

**LAB MANUEL**

**NAME: JAMAL AHMED**

**ID: CSC-22S-065**

**SUBJECT: COMPUTER ORGANIZATION AND**

**ASSEMBLY LANGUAGE**

**SECTION: 3B**

**SUBMITED TO: SAHAR ZAFAR**

**==============================================**

**Table of Contents**

[**INTRODUCTION** 2](#_Toc137600821)

[**BASIC DOS FUNCTIONS** 7](#_Toc137600822)

[**VARIBLE** 9](#_Toc137600823)

[**EXCHANGE KEYWORDS** 17](#_Toc137600824)

[**CASE CONVERSION** 21](#_Toc137600825)

[**LOOP** 29](#_Toc137600826)

[**JUMP** 36](#_Toc137600827)

[**COMPARISON** 38](#_Toc137600828)

[**STRINGS** 42](#_Toc137600829)

[**BIT MANUPULATION** 45](#_Toc137600830)

[**STACK, MACRO AND SUBROUTINES** 50](#_Toc137600831)

[**GRAPH STRAIGHT LINES** 56](#_Toc137600832)

[**GRAPHIC MODE (text & background)** 62](#_Toc137600833)

[**ARITHMATICS(MUL-DIV)** 65](#_Toc137600834)

[**FILE HANDLING** 69](#_Toc137600835)

**LAB 1**

# **INTRODUCTION**

Tasm + Note pad/ Notepad +

Save file with .asm in C drive tasm 1.4 folder

How to Run the program?

You need to follow three steps

Tasm filename.asm ; helps you to find error in the program

Tlink filename .obj ; Create object file for machine

Filename or file name.exe ; create executable file or show result on screen

Create program to print a character.

Segments

.model small

.stack 100h

.data

.code

Start:

Label used for starting the program code

Code for print a character

Mov ah, 2h

Mov dl,’N’

Int 21h

Code for Exit from the code

Mov ah, 4ch

Int 21h

End start

End the Label

**Task**

Create a program which print 0 on screen with the help of ASCII code.

Create a program which print A character with the help of ASCII code.

Create a program which print your name with the help of function 02.

Write all errors which you know about in today class and write how you correct them.

**TASK #1**

**Create a program which print 0 on screen with the help of ASCII code**

**Source code;-**

.model small

.stack 100h

.data

.code

start:

mov ah,02

mov dl,48

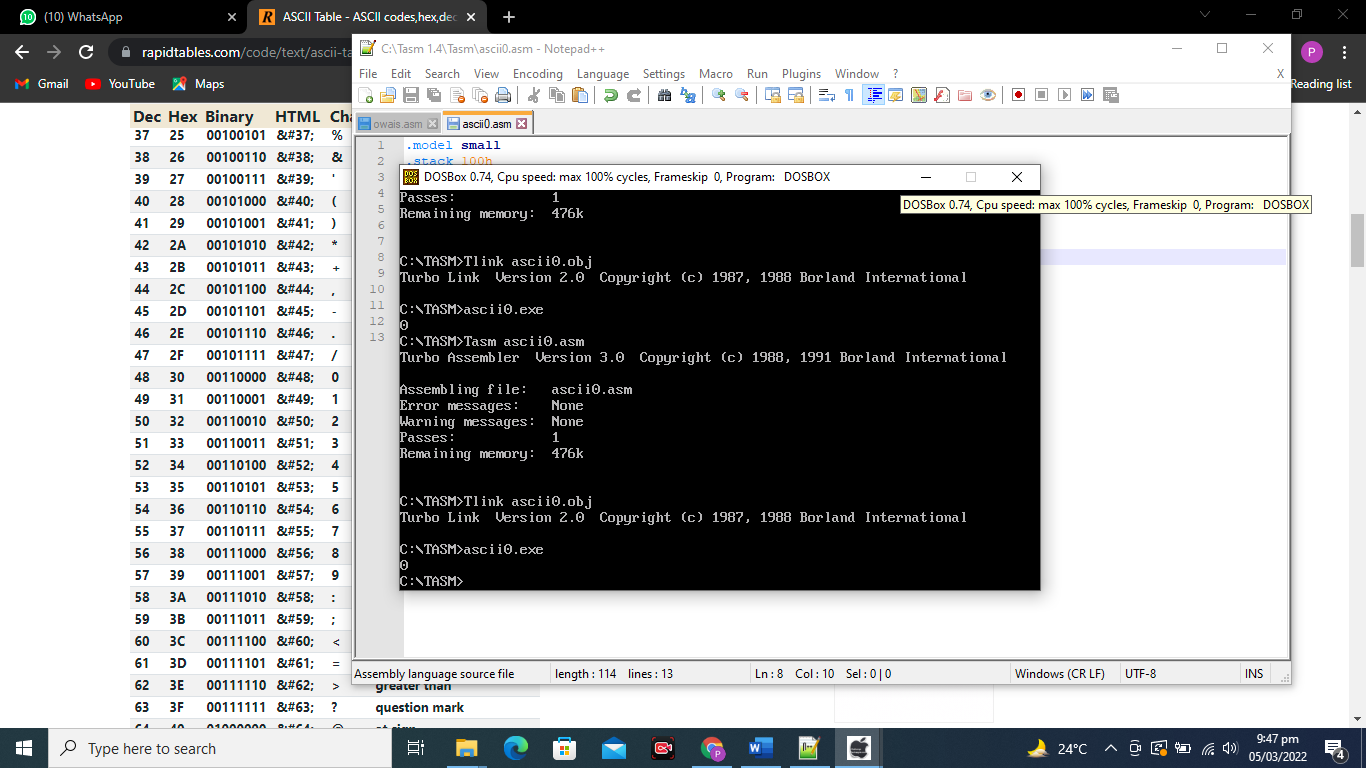
int 21h

mov ah,4ch

int 21h

end start

**OUT PUT;-**



**TASK#2**

**Create a program which print A character with the help of ASCII code.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov ah,02

mov dl,65

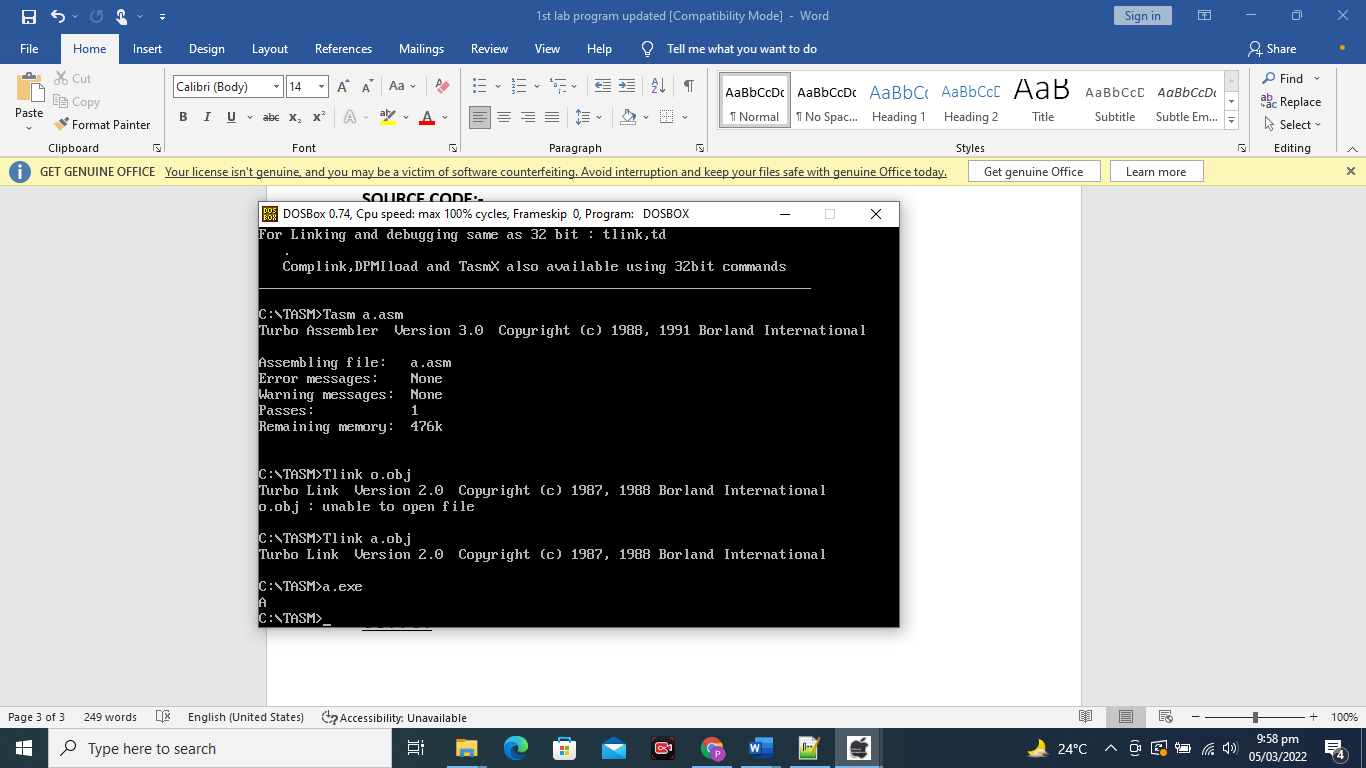
int 21h

mov ah,4ch

int 21h

end start

**OUT PUT;-**



**TASK # 3**

**Create a program which print your name with the help of function 02.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov ah,02

mov dl,'J'

int 21h

mov ah,02

mov dl,'A'

int 21h

mov ah,02

mov dl,'M'

int 21h

mov ah,02

mov dl,'A'

int 21h

mov ah,02

mov dl ,'L'

