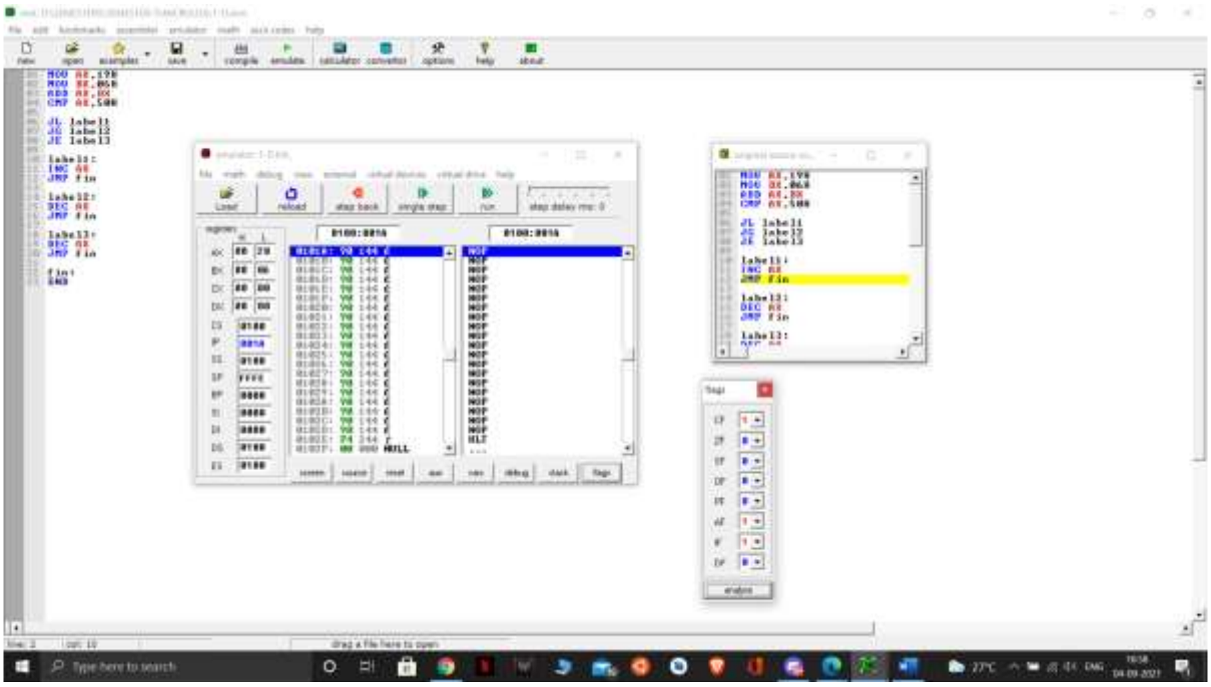


```
MOV AX,17H
MOV BX,06H
ADD AX,BX
CMP AX,50H

JL label1
JG label2
JE label3

label1:
INC AX
JMP fin

label2:
DEC AX
JMP fin

label3:
DEC AX
JMP fin

fin:
END
```

```
MOV AX, 00017h
MOV BX, 00006h
ADD AX, BX
CMP AX, 00050h
JL 0113h
JNLE 014h
JZ 017h
INC AX
JMP 01Ah
DEC AX
JMP 01Ah
DEC AX
JMP 01Ah
NOP
```



```
MOV AX,17H
MOV BX,06H
ADD AX,BX
CMP AX,50H

JL label1
JG label2
JE label3

label1:
INC AX
JMP fin

label2:
DEC AX
JMP fin

label3:
DEC AX
JMP fin

fin:
END
```

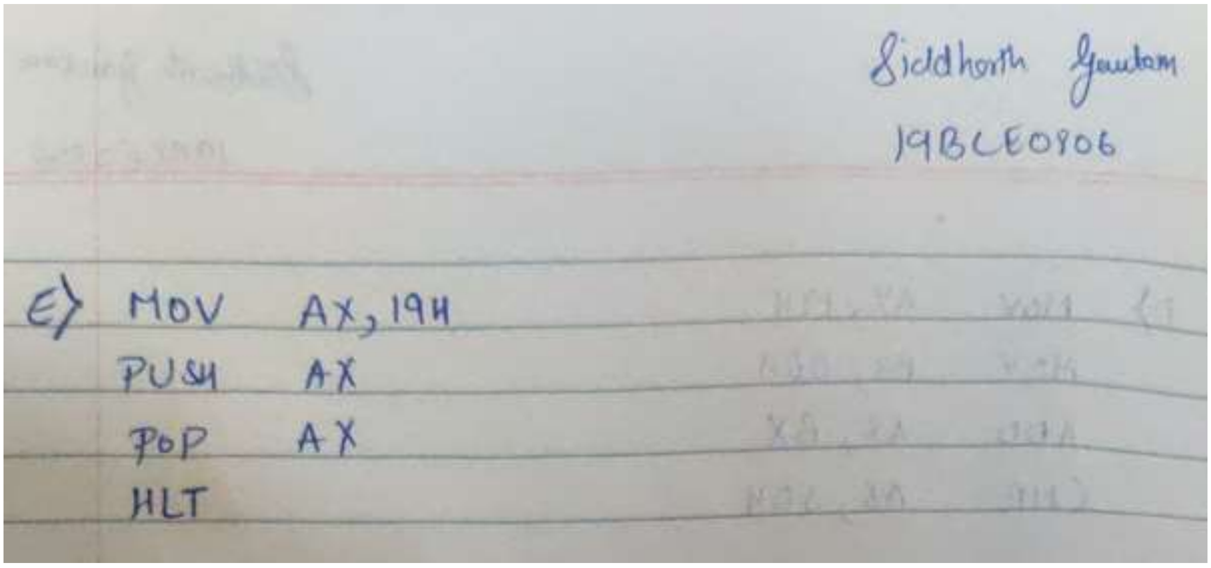


```
MOV AX,170
MOV BX,060
ADD AX,BX
CMP AX,500

JL label1
JG label2
JE label3

label1:
INC AX
JMP fin

label2:
DEC AX
JMP fin

label3:
DEC AX
JMP fin

fin:
END
```

