**Microprocessor Lab Record**

**BY**

**SRUTI VERMA-18BCE1005**

A lab report submitted to

**Prof. Dr. Prathiba A**

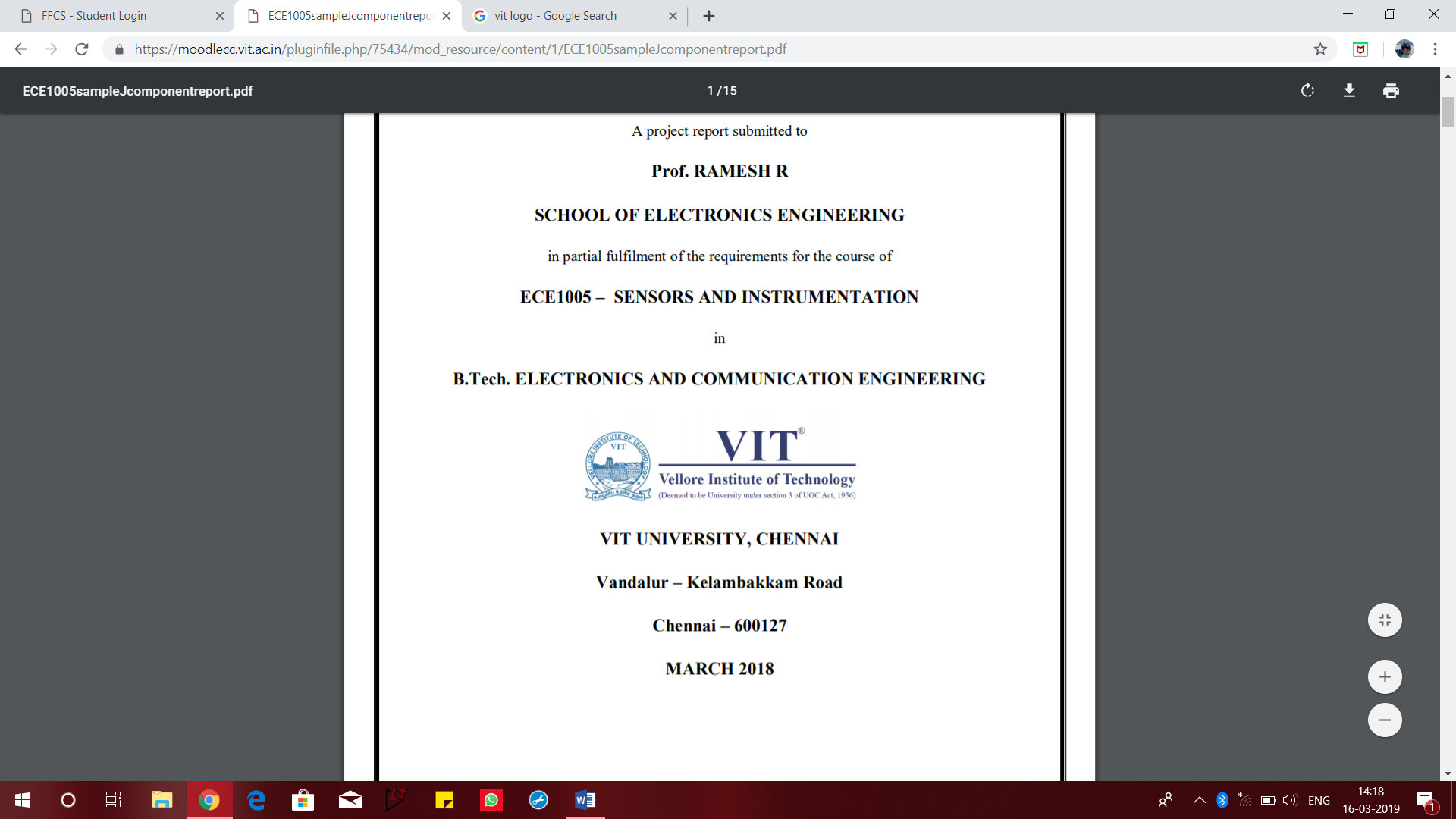
**SCHOOL OF ELECTRONICS ENGINEERING**

In partial fulfillment of the requirements for the course of

**CSE2005-MICROPROCESSOR AND INTERFACING**

IN

**B.tech COMPUTER SCIENCE ENGINEERING**



**VIT UNIVERSITY, CHENNAI**

**VANDALUR KELAMBAKAM ROAD**

**CHENNAI-600127**

**MARCH-2020**

**LAB – 1**

**Aim:**

To understand and analyze different addressing modes

**Software used:** MASM

**Code:**

code segment

assume cs:code

start:

mov ax,1234h

mov dx,ax

mov si,1000h

mov al,[si]

mov bl,[si+1000h]

mov bx,1000h

mov ax,[si+bx];