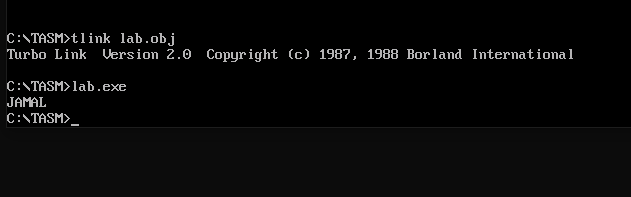
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**;-



**TASK #4**

**Write all errors which you know about in today class and write how you correct them**

**NEED EXPRESSION ERROR**

This error arrives during program because of add (:) after end start label

**FILE NOT FOUND**

This error arrives during program, because we did not save our file with .asm extension

**Lab 2**

# **BASIC DOS FUNCTIONS**

**Functions for take input.**

01= for single character

3fh= for string input

1. **Create a program which take input from user.**

.model small

.stack 100h

.data

.code

Start:

Mov ah, 01

Int 21h

Mov ah, 4ch

Int 21h

End start



1. **Create a program which take input and print it on output screen.**

.model small

.stack 100h

.data

.code

Start:

Code for input

Mov ah, 01

Int 21h

Mov ah, 02

Code for print input character

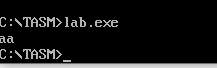
Mov dl, al

Int 21h

Mov ah, 4ch

Int 21h

End start



1. **Create a program which takes input a character from user and print it on new line.**

. model small

. stack 100h

.data

. code

Start:

Mov ah, 01

Int 21h

Mov dl, al

Mov ah,02

Mov dl, 0AH

Int 21h

Mov ah,02

Mov dl, 0DH

Int 21h

Mov ah,02

Mov dl, bl

Int 21h

Mov ah, 4ch

Int 21h

End start

1. **Take string as an input from user.**

.model small

.stack 100h

.data

.code

Start :

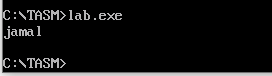
Mov ah, 3fh

Int 21h

Mov ah, 4ch

Int 21h

End start



**Lab 3**

# **VARIBLE**

Function 9 : is use to display string on screen

LEA :This instruction load address of variable in destination register.

Offset : Put address of variable (which contain string) in the destination register.

.data: responsible to declare variable

**CODE;-**

.model small

.stack 100h

.data

var db 10,"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$"

.code

start:

mov ax,@data

mov ds,ax

mov dx,offset var

mov ah,09

int 21h

mov ah,02

mov dl,10

int 21h

mov ah,02

mov dl,'A'

int 21h

mov dx,offset var

mov ah,09

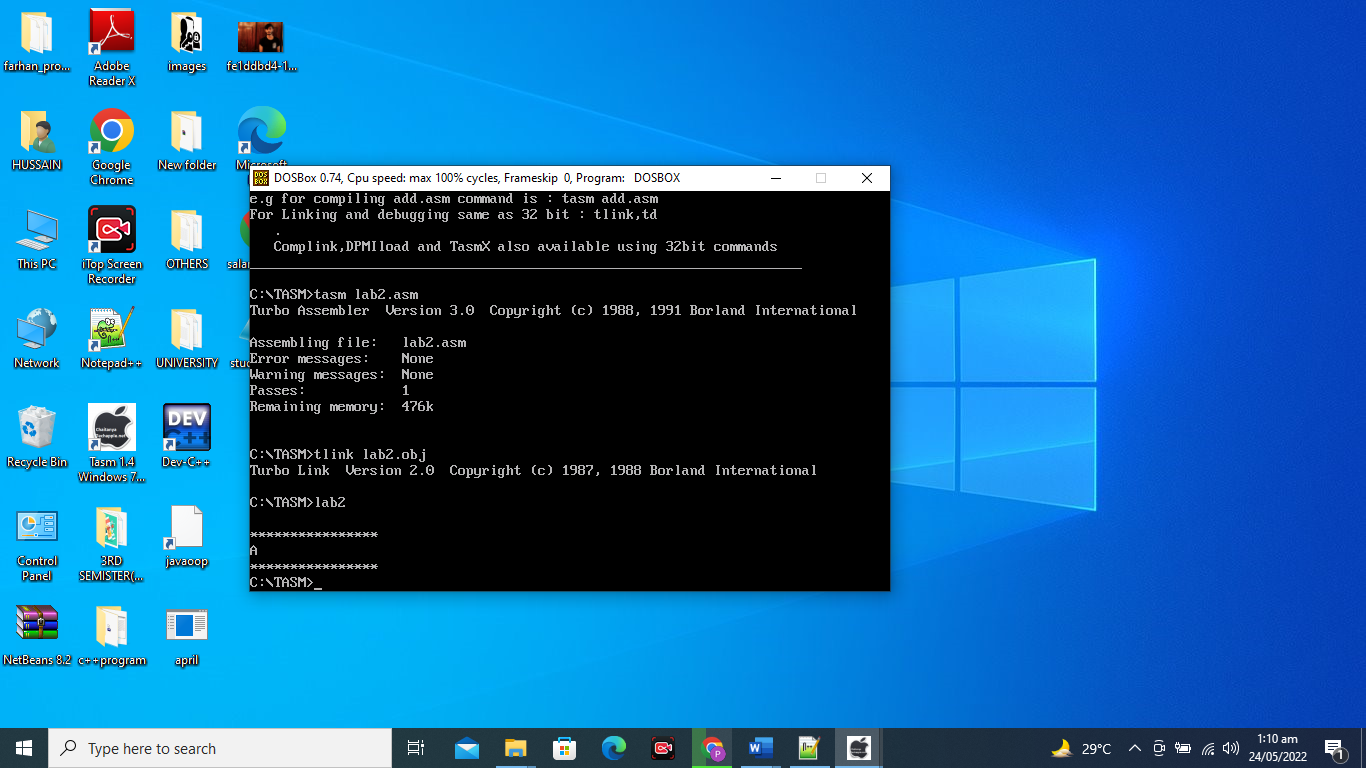
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**Object: To display a string on the screen.**

**Assembly Code:**

.model small

.stack 100h

.data

str1db 'Sindh Madressatul Islam University$' ;define byte

.code

main proc

;initialize ds

Mov ax,@data ;name of data segment defined by .data

Mov ds,ax ;initialize ds;display string

lea dx,str1 ;get string

mov ah,09h ;display string function

int 21h ;display string

;return to DOS

mov ah,4ch ;exit program with return code

int 21h ;DOS exit

mainendp

end main



**2. To display string on screen. (offset)**

. model small

. stack 100h

.data

Msg db " Hello world$"

.code

start:

mov ax,@data

Mov ds,ax

mov ah,09

mov dx, offset msg

int 21h

mov ah,4ch

int 21h

end start



**2. Display String on new line.**

.MODEL SMALL

.STACK 100H

.DATA

STRING\_1 DB 'SMI$'

STRING\_2 DB 'University $'

. CODE

MAIN PROC

MOV AX, @DATA ; initialize DS

MOV DS, AX

LEA DX, STRING\_1 ; load & display the STRING\_1

MOV AH, 9

INT 21H

MOV AH, 2 ; carriage return

MOV DL, 0DH

INT 21H

MOV DL, 0AH ; line feed

INT 21H

LEA DX, STRING\_2 ; load & display the STRING\_2

MOV AH, 9

INT 21H

MOV AH, 4CH ; return control to DOS

INT 21H

MAIN ENDP

END MAIN

**Lab Task:**

**1. Assembly code displaying bio data (multiple strings) on new line.**

.model small

.stack 100h

.data

str1db 'Name: JAMAL$' ;define byte

str2db 'Roll no.: 22S-065$'

str3db 'Section: 3B$'

str4db 'University: Sindh Madressatul Islam University.$'

.code

main proc

;initialize ds

movax,@data ;name of data segment defined by .data

movds,ax ;initialize ds

;display string

lea dx,str1 ;get string

mov ah,09h ;display string function

int 21h ;display string

mov dl,10 ;line feed

mov ah,02h

int 21h

;display string

lea dx,str2 ;get string

mov ah,09h ;display string function

int 21h ;display string

mov dl,10 ;line feed

mov ah,02h

int 21h

;display string

lea dx,str3 ;get string

mov ah,09h ;display string function

int 21h ;display string

mov dl,10 ;line feed

mov ah,02h

int 21h

;display string

lea dx,str4 ;get string

mov ah,09h ;display string function

int 21h ;display string

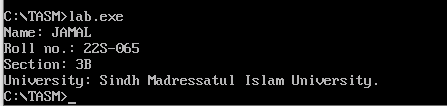
;return to DOS

mov ah,4ch ;exit program with return code

int 21h ;DOS exit

mainendp

end main



**Task:**

**1. Create a program which shows Multiple string on output screen**

**2: Create this format using Assembly Program.**

\*\*\*\*\*\*\*\*\*\*\*\*\*

Name:

Input Name from User

Father Name

Input father Name

Roll No:

Input roll No:

Institute Name:

Input Institute Name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**TASK**

**1. Create a program which shows Multiple string on output screen**

**SOURCE CODE**

.model small

.stack 100h

.data

var1 db 10,"MY NAME IS OWAIS BIN AMIR$"

var2 db 10,"I AM STUDY IN SMIU UNIVERSITY$"

var3 db 10,"ENROLL IN BSCS PROGRAM$"

.code

start:

mov ax,@data

mov ds,ax

mov dx,offset var1

mov ah,09

int 21h

mov dx,offset var2

mov ah,09

int 21h

mov dx,offset var3

mov ah,09

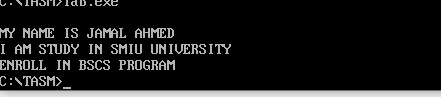
int 21h

mov ah,4ch

int 21h

end start

OUT PUT



**2: Create this format using Assembly Program.**

\*\*\*\*\*\*\*\*\*\*\*\*\*

Name:

Input Name from User

Father Name

Input father Name

Roll No:

Input roll No:

Institute Name:

Input Institute Name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SOURCE CODE;-**