E)

Push the Data 1 to stack and Pop it from stack.

CODE:

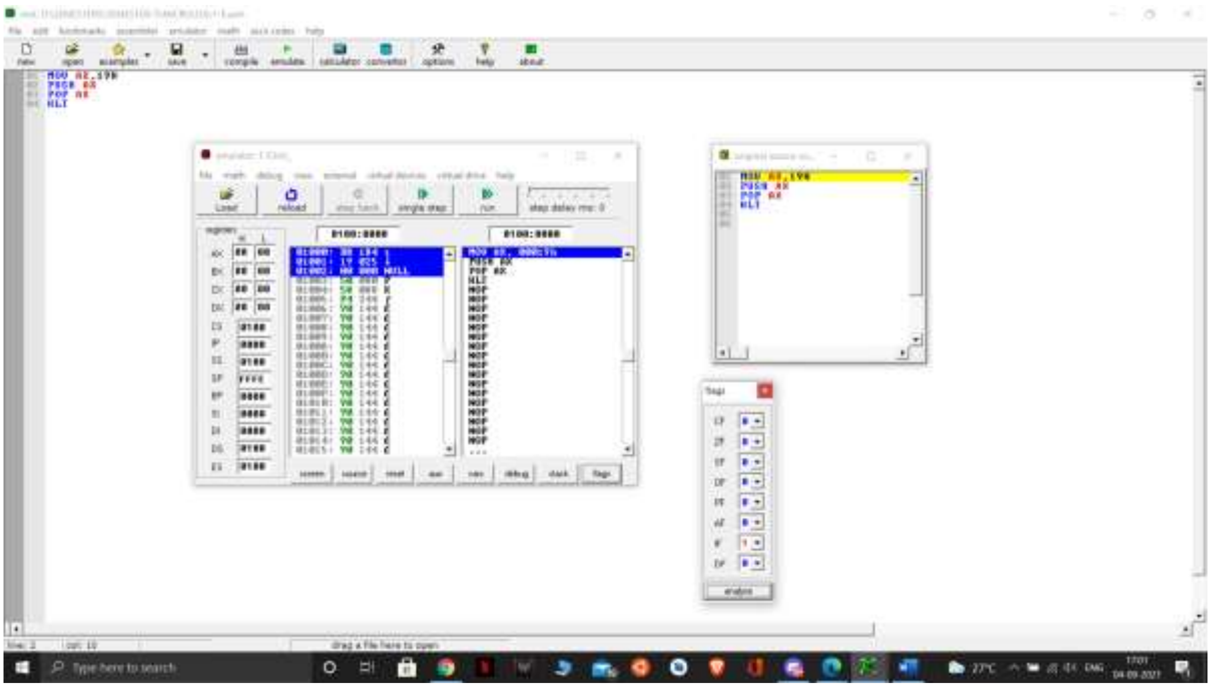


Siddharth Gautam
19BCE0806

```
E) MOV   AX, 19H
   PUSH  AX
   POP   AX
   HLT
```