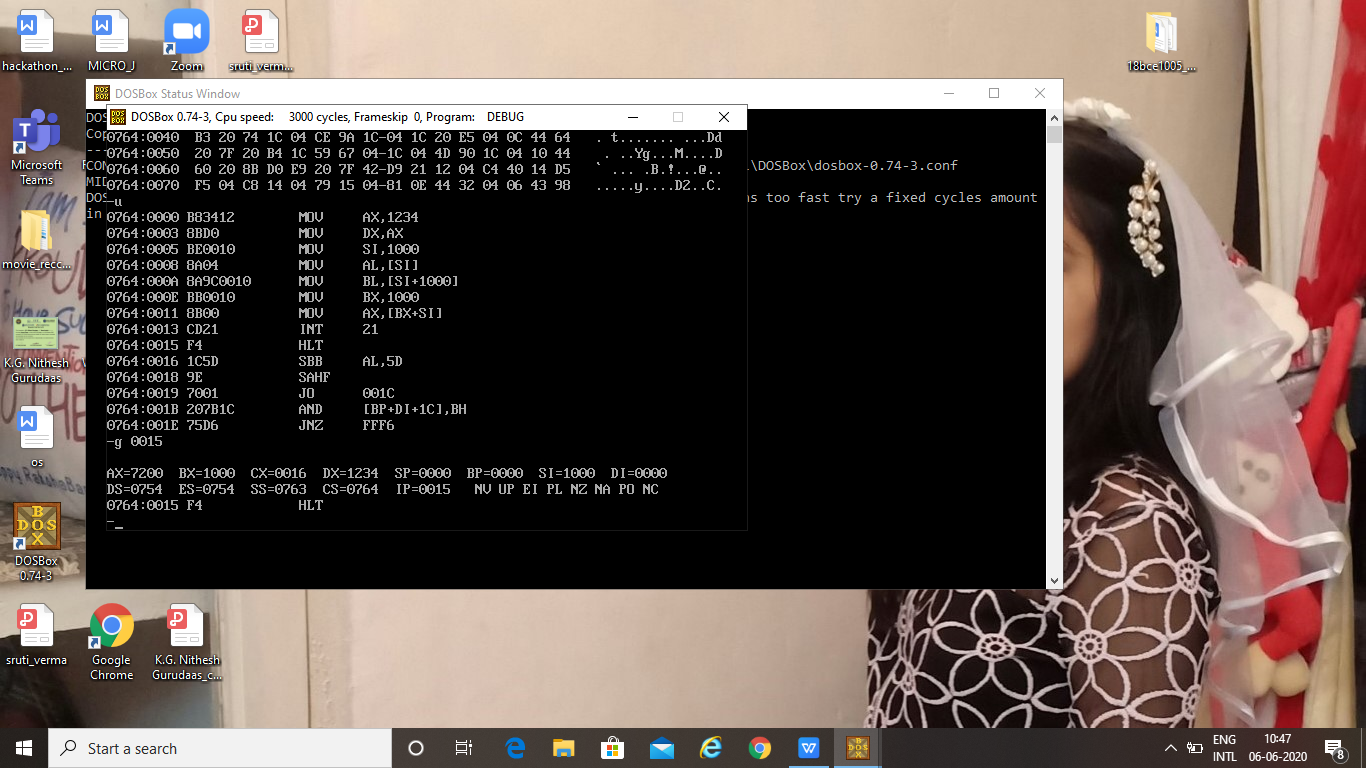
int 21h

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected



**Aim:**

To add two numbers and verify the result

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

opr1 dw 1234h

opr2 dw 0002h

result dw 01 dup(?)

data ends

code segment

start:

mov ax,data

mov ds,ax

mov ax,opr1

mov bx,opr2

clc

add ax,bx

mov di,offset result

mov [di],ax

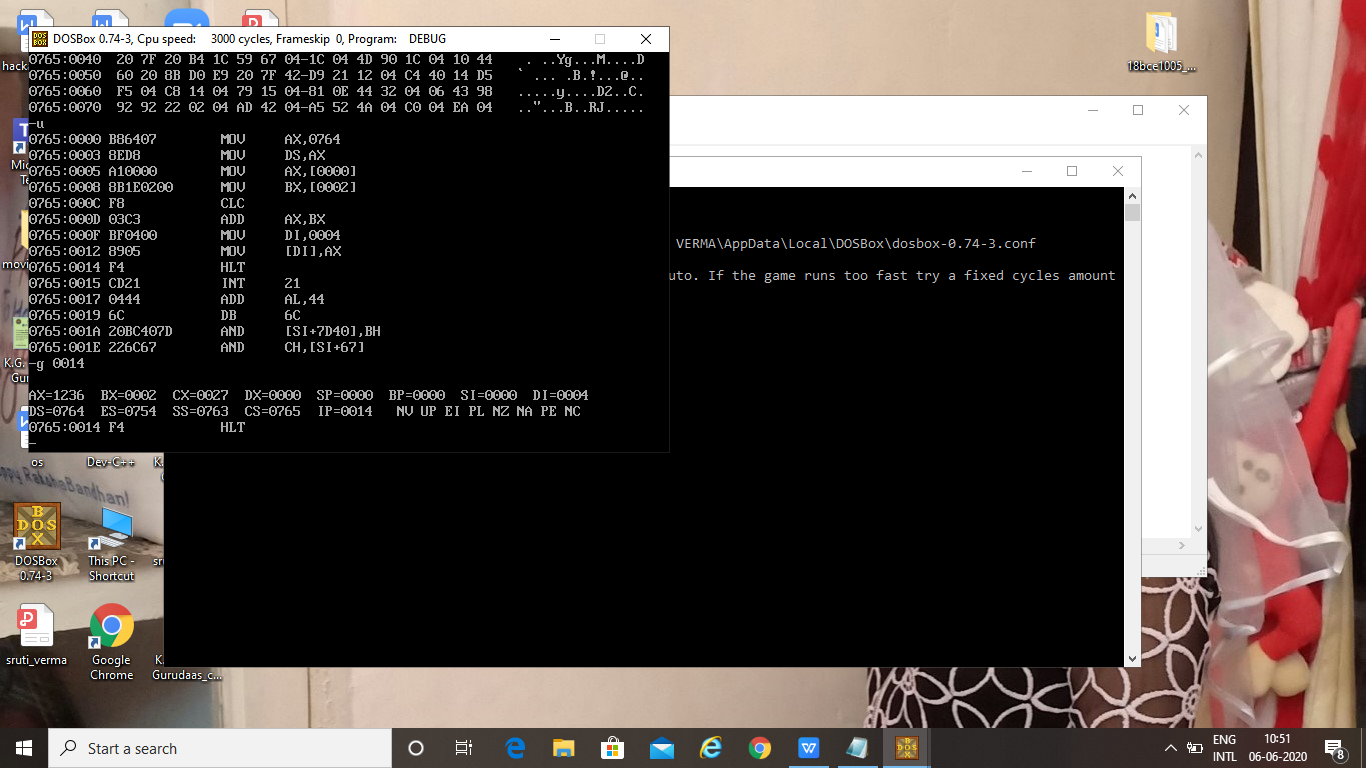
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**LAB - 2**



**Aim:**

To add two 16-bit numbers using 16-bit registers

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

opr1 dw 1101h

opr2 dw 1005h

result dw 01 dup(?)