.model small

.stack 100h

.data

var1 db 10,"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$"

var2 db 10,"NAME$"

var3 db 10,"FATHER$"

var4 db 10,"ROLL NO$"

var5 db 10,"INSTITUE NAME$"

var6 db 10,"\*\*\*\*\*\*\*\*\*\*\*\*\*\*$"

.code

start:

mov ax,@data

mov ds,ax

mov dx,offset var1

mov ah,09

int 21h

mov dx,offset var2

mov ah,09

int 21h

mov ah,2

mov dl,10

int 21h

mov ah,3fh

int 21h

mov dx,offset var3

mov ah,09

int 21h

mov ah,2

mov dl,10

int 21h

mov ah,3fh

int 21h

mov dx,offset var4

mov ah,09

int 21h

mov ah,2

mov dl,10

int 21h

mov ah,3fh

int 21h

mov dx,offset var5

mov ah,09

int 21h

mov ah,2

mov dl,10

int 21h

mov ah,3fh

int 21h

mov dx,offset var6

mov ah,09

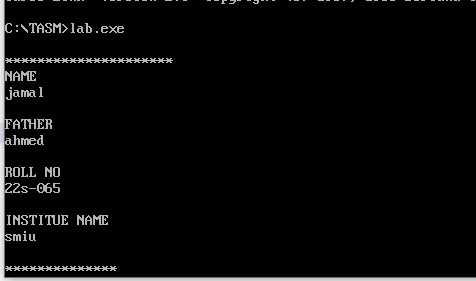
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT;-**



**Lab 4**

# **EXCHANGE KEYWORDS**

**2. To create a program for Exchange the value between different registers.**

. model small

. stack 100h

. code

start:

mov al, 49 ; ascii code of 1

mov bl, 50

xchgal,bl

mov ah,02

mov dl,al

int 21h

mov ah,02

movdl,bl

int 21h

Mov ah, 4ch

Int 21h

end start



**Task 1: Create a program with any new mnemonic.**

**Task 2: Use variable in above program.**

**Task 3: Create a above program using input.**

**Task 4: Create a program which exchange characters.**

**Task 2: Use variable in above program.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var1 db 10,"THE VLAUE OF A=1 AND B=2 BEFORE CHANGE$"

var2 db 10,"THE VLAUE OF A AFTER CHANGE$"

var3 db 10,"THE VLAUE OF B AFTER CHANGE$"

.code

start:

mov ax,@data

mov ds,ax

mov al,49

mov bl,50

mov dx,offset var1

mov ah,09

int 21h

xchgal,bl

mov dx,offset var2

mov ah,09

int 21h

mov ah,02

mov dl,al

int 21h

mov dx,offset var3

mov ah,09

int 21h

mov ah,02

mov dl,bl

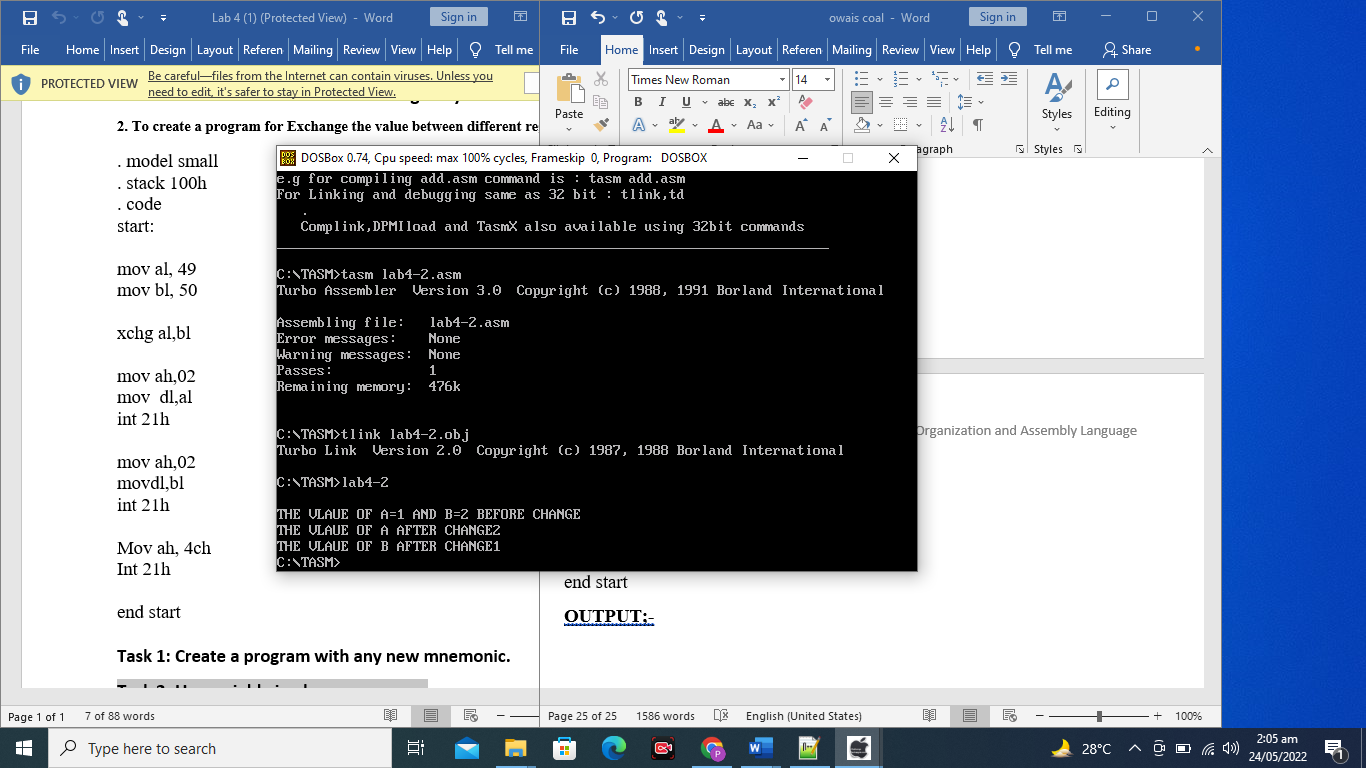
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**;-



**Task 4: Create a program which exchange characters.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var1 db 10,"VALUES BEFORE CHANGES A=1 B=2$"

var2 db 10, "VALUES OF A AFTER CHANGE : $"

var3 db 10,"VALUES OF B AFTER CHANGE : $"

.code

start:

mov ax,@data

mov ds,ax

mov dx,offset var1

mov ah,9

int 21h

mov al,49

mov bl,50

xchgal,bl

mov dx,offset var2

mov ah,09

int 21h

mov ah,02

mov dl,al

int 21h

mov dx,offset var3

mov ah,09

int 21h

mov ah,02

mov dl,bl

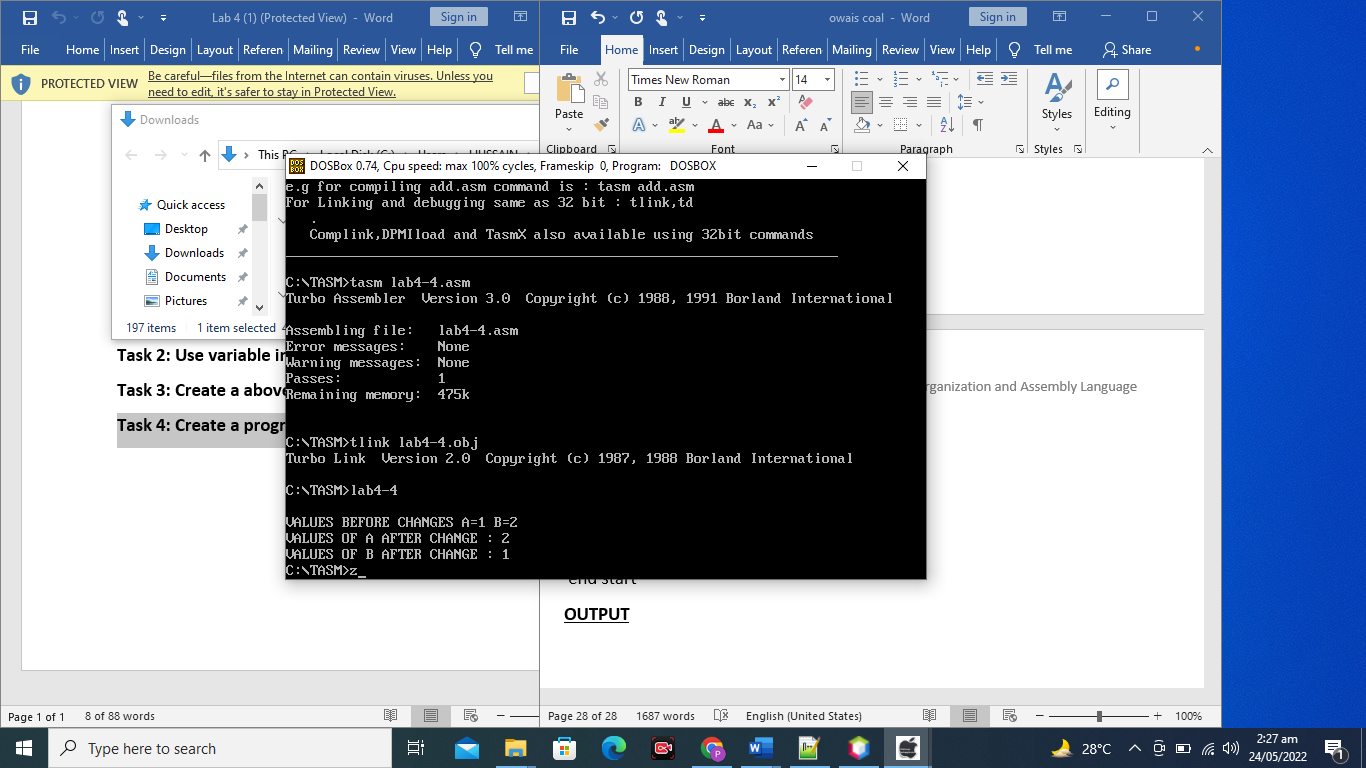
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**Lab 5**