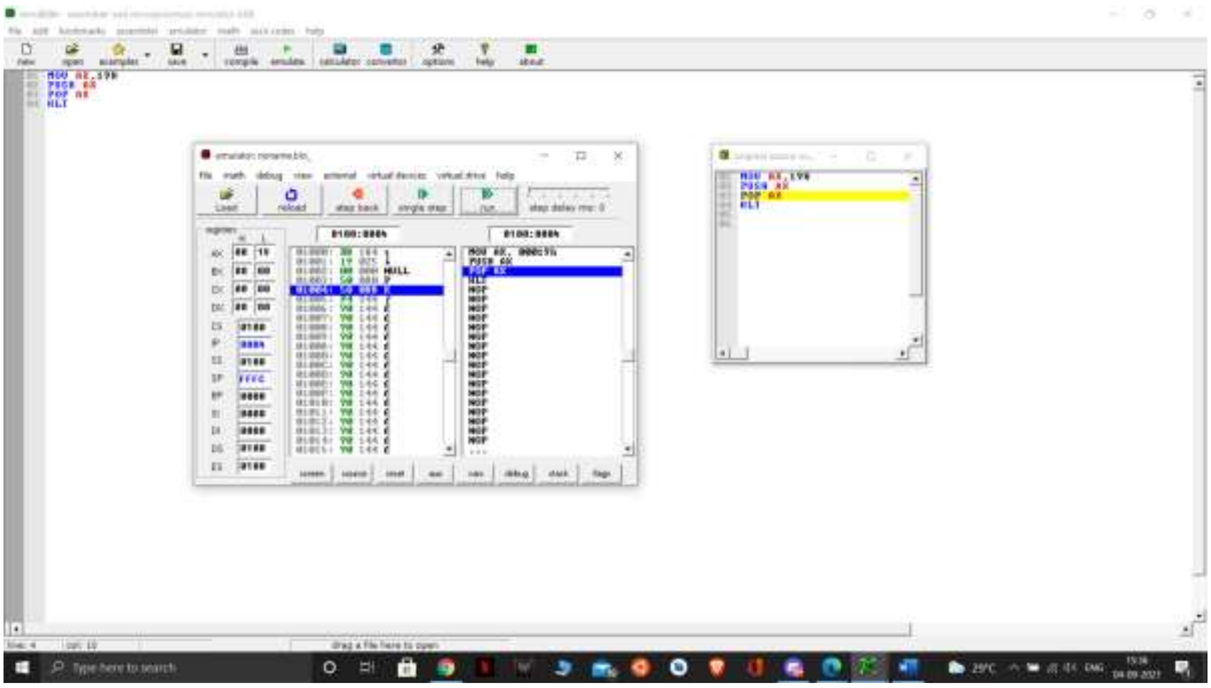
OUTPUT:

Originally the Stack Pointer (SP) is FFFE.



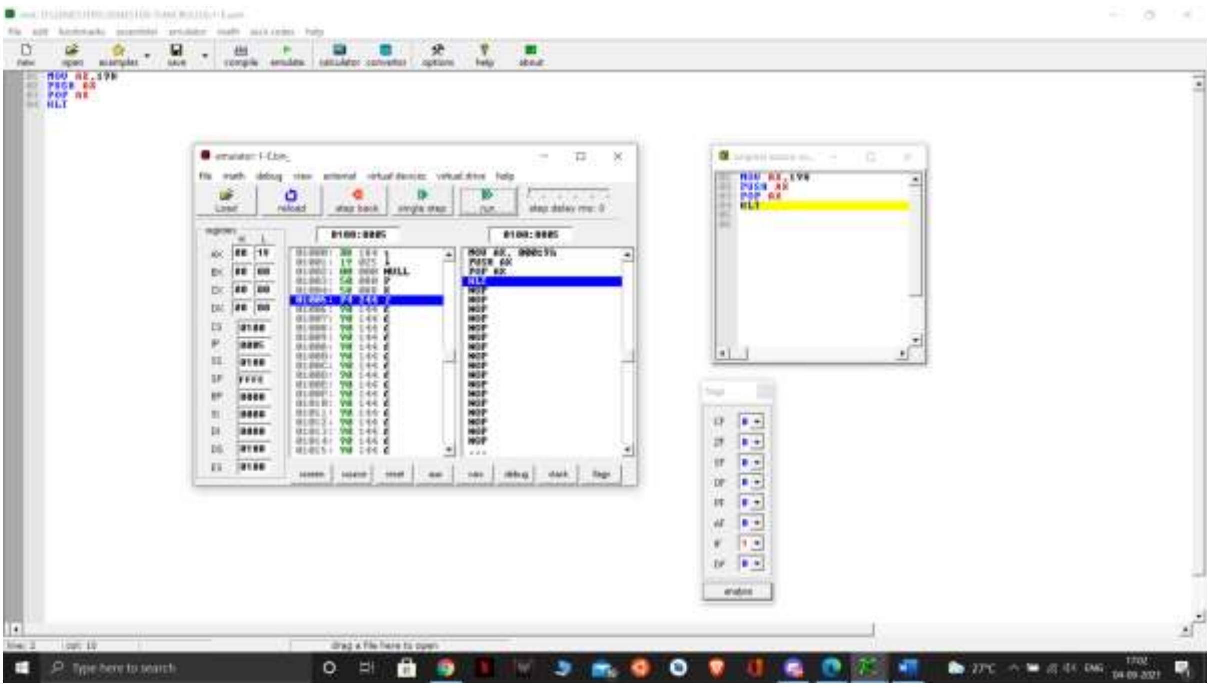
```
MOV AX,19H
PUSH AX
POP AX
HLT
```

After we push the data1(19H) value in the stack the, stack pointer changes to FFFC.



```
MOV AX,19H
PUSH AX
POP AX
HLT
```