data ends

code segment

start:

mov ax,data

mov ds,ax

mov ax,opr1

mov bx,opr2

add ax,bx

mov di,offset result

mov [di],ax

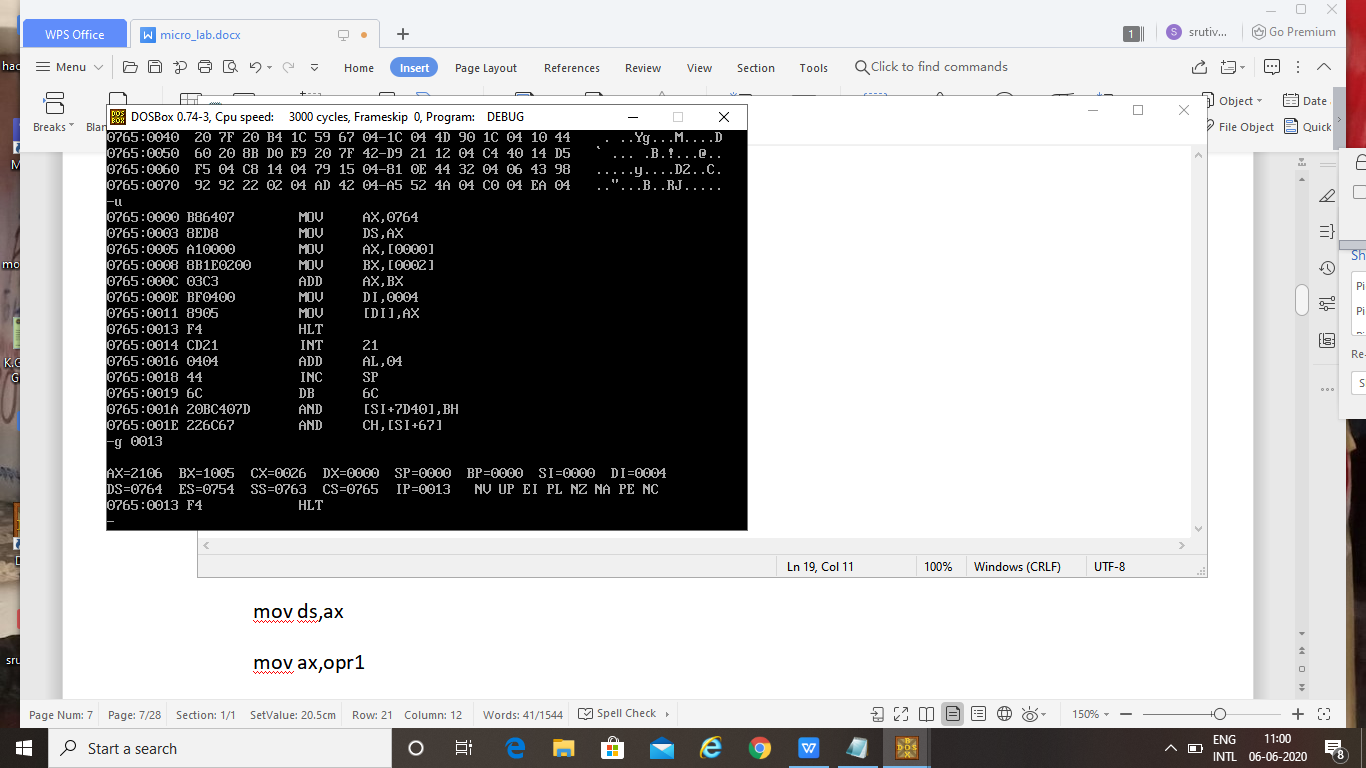
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected



**Aim:**

To add two 16-bit numbers using 8-bit registers

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

opr1 db 24h

opr2 db 15h

opr3 db 16h

opr4 db 24h

result dw 01 dup(?)

data ends

code segment

start:

mov ax,data

mov ds,ax

mov ah,opr1

mov al,opr2

mov bh,opr3

mov bl,opr4

add al,bl

adc ah,bh

mov di,offset result

mov [di],ax

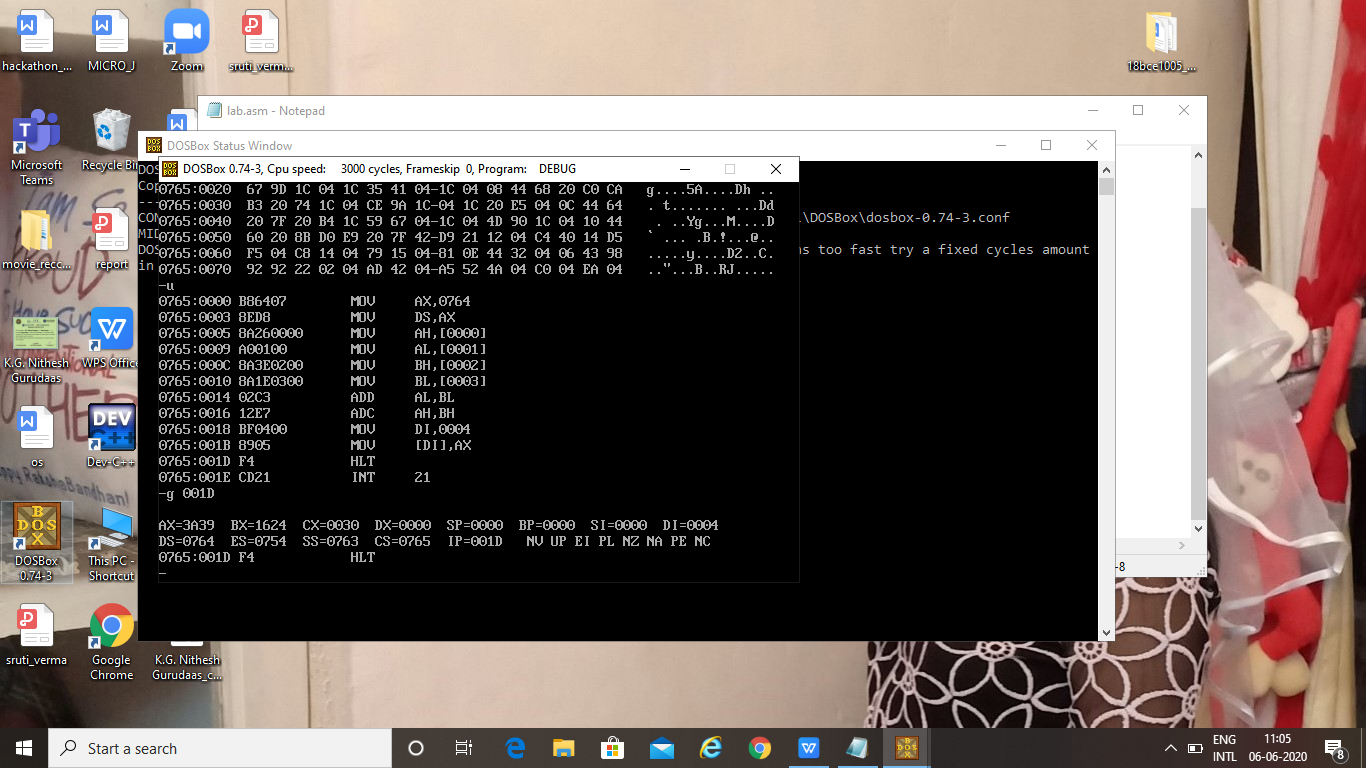
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected



**Aim:**

To multiply two numbers – 0FFFh and 0FFEh

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

opr1 dw 0FFFh

opr2 dw 0FFEh

result dw 01 dup(?)

data ends

code segment

start:

mov ax,data

mov ds,ax

mov ax,opr1

mov bx,opr2

mul bx

mov di,offset result

mov [di],ax

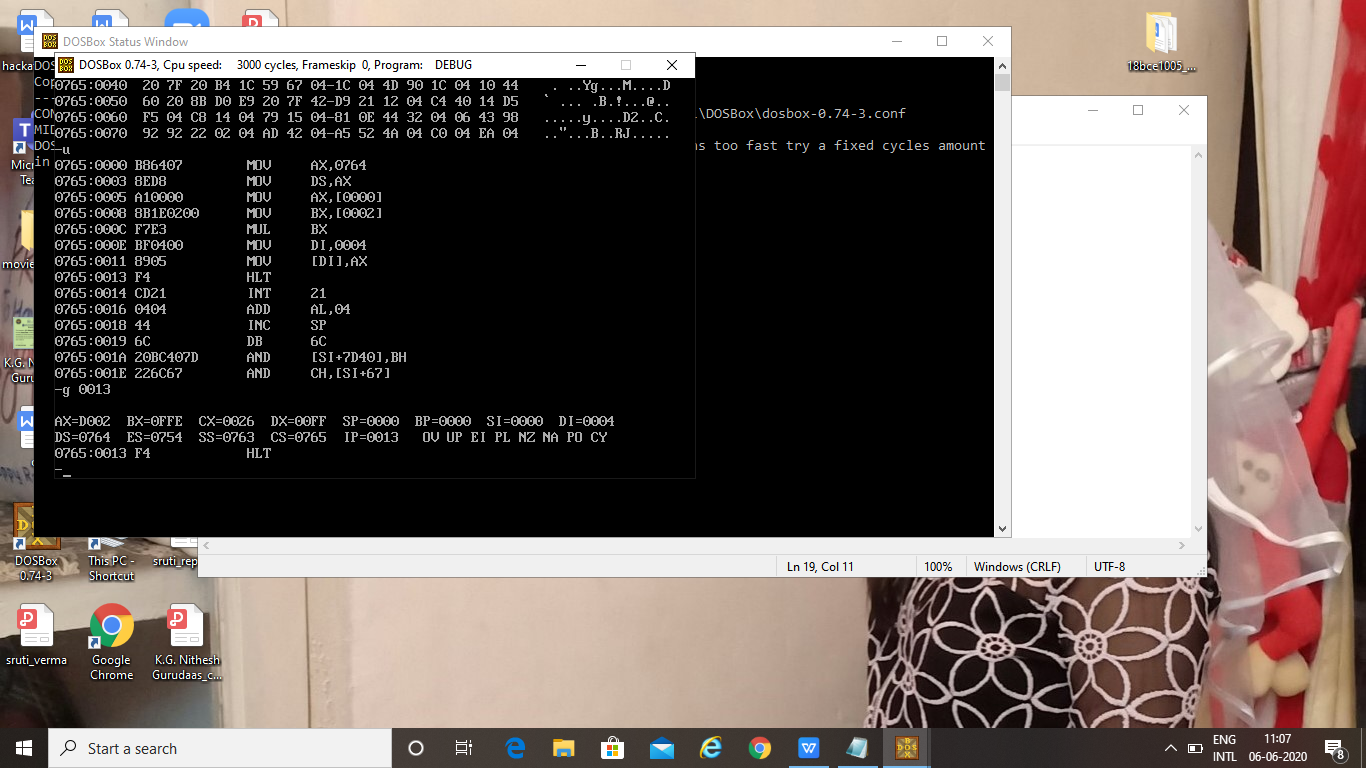
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**LAB – 3**



**Aim:**

To write 8086 ALP to find summation of two arrays

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

mat1 dw 0022h,0011h,0020h,0033h,0016h,0011h,0013h

mat2 dw 0020h,0013h,0010h,0023h,0015h,0042h,0031h

resmat dw 7 dup(0)

data ends

code segment

start:

mov cx,07h

mov bx,cx

mov ax,data

mov ds,ax

mov ax,00h

rpt:

add ax,mat1[SI]

add ax,mat2[SI]

mov resmat[SI],ax

mov ax,00h

add SI,02h

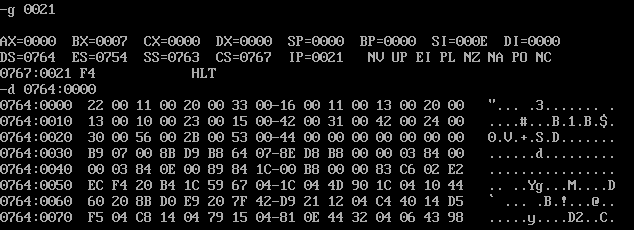
loop rpt

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**ii)**

**Aim:**

To write 8086 ALP to find product of two arrays

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

mat1 dw 0022h,0011h,0020h,0033h,0016h,0011h,0013h

mat2 dw 0020h,0013h,0010h,0023h,0015h,0042h,0031h

resmat dw 7 dup(0)