# **CASE CONVERSION**

Program #1

**P#1: To convert the small letter into capital.**

.model small

.stack 100h

.data

str1 db 'Enter UPPERcase letter: - $'

str2 db 0dh,0ah,'CONVERTED LETTER: - $'

.code

start:

;initialie data segment

Mov ax,@data

Mov ds,ax

;display string

mov ah,09

lea dx,str1

int 21h

;input character

mov ah, 01h

int 21h

;lower to upper case conversion

sub al,20h

mov bl,al

;display string

mov ah,09

lea dx,str2

int 21h

;display character

;display string

mov ah,02

mov dl,bl

int 21h

mov ah,4ch

int 21h

end start



**Program #2**

P#1 EQU

.model small

.stack 100h

.data

p equ 41h

.code

start:

mov ax,@data

mov ds,ax

mov ah,02

mov dl,p

int 21h

mov ah,4ch

int 21h

end start

P#2 EQU

.model small

.stack 100h

.data

Msg1 equ 'type your name$'

Msg db msg1

p equ 41h

.code

start:

mov ax,@data

mov ds,ax

Mov ah,09

lea dx, msg

int 21h

mov ah,02

mov dl ,p

int 21h

mov ah,4ch

int 21h

end start

**P#3: Use EQU in program for new line.**

.model small

.stack 100h

.data

line equ 0AH

carriage equ 0DH

.code

start:

mov ah,02

mov dl, 'a'

int 21h

mov ah, 02

mov dl,line

int 21h

mov ah, 02

mov dl,carriage

int 21h

mov ah,02

mov dl, 'b'

int 21h

mov ah,4ch

int 21h

end start

**Task**

1. **Create a program which convert capital to small.**
2. **Create a program which print! on 1, @ on 2, # on 3 and so on up to 9.**
3. **Create a program using EQU**

**a -Create a program which convert capital to small.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var1 DB 10,"ENTER THE LETTER IN CAPITAL $"

var2 DB 10,"YOUR LETTER CONVERTED INTO SMALL $"

.code

start:

mov ax,@data

mov ds,ax

mov dx,offset var1

mov ah,09

int 21h

mov ah,01

int 21h

add al,32

mov dx,offset var2

mov ah,09

int 21h

mov dl,al

mov ah,02

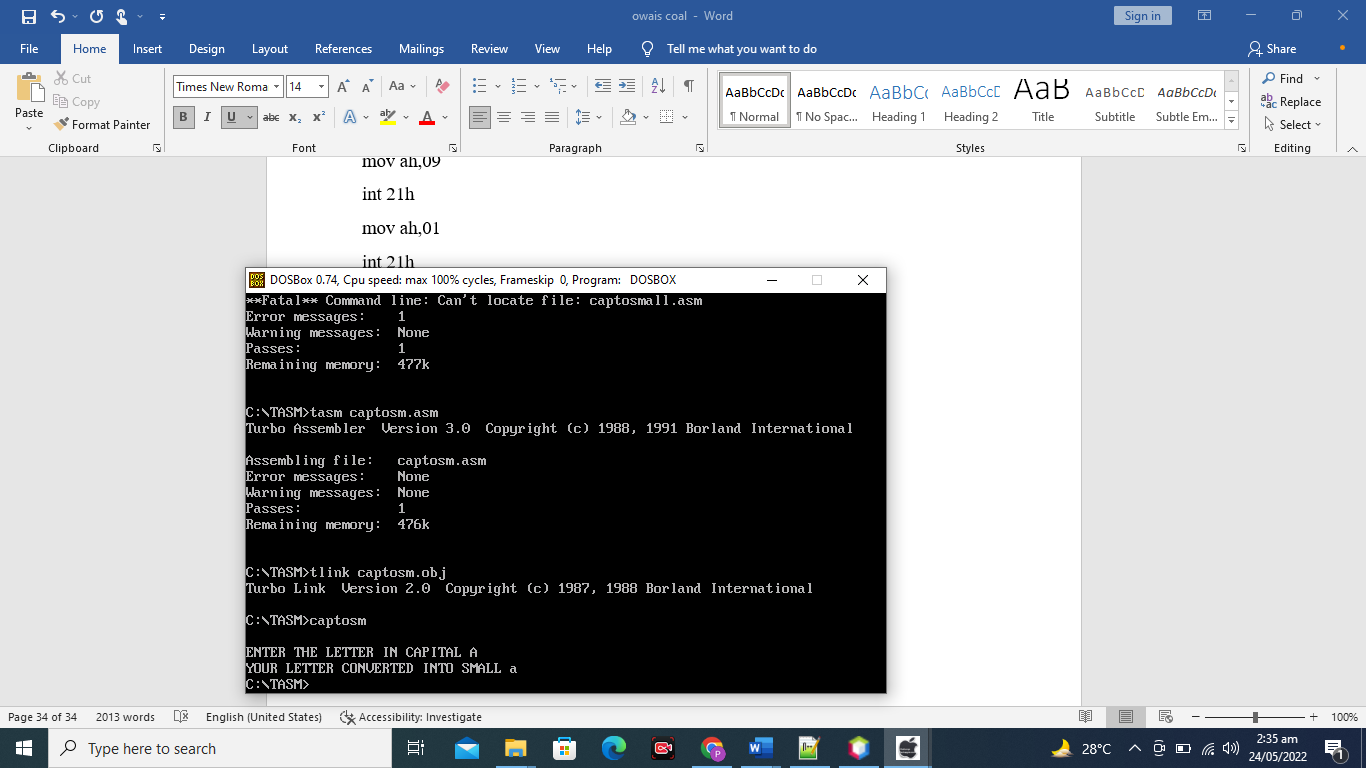
int 21h

mov ah,4ch

int 21h

end start

OUTPUT



**B.Create a program which print! on 1, @ on 2, # on 3 and so on up to 9.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var1 db 10,"!$"

var2 db 10,"@$"

var3 db 10,"#"

var4 db 10,"^$"

var5 db 10,"&$"

var6 db 10,"\*$"

var7 db 10,"-$"

var8 db 10,"+$"

var9 db 10,")$"

.code

start:

mov ax,@data

mov ds,ax

mov ah,01

int 21h

cmp al,49

je e1

cmp al,50

je e2

cmp al,51

je e3

cmp al,52

je e4

cmp al,53

je e5

cmp al,54

je e6

cmp al,55

je e7

cmp al,56

je e8

cmp al,57

je e9

e1:

mov dx,offset var1

mov ah,09

int 21h

mov ah,4ch

int 21h

e2:

mov dx,offset var2

mov ah,09

int 21h

mov ah,4ch

int 21h

e3:

mov dx,offset var3

mov ah,09

int 21h

mov ah,4ch

int 21h

e4:

mov dx,offset var4

mov ah,09

int 21h

mov ah,4ch

int 21h

e5:

mov dx,offset var5

mov ah,09

int 21h

mov ah,4ch

int 21h

e6:

mov dx,offset var6

mov ah,09

int 21h

mov ah,4ch

int 21h

e7:

mov dx,offset var7

mov ah,09

int 21h

mov ah,4ch

int 21h

e8:

mov dx,offset var8

mov ah,09

int 21h

mov ah,4ch

int 21h

e9:

mov dx,offset var9

mov ah,09

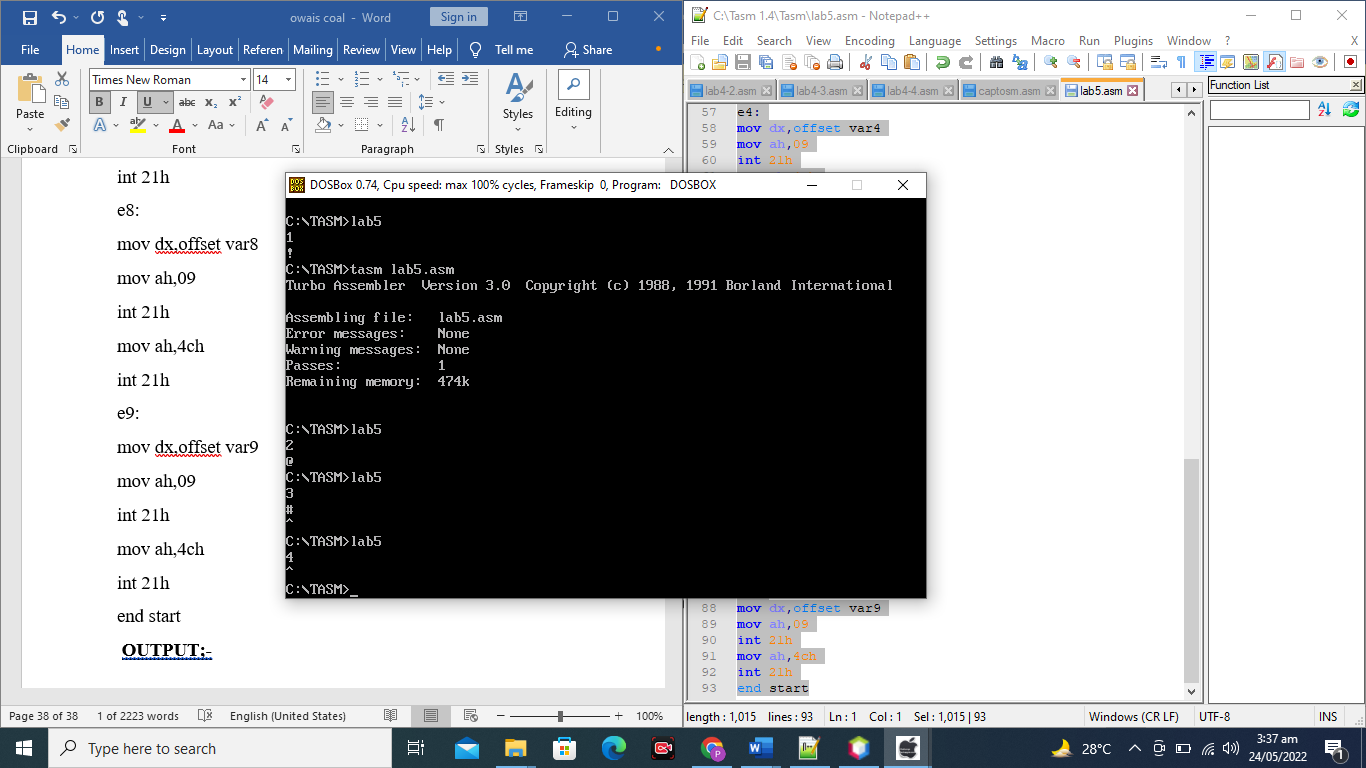
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**;-