When we pop the data1 value from the stack the stack pointer returns back to its original state i.e. FFFE.



```
MOV AX,19H
PUSH AX
POP AX
HLT
```

```
0100:0005

01000: B8 184 ╕
01001: 19 025 ↓
01002: 00 000 NULL
01003: 50 080 P
01004: 58 088 X
01005: F4 244 ⌠
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0005

MOV AX, 00019h
PUSH AX
POP AX
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```
registers
     H   L
AX  00  19
BX  00  00
CX  00  00
DX  00  00
CS  0100
IP  0005
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

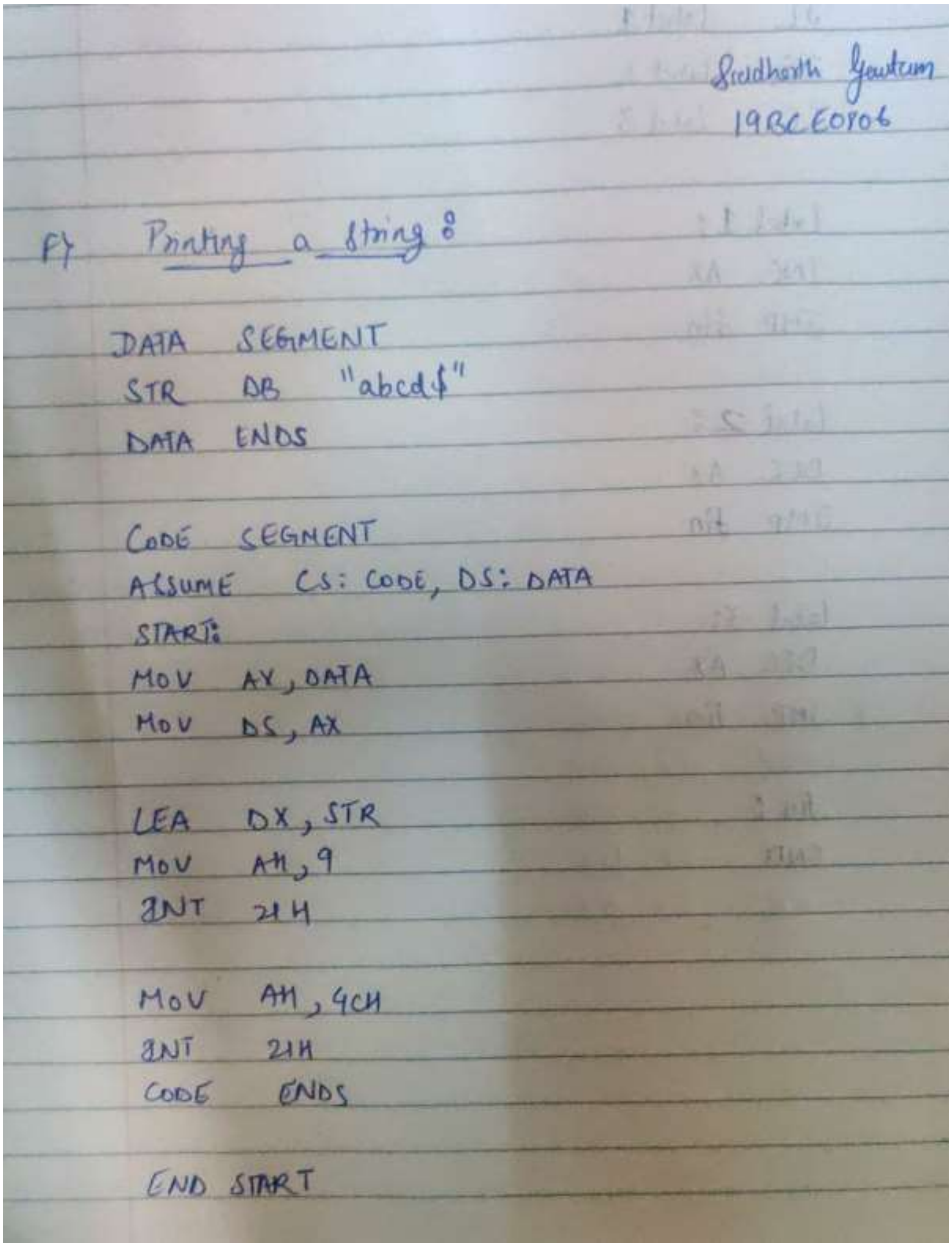
```
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: 1-Ebin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
analyse
drag a file here to open
Type here to search
27°C ENG
17:02
04-09-2021

F)

Perform any two string operations on four letter word (ABCD/abcd)

CODE:



Sidharth Gautam
19BCE0806

F) Printing a String :

```
DATA  SEGMENT
STR   DB   "abcd$"
DATA  ENDS

CODE  SEGMENT
ASSUME   CS: CODE, DS: DATA
START:
MOV   AX, DATA
MOV   DS, AX

LEA   DX, STR
MOV   AH, 9
INT   21H

MOV   AH, 4CH
INT   21H
CODE   ENDS

END START
```