data ends

code segment

start:

mov cx,07h

mov bx,cx

mov ax,data

mov ds,ax

mov ax,00h

rpt:

add ax,mat1[SI]

mul mat2[SI]

mov resmat[SI],ax

mov ax,00h

add SI,02h

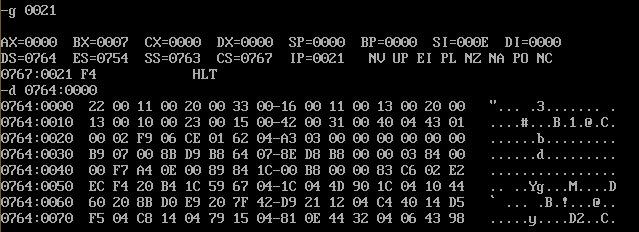
loop rpt

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**iii)**

**Aim:**

To write ALP to generate Fibonacci series

**Software used:** MASM

**Code:**

assume cs:code

code segment

start:

mov al,00h

mov si,500h

mov [si],al

add si,01h

add al,01h

mov [si],al

mov cx,0005h

l1:

mov al,[si-1]

add al,[si]

add si,01h

mov [si],al

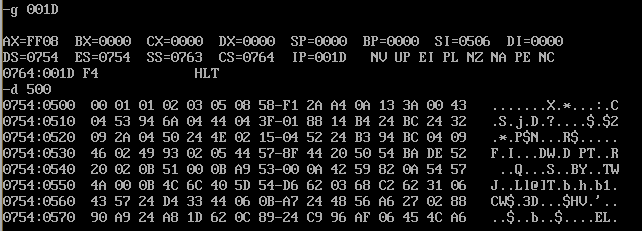
loop l1

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**LAB – 4**



**Aim:**

Writing 8086 ALP to find summation of 2 arrays

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

arr db 05h,01h,03h,02h,04h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov bx,04h

back :mov cx,bx

lea si,arr

l2 :mov al,[si]

cmp al,[si+1]

mov [si],al

l1 :inc si

loop l2

dec bx

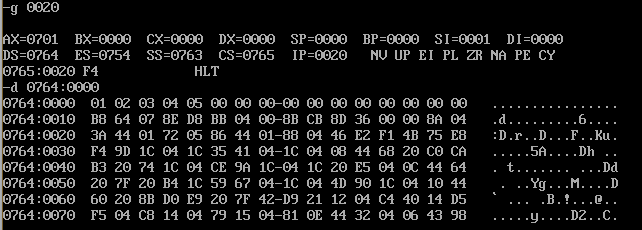
jnz back

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as desired.



**Aim:**

To write an 8086 ALP to sort an array in descending order.

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

arr db 05h,01h,03h,02h,04h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov bx,04h

back :mov cx,bx

lea si,arr

l2 :mov al,[si]

cmp al,[si+1]

jnc l1

xchg al,[si+1]

mov [si],al

l1 :inc si

loop l2

dec bx

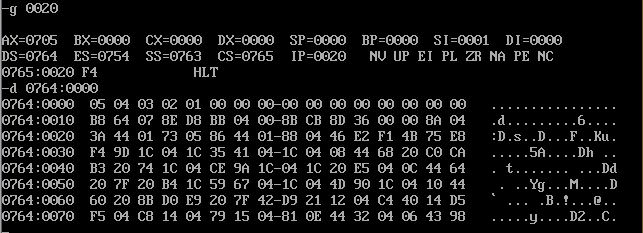
jnz back

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**iii)**

**Aim:**

To write 8086 ALP to find the largest value in an array

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

arr db 23h,12h,45h,65h,0ABh

lrg db ?

data ends

code segment

start:

mov ax,data

mov ds,ax

mov cx,04h

lea si,arr

mov al,[si]

next:

cmp al,[si+1]

jnc back

mov al,[si+1]

back:

inc si

loop next

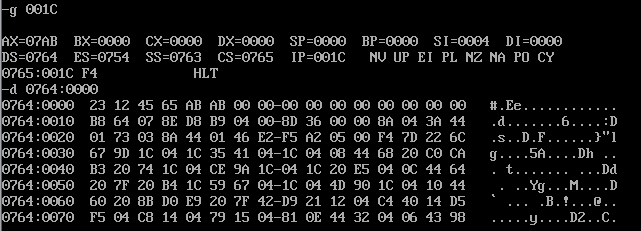
mov lrg,al

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected



**Aim:**

To write ALP to find the smallest value in an array

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

arr db 23h,12h,45h,65h,0ABh

small db ?

data ends

code segment

start:

mov ax,data

mov ds,ax

mov cx,04h

lea si,arr

mov al,[si]

next:

cmp al,[si+1]

jc back

mov al,[si+1]

back:

inc si

loop next

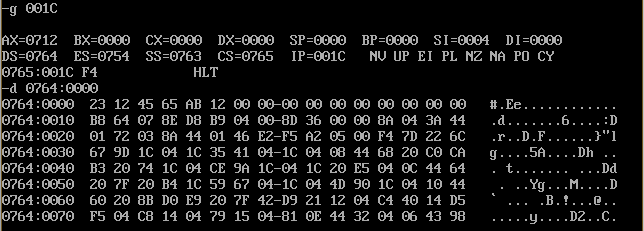
mov small,al

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**LAB – 5**

**Aim:**

To convert given BCD number to hexadecimal

**Software Used:** MASM

**Code:**

assume cs:code, ds:data

data segment

bcd db 28h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov bl,bcd

mov al,bcd

and bl,0Fh

and al,0F0h

mov cl,04h

ror al,cl

mov dl,0Ah

mul dl

add bl,al

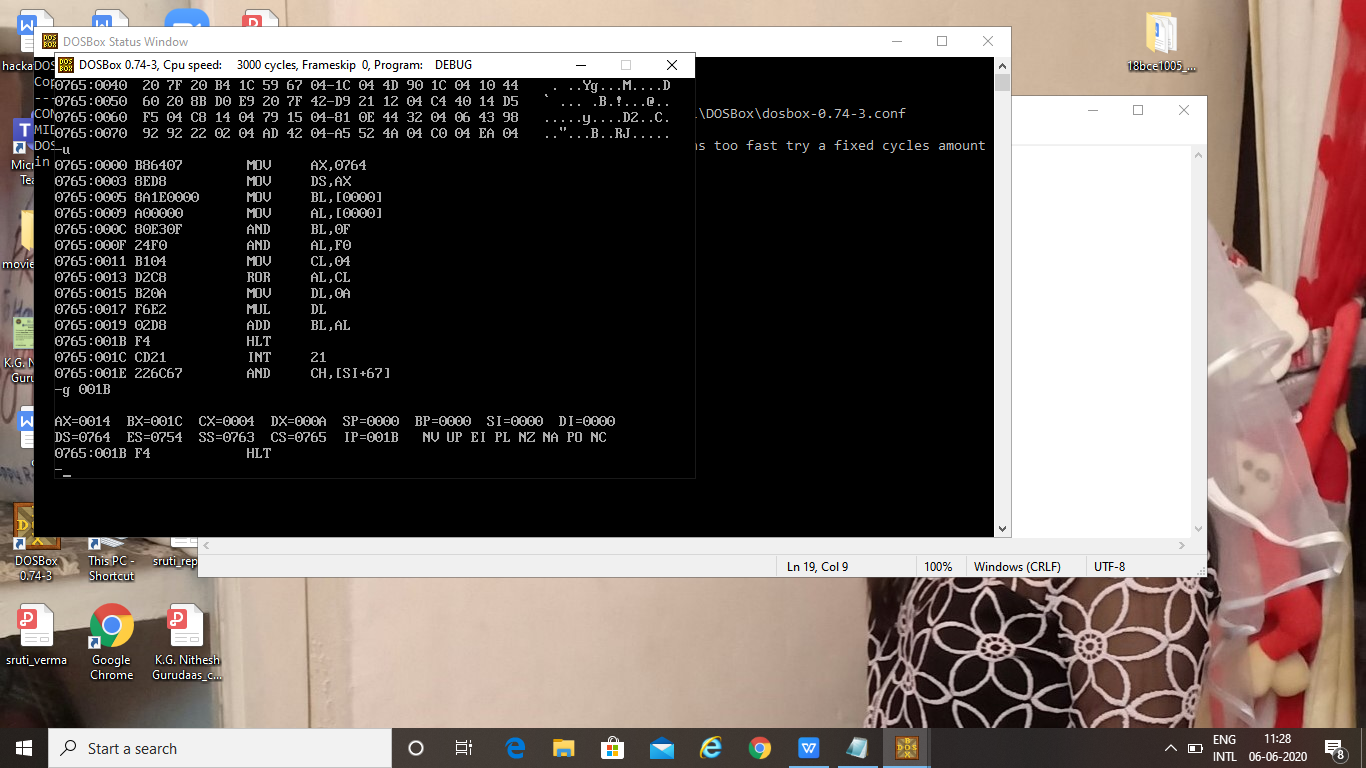
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected.

**Aim:**

To add two matrices and store it in another matrix.

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

arr1 db 1h,2h,3h,0h,1h,2h,3h,0h,1h

arr2 db 1h,2h,3h,0h,1h,2h,3h,0h,1h

arr3 db 9 dup(0)

data ends

code segment

start:

mov ax,data

mov ds,ax

mov cx,09h

lea si,arr1

lea di,arr2

L1 :mov bl,[si]

mov al,[di]

add al,bl

mov [si],al

inc si

inc di

loop L1

lea di,arr3

lea si,arr1

L2 :mov bl,[si]

mov [di],bl

inc si

inc di

loop L2

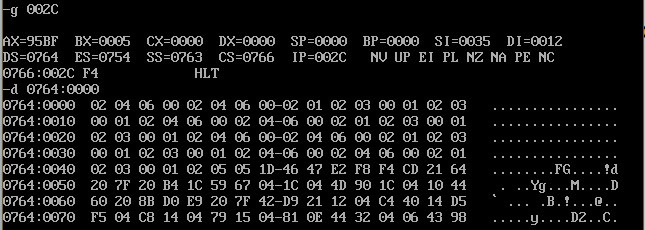
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected.