Siddharth
Gautam
19BCE0806

Compare two string :

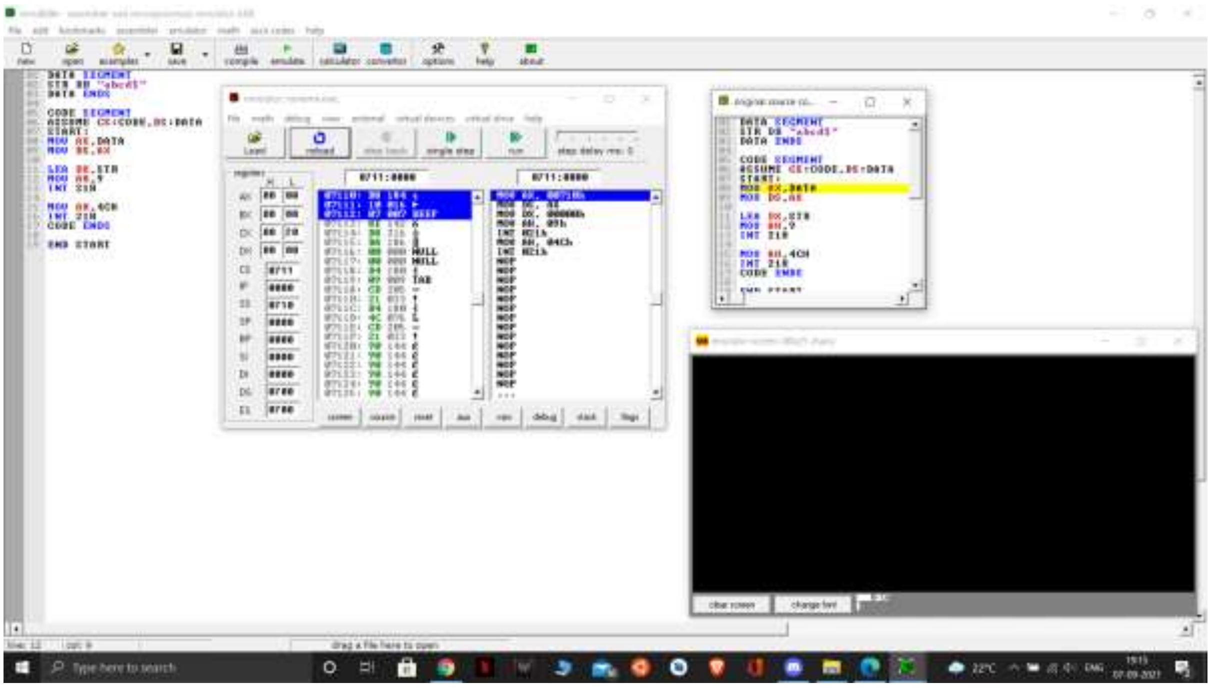
```
Data Segment
    str1 db "ABCD$"
    strlen1 db $-str1
    str2 db "ABCD$"
    strlen2 db $-str2
    streq db "Strings are Equal $"
    struneq db "Strings are Unequal$"
Data Ends
Code Segment
        Assume cs: code , ds: data
        Begin :
            Mov ax, data
            Mov ds, ax
            Mov es, ax
            lea si, str1
            lea di, str2
            Mov Cx, 6
            Mov Al, strlen1
            Mov bl, strlen2
            CMP Al, Bl
            jne Not_Equal
            repe Cmpsb
            jne Not_Equal
            jmp Equal
        Not_Equal :
            mov ah, 09h
            lea dx, struneq
            int 21H
            Jmp Exit
```

Siddharth Gautam
19BLE0R06

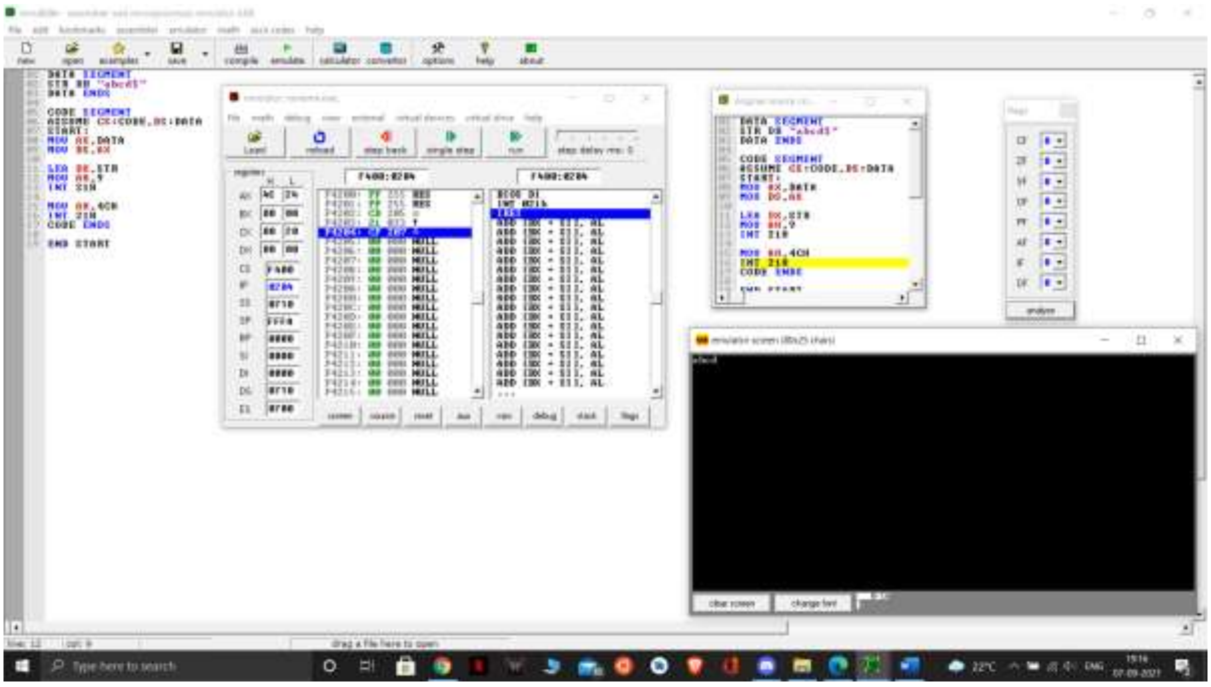
```
Equal:
    Mov ah, 09H
    lea dx, strey
    Int 21H
Exit:
    MOV AX, 4C00h
    int 21H.
Code Ends
End Begin
```

OUTPUT:

Printing a string:



```
DATA SEGMENT
STR DB "abcd$"
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
MOV AX,DATA
MOV DS,AX

LEA DX,STR
MOV AH,9
INT 21H

MOV AH,4CH
INT 21H
CODE ENDS

END START
```

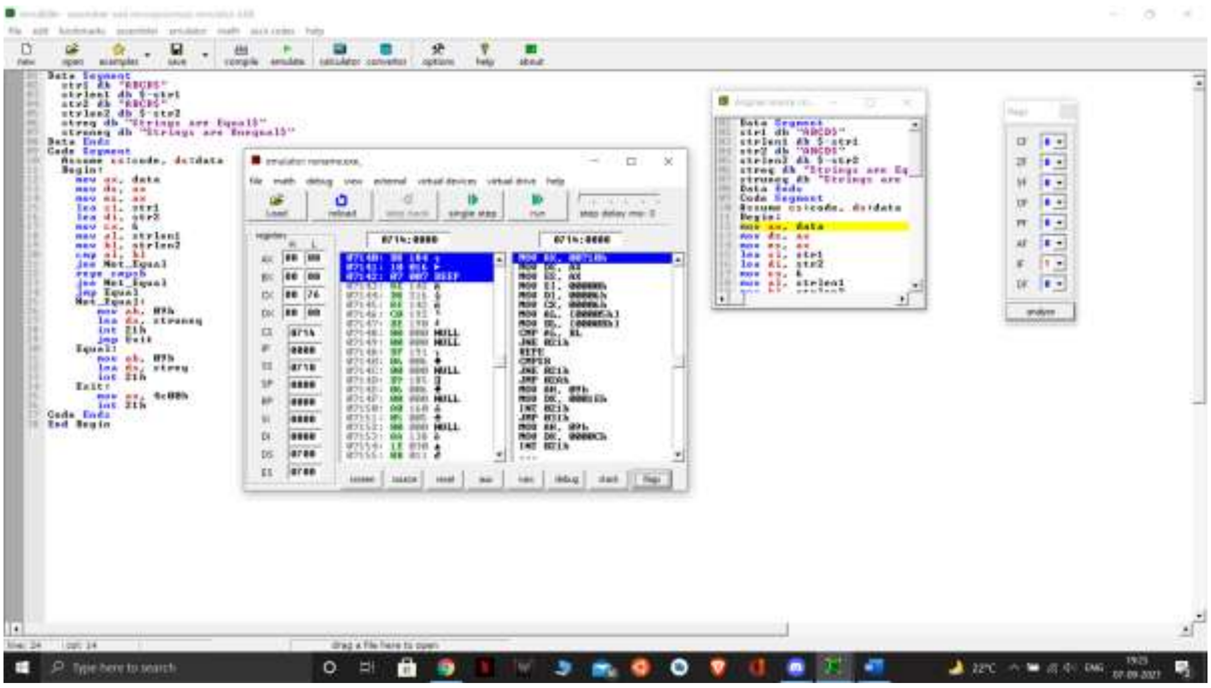


```
DATA SEGMENT
STR DB "abcd$"
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
MOV AX,DATA
MOV DS,AX

LEA DX,STR
MOV AH,9
INT 21H

MOV AH,4CH
INT 21H
CODE ENDS

END START
```