**LAB – 6**

**Aim:**

To calculate 2’s complement of a number

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

var1 db 24h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov ax,0000h

mov al,var1

not al

mov bl,al

adc al,00000001b

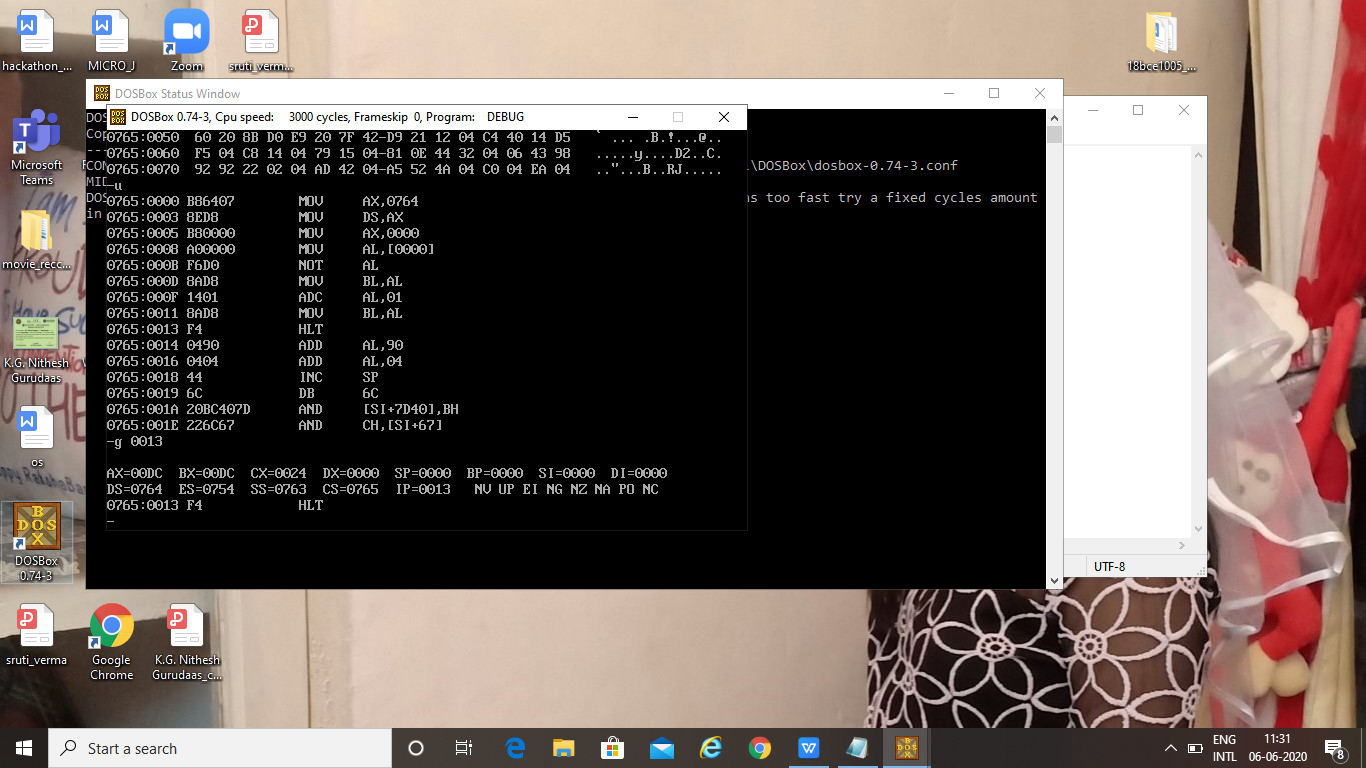
mov bl,al

hlt

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected



**Aim:**

To find permutation using the permutation formula:

nPr = n!/(n-r)!

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

num db 05h

re db 02h

num1 db 0h

den db 0h

data ends

code segment

fact proc near

cmp cl,0h

je return

mul cl

dec cl

call fact

return :ret

fact endp

start:

mov ax,data

mov ds,ax

mov ax,0000h

mov al,num

mov dl,num

sub dl,01h

mov bl,re

mov cl,dl

call fact

mov num1,al

mov al,num

sub al,bl

mov dl,al

sub dl,01h

mov cl,dl

call fact

mov bl,al

mov al,num1

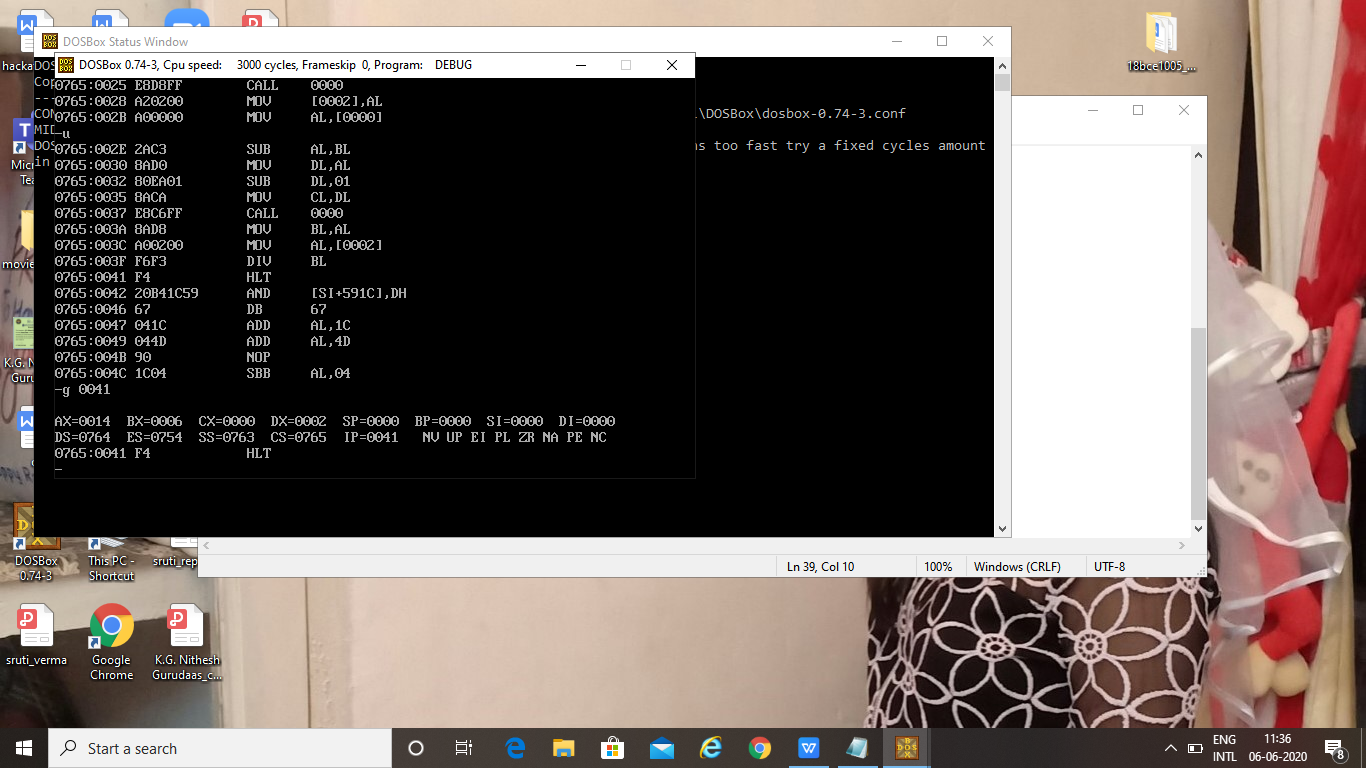
div bl

hlt

code ends

end start

**Output:**



**Result:**

The code is executed and result is obtained as expected.