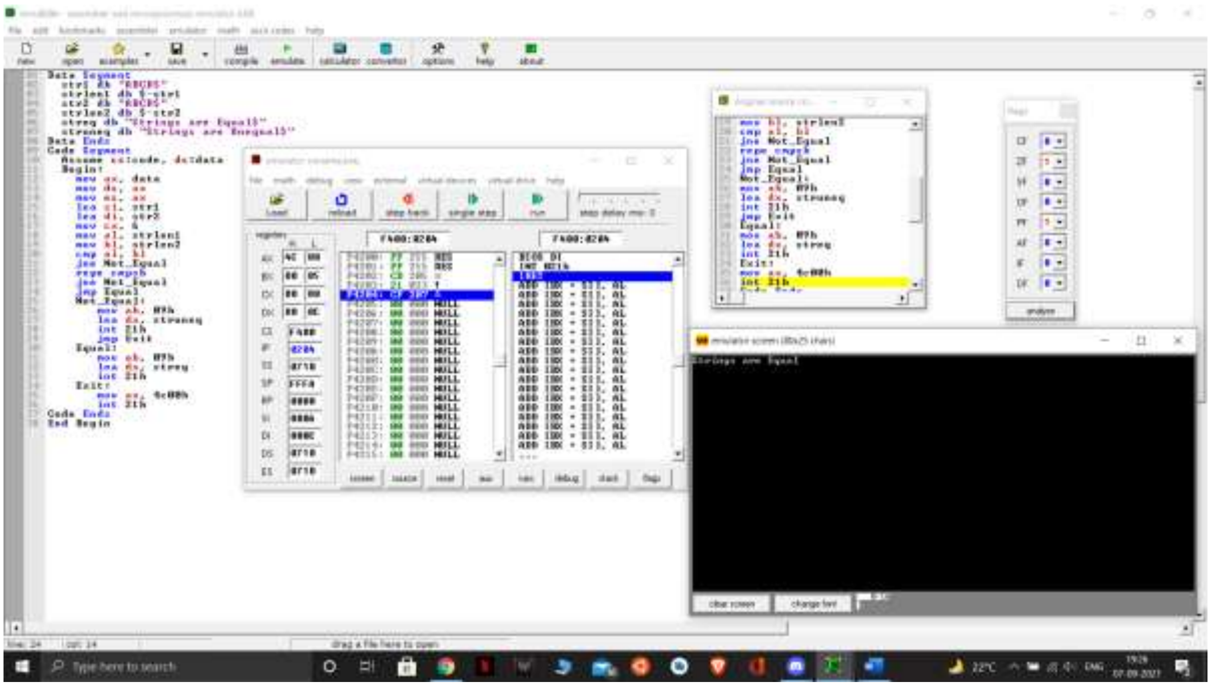
```
abcd
```

Compare two strings:

When strings are equal

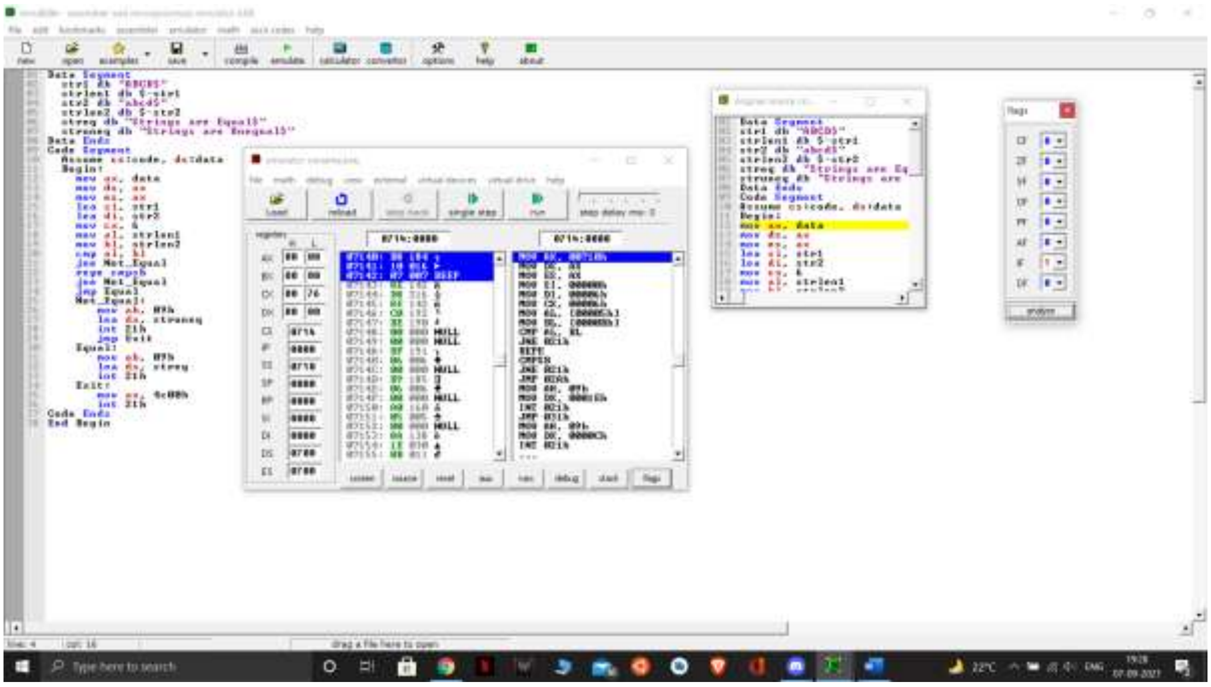


```
Data Segment
  str1 db "ABCDE"
  strlen1 db $-str1
  str2 db "ABCDE"
  strlen2 db $-str2
  streq db "Strings are Equal$"
  strneq db "Strings are Unequal$"
Data Ends
Code Segment
  Assume cs:code, ds:data
  Begin:
    mov ax, data
    mov ds, ax
    mov es, ax
    lea si, str1
    lea di, str2
    mov cx, 6
    mov al, strlen1
    mov bl, strlen2
    cmp al, bl
    jne Not_Equal
    repe cmpsb
    jne Not_Equal
    jmp Equal
    Not_Equal:
       mov ah, 09h
       lea dx, strneq
       int 21h
       jmp Exit
    Equal:
       mov ah, 09h
       lea dx, streq
       int 21h
    Exit:
       mov ax, 4c00h
       int 21h
Code Ends
End Begin
```

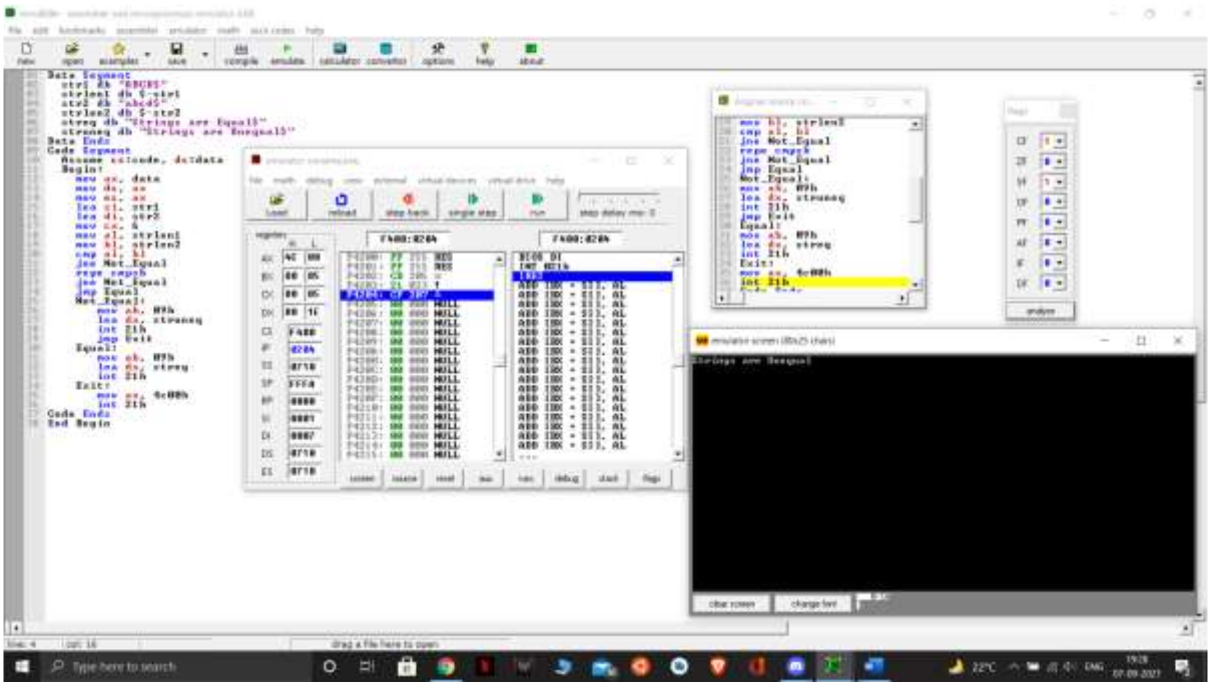


```
Strings are Equal
```

When strings are not equal



```
Data Segment
   str1 db "ABCD$"
   strlen1 db $-str1
   str2 db "abcd$"
   strlen2 db $-str2
   streq db "Strings are Equal$"
   strneq db "Strings are Unequal$"
Data Ends
Code Segment
   Assume cs:code, ds:data
   Begin:
      mov ax, data
      mov ds, ax
      mov es, ax
      lea si, str1
      lea di, str2
      mov cx, 4
      mov al, strlen1
      mov bl, strlen2
      cmp al, bl
      jne Not_Equal
      repe cmpsb
      jne Not_Equal
      jmp Equal
      Not_Equal:
         mov ah, 09h
         lea dx, strneq
         int 21h
         jmp Exit
      Equal:
         mov ah, 09h
         lea dx, streq
         int 21h
      Exit:
         mov ax, 4c00h
         int 21h
Code Ends
End Begin
```



```
Strings are Unequal
```