**Aim:**

To find combination using the formula:

nCr = n!   
 (r! (n-r)!)

**Software used:** MASM

**Code:**

assume cs:code,ds:data

data segment

num db 05h

re db 02h

num1 db 0h

den db 0h

data ends

code segment

fact proc near

cmp cl,0h

je return

mul cl

dec cl

call fact

return :ret

fact endp

start:

mov ax,data

mov ds,ax

mov ax,0000h

mov al,num

mov dl,al

sub dl,01h

mov cl,dl

call fact

mov num1,al

mov ax,0000h

mov al,re

mov dl,al

sub dl,01h

mov cl,dl

call fact

mov den,al

mov ax,0000h

mov al,num

mov bl,re

sub al,bl

mov dl,al

sub dl,01h

mov cl,dl

call fact

mul den

mov bl,al

mov al,num1

div bl

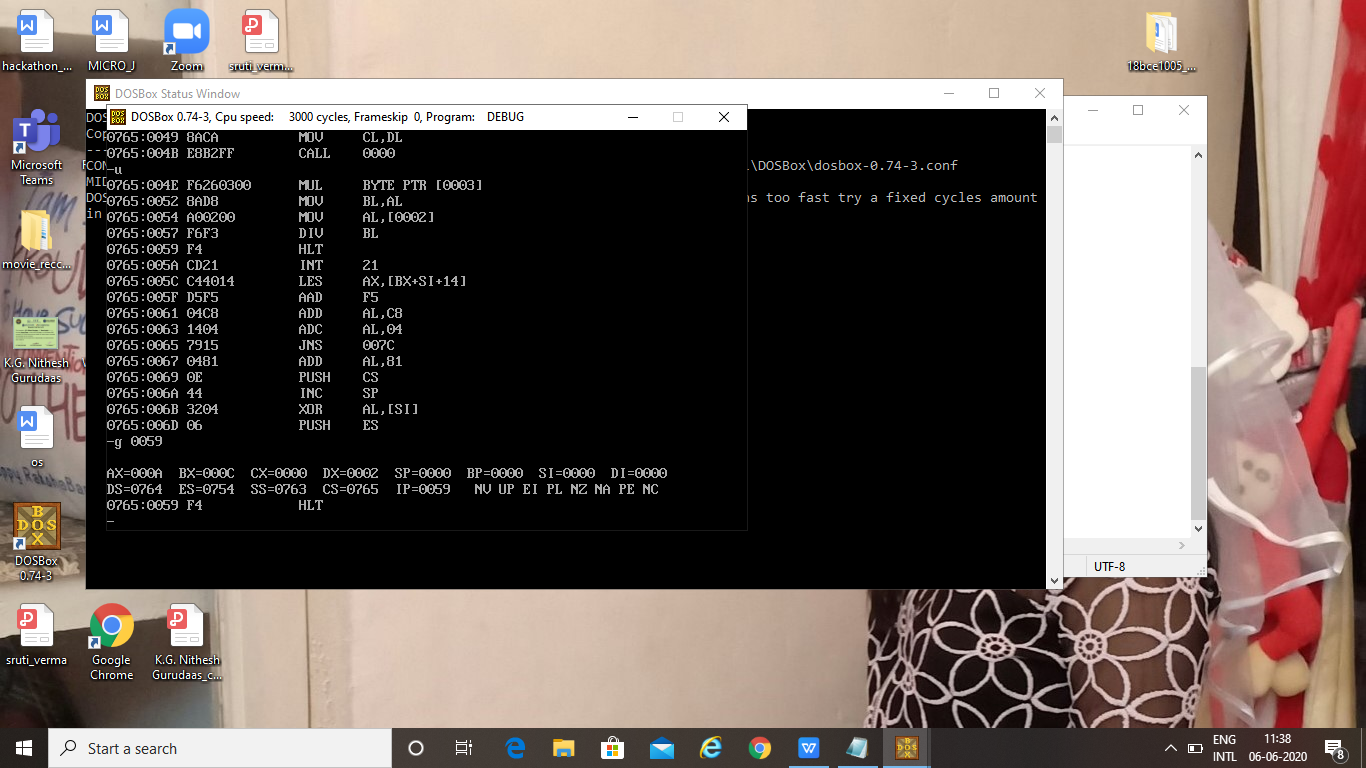
hlt

int 21h

code ends

end start

**Output:**



**Result:**

Code is executed and output is obtained as expected

**LAB – 7**

**Aim:**

To rotate stepper motor in: a) Clockwise direction b) Anti-clockwise direction

**Hardware used:**

8086 kit, Stepper motor interface, Stepper motor, Bus, Keyboard

**Code:**

Address:

1000 MOV CX, 04H

1004 MOV SI, 4200

1008 MOV AL, [SI]

100A OUT 0C0H, AL

100C MOV BX, 0FFFFH

1010 DEC BX

1011 JNZ 1010

1013 INC SI

1014 LOOP 1008

1016 JMP 1000

1019 HLT

a) For clockwise direction:

SB 4200 -> 09, 05, 06, 0A

b) For anti-clockwise direction:

SB 4200 -> 0A, 06, 05, 09

**Result:**

Code is executed and output is obtained as expected

**LAB – 8**

**Aim:**

To convert analog signal to digital using 8086 and ADC

**Hardware used:**

8086 kit, ADC, Bus, Keyboard

**Code:**

1000 MOV AL, 10H

1002 OUT 0C8H, AL

1004 MOV AL, 18H

1006 OUT 0C8H, AL

1008 HLT

**Result:**

Code is executed and result is obtained as expected.

**Aim:**

To convert analog signal to digital signal by sending SOC signal through 8086

**Hardware used:**

8086 kit, ADC, Bus, Keyboard

**Code:**

MOV AL, 00h

OUT Port C, AL

MOV AL, 01h

OUT Port C, AL

MOV AL, 00h

OUT Port C, AL

WAIT: IN AL, Port C

RCR

JNC WAIT

IN AL, Port A

HLT

**Result:**

Code is executed and output is obtained as expected.