**Create a program using EQU**

SOURCE CODE;-

.model small

.stack 100h

.data

exit equ 4ch

print equ 02

.code

start:

mov ah,01

int 21h

mov dl,al

mov ah,print

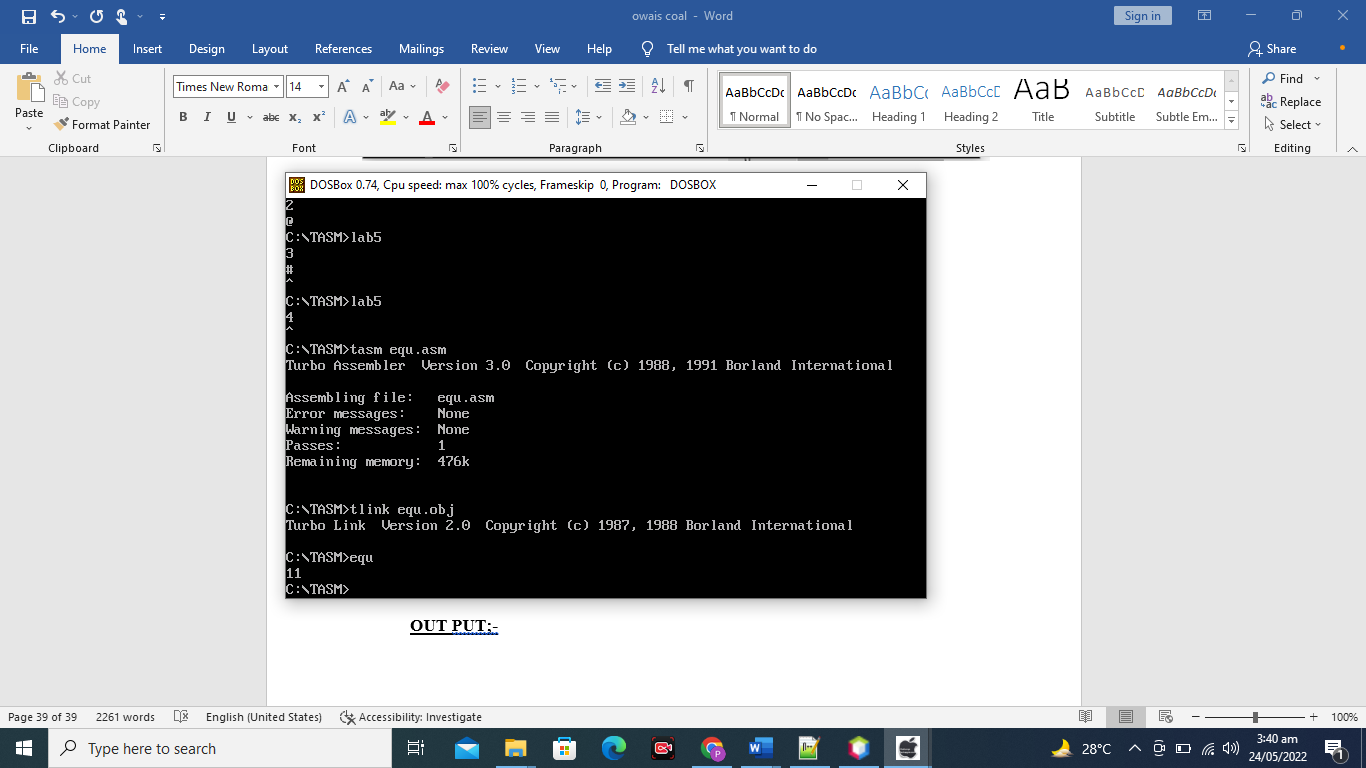
int 21h

mov ah,exit

int 21h

end start

**OUT PUT;-**



**LAB 6**

# **LOOP**

**Write a Program to Display ASCII characters from A to Z throughLooping.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,65

l1:

mov ah,02

int 21h

inc dl

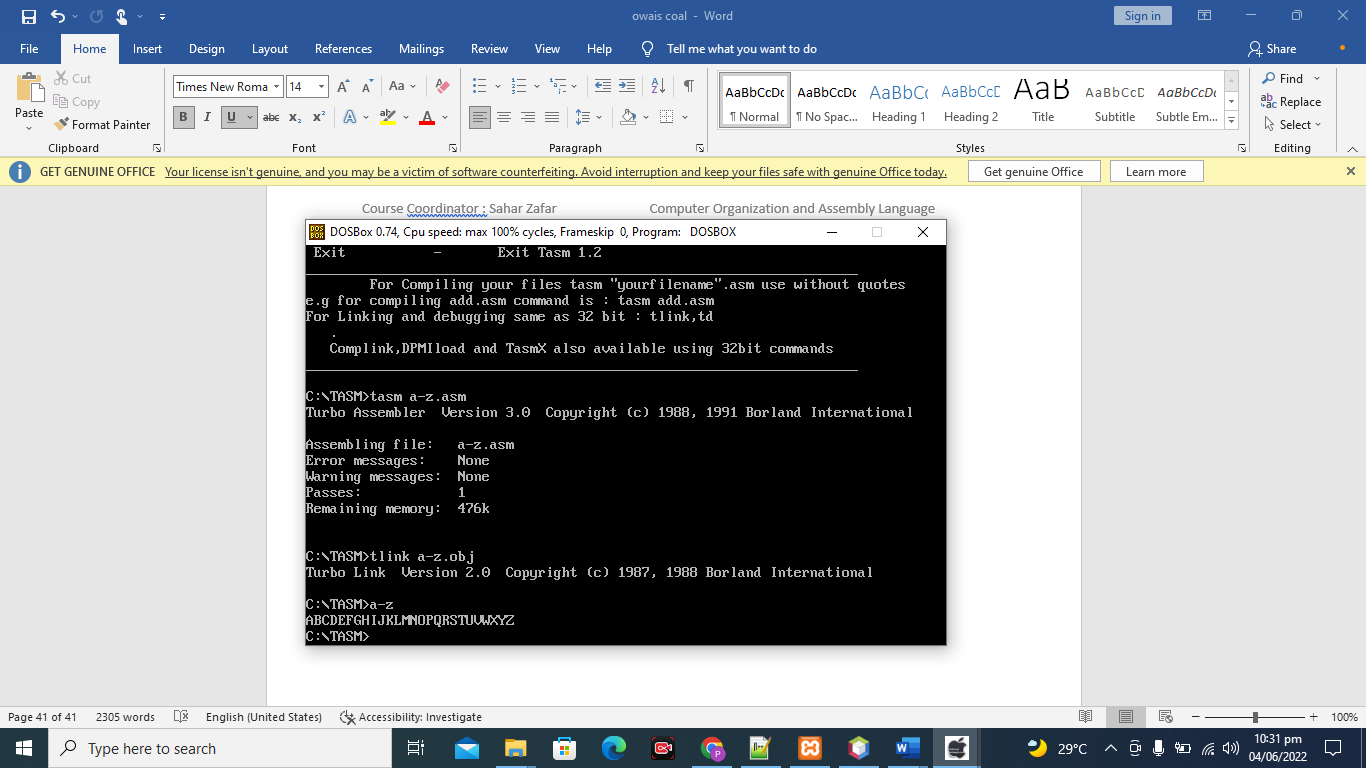
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**Write a Program to Display ASCII characters from a to z throughLooping.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,97

l1:

mov ah,02

int 21h

inc dl

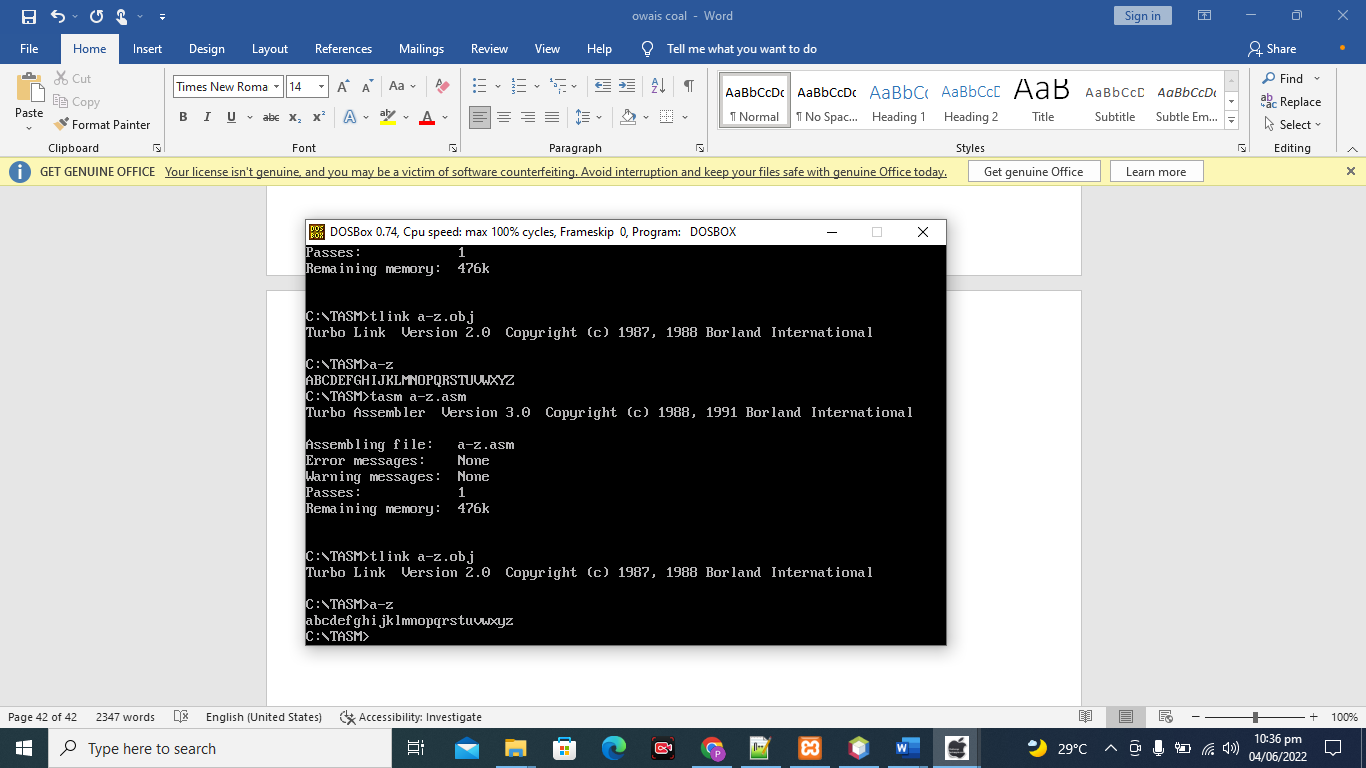
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**Write a Program to Display ASCII characters from 0 to 9 throughLooping.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,10

mov dl,48

l1:

mov ah,02

int 21h

inc dl

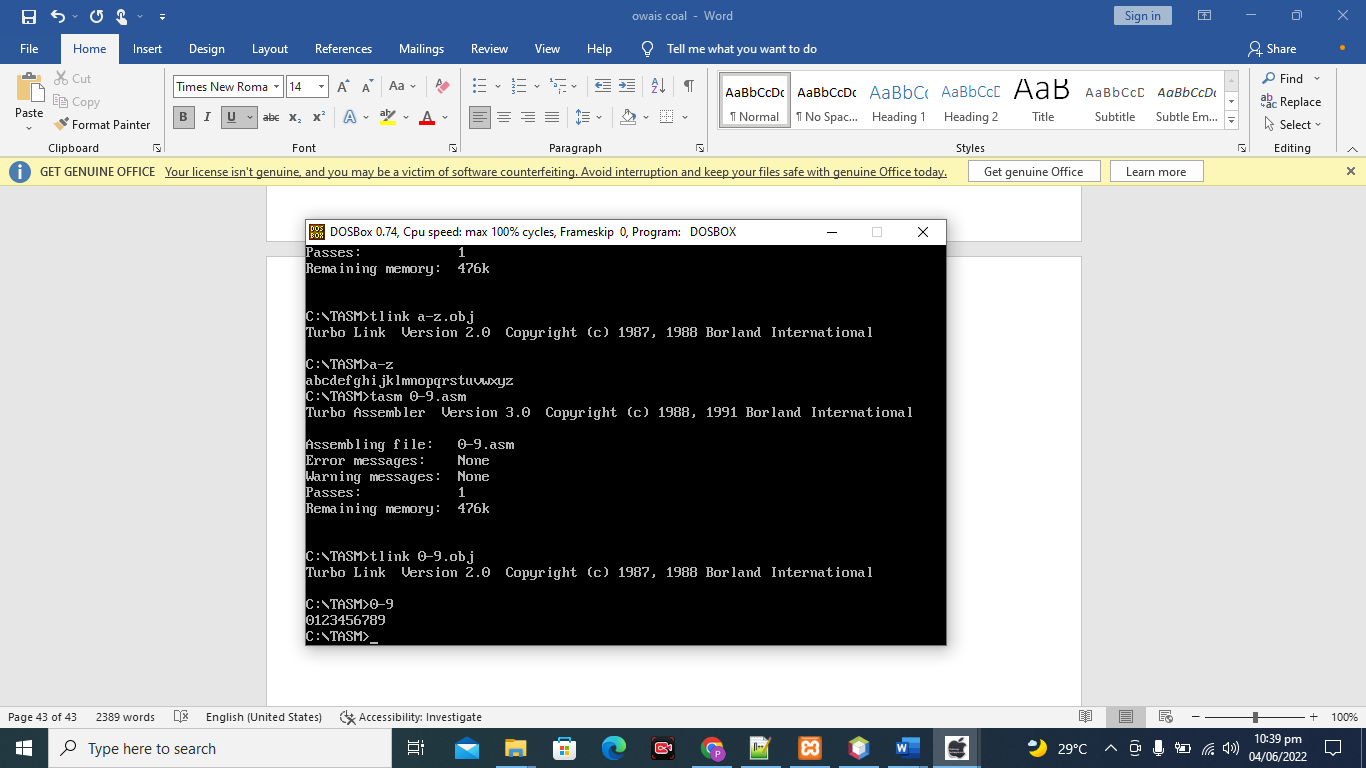
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**Write a Program to Display ASCII characters from Z to A through Looping.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,90

l1:

mov ah,02

int 21h

dec dl

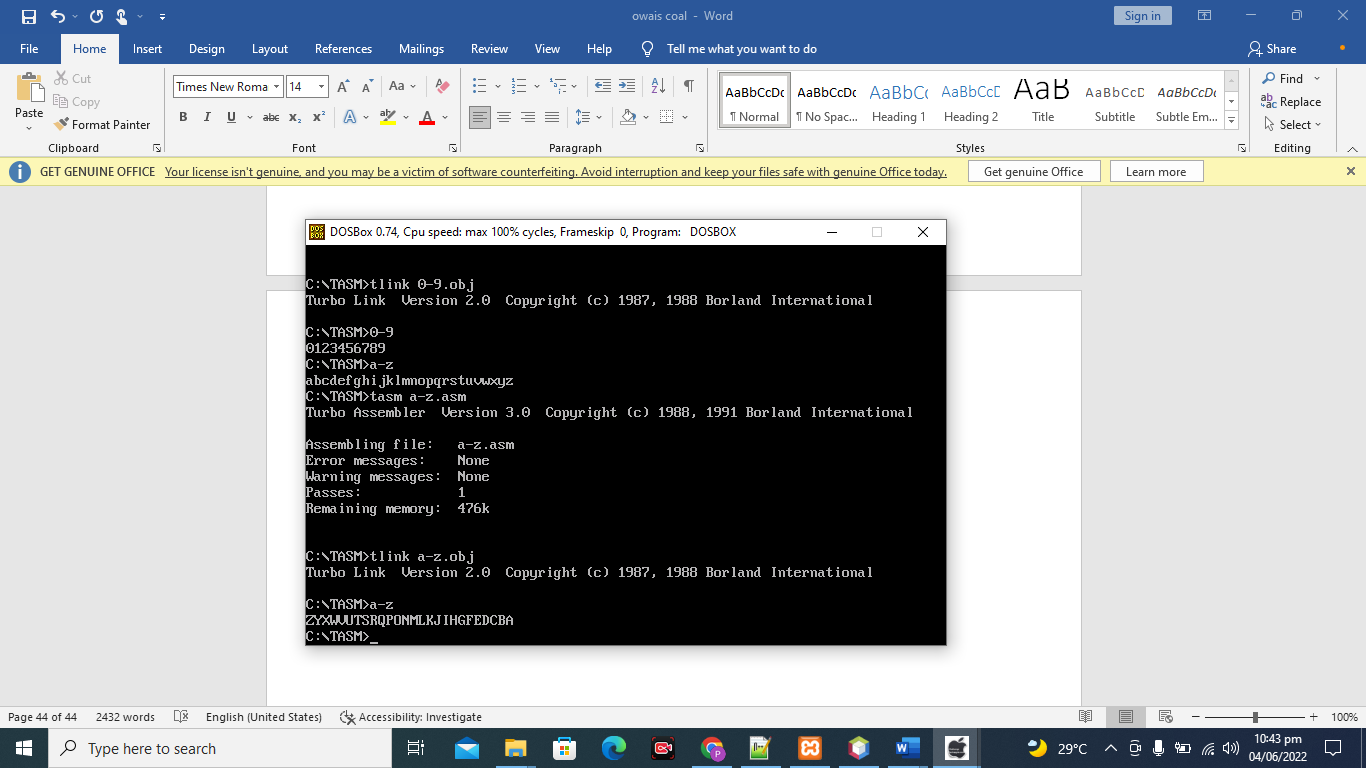
loop l1

mov ah,4ch

int 21h

end start

**OUT PUT;-**



**Write a Program to Display ASCII characters from 9 to 0 through Looping.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,10

mov dl,57

l1:

mov ah,02

int 21h

DEC dl

loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**Write a Program to Display ASCII characters from Z-A & a-z through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,90

l1:

mov ah,02

int 21h

dec dl

loop l1

mov cx,26

mov dl,97

l2:

mov ah,02

int 21h

dec dl

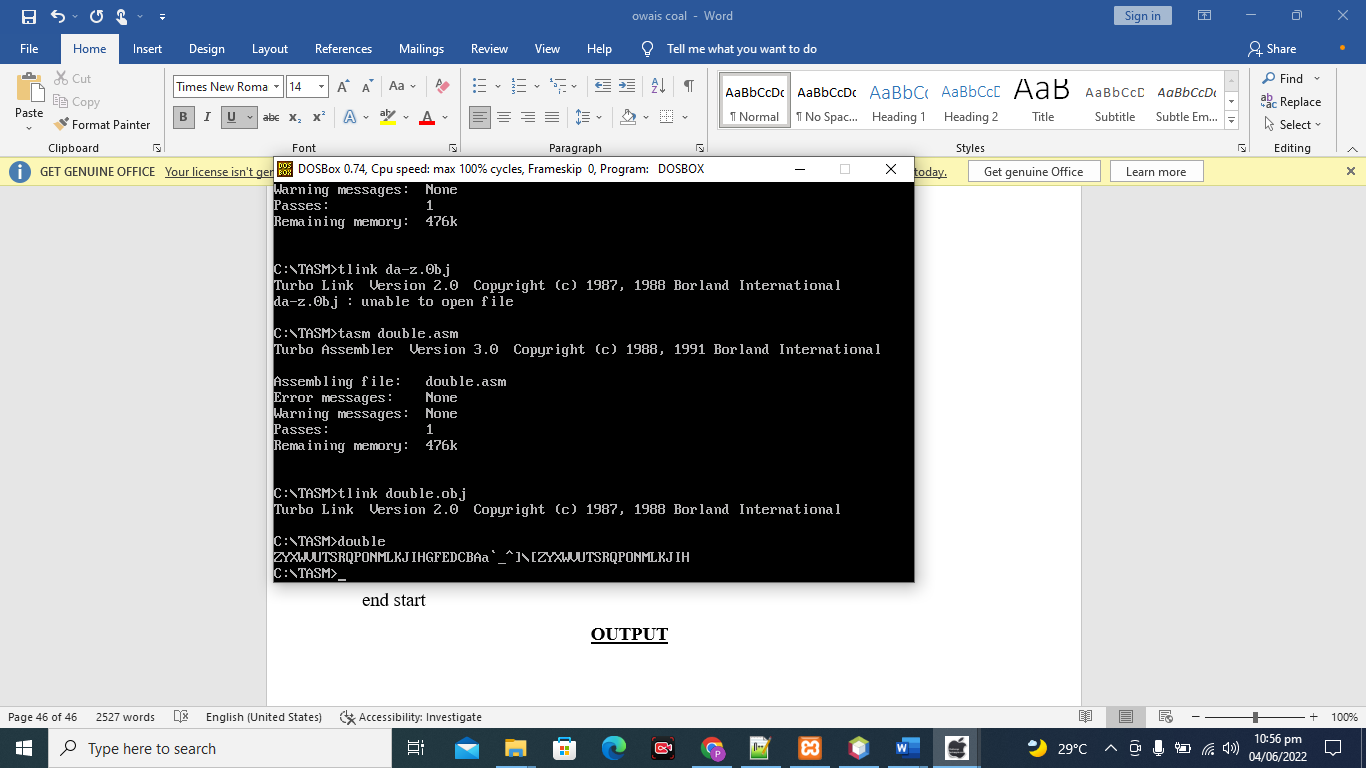
loop l2

mov ah,4ch

int 21h

end start

**OUTPUT**



**Program to print name 10 times**

**SOURCE CODE;-**

.model small

.stack 100h

.data

name2 db 10,"JAMAL $"

.code

start:

mov ax,@data

mov ds,ax

mov cx,10

l1:

mov dx,offset name2

mov ah,09

int 21h

inc dx

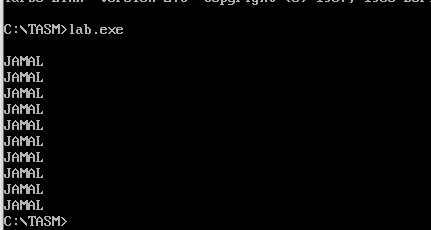
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**

****

**Write a program in Assembly Language to print following pattern:**

**0iiiiii0iiiiii0iiiiii0iiiiii0**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov cx,5

l1:

mov ah,2

mov dl,48

int 21h

mov bx,cx

mov cx,6

mov dl,'i'

l2:

mov ah,2

int 21h

loop l2

mov cx,bx

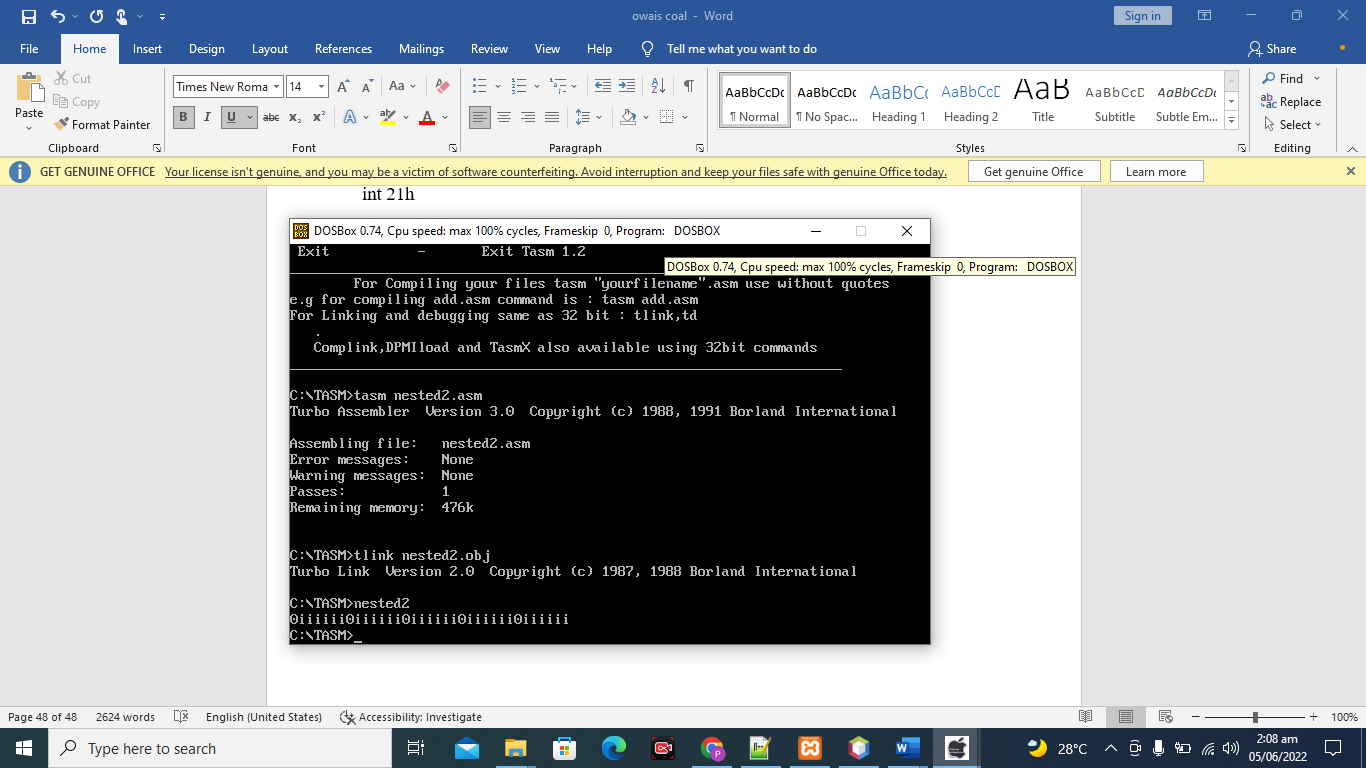
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**LAB 7**

# **JUMP**

.MODEL SMALL

.STACK 100H

.DATA

smsdb 10,"GUESS THE NUMBER $"

abovemsdb 10,"YOU GUESS ABOVE GO LOW $"

lowmsdb 10, "YOU GUESS LOW GO ABOVE $"

foundmsdb 10,"YOU FIND IT CONGRTAS $"

.CODE

start:

mov ax,@data

mov ds,ax

mov dx,offsetsms

mov ah,09

int 21h

input:

mov ah,01

int 21h

cmp al,'5'

jg above

jb below

je found

above:

mov dx,offsetabovems

mov ah,09

int 21h

jmp input

below:

mov dx,offsetlowms

mov ah,09

int 21h

jmp input

found:

mov dx,offsetfoundms

mov ah,09

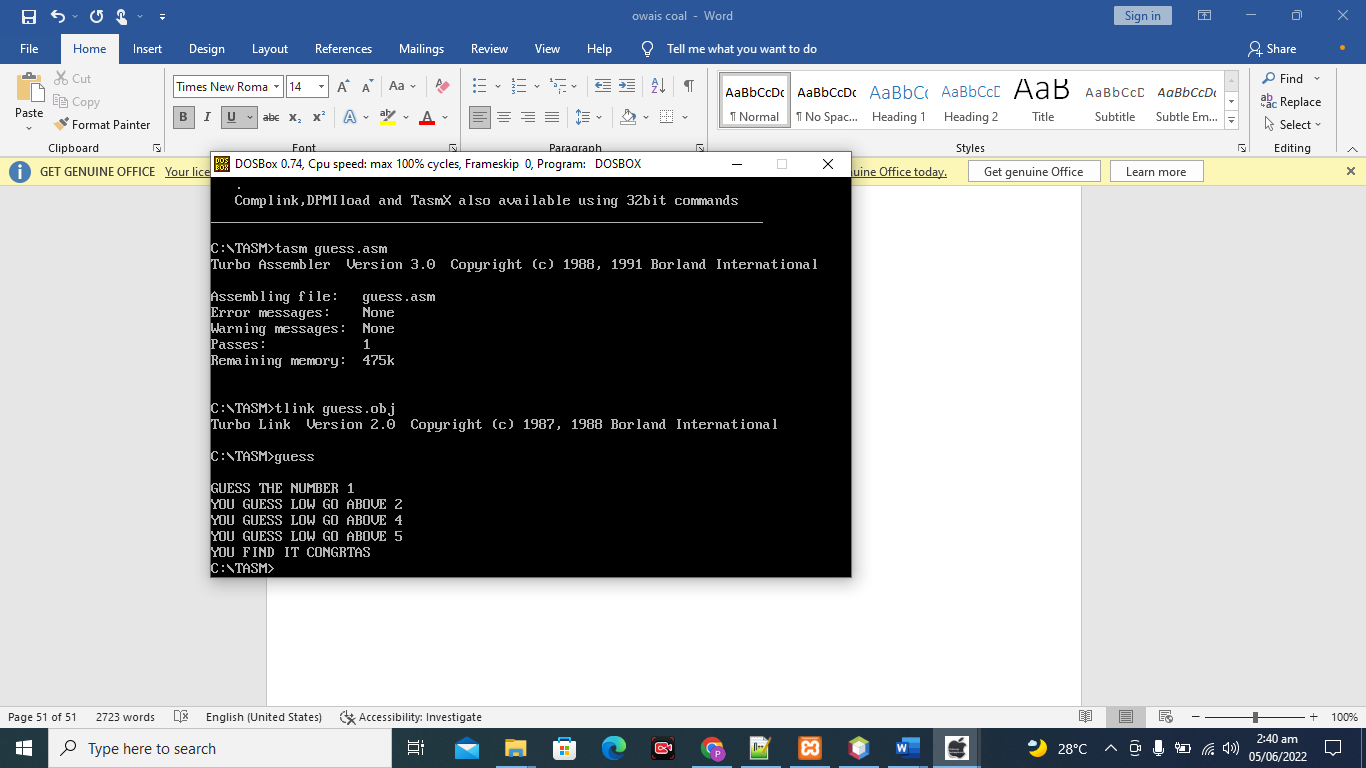
int 21h

mov ah,4ch

int 21h

end start

**OUT PUT**



**Create any programs using different jumps category.\**

**SOURCE CODE;-**

.model small

.stack 100h

.data

age db 10,"ENTER YOUR AGE $"

notvaliddb 10,"YOU ARE NOT GO $"

valid db 10," YOU ARE G0 $"

.CODE

start:

mov ax,@data

mov ds,ax

mov dx,offset age

mov ah,09

int 21h

mov ah,01

int 21h

cmp al,'9'

je equal

jbnotequal

equal:

mov dx,offset valid

mov ah,09

int 21h

mov ah,4ch

int 21h

notequal:

mov dx,offsetnotvalid

mov ah,09

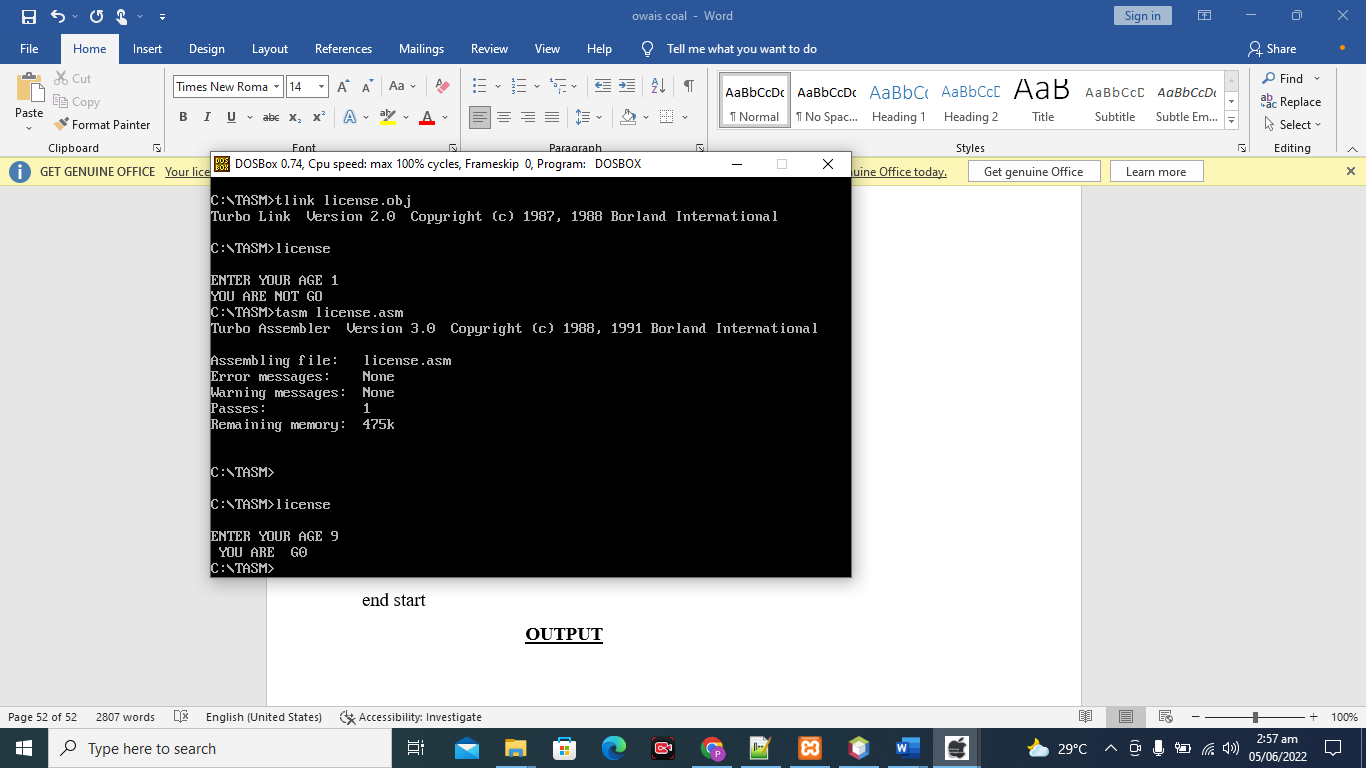
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**LAB 08**

# **COMPARISON**

**Program will take input until user press capital A**

**SOURCE CODE;-**

.MODEL SMALL

.stack 100h

.DATA

.CODE

start:

repeat:

mov ah,01

int 21h

cmpal,'a'

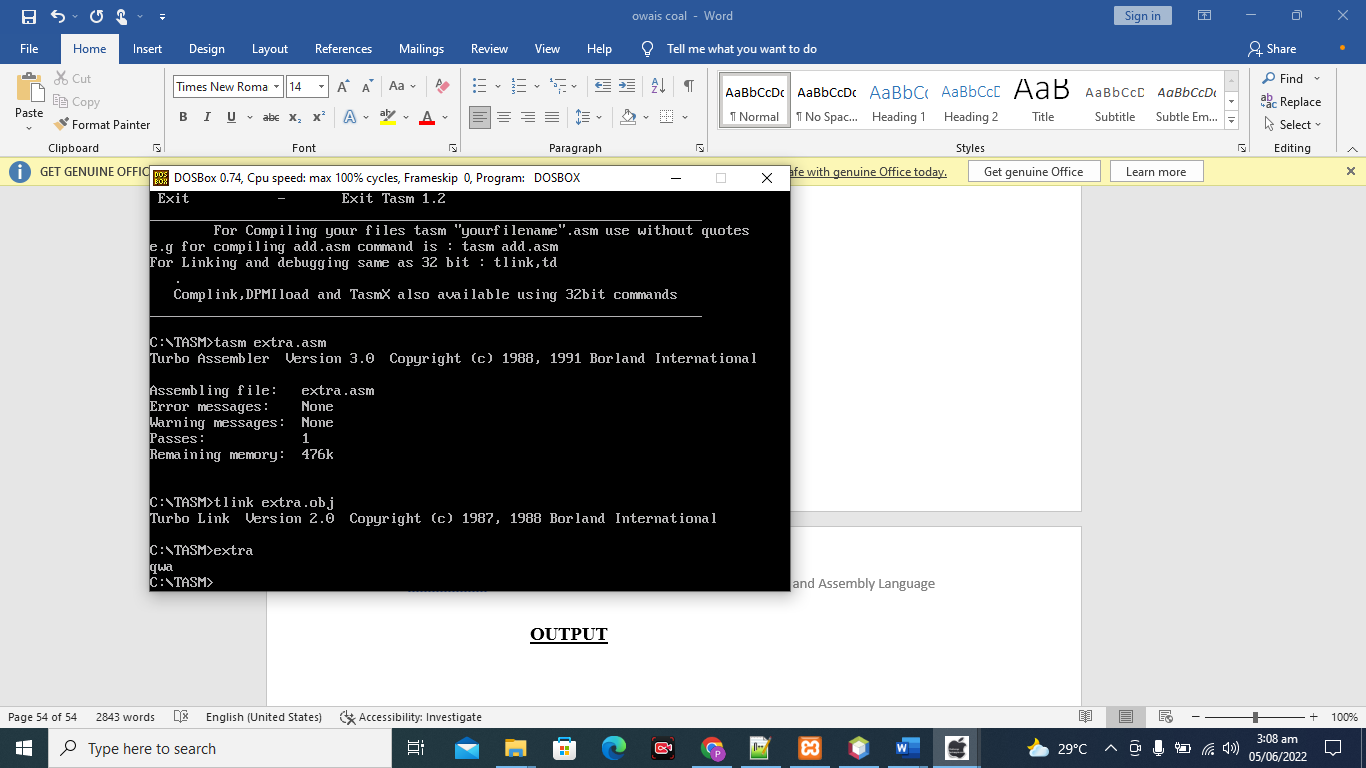
jne repeat

mov ah,4ch

int 21h

end start

**OUTPUT**



**Create a program using OR condition**

**SOURCE CODE;-**

.model small

.stack 100h

.code

start:

mov ah,01

int 21h

cmpal,'Y'

je then

cmp al, 'y'

je then

then :

mov dl, al

mov ah ,02

int 21h

jmp endif\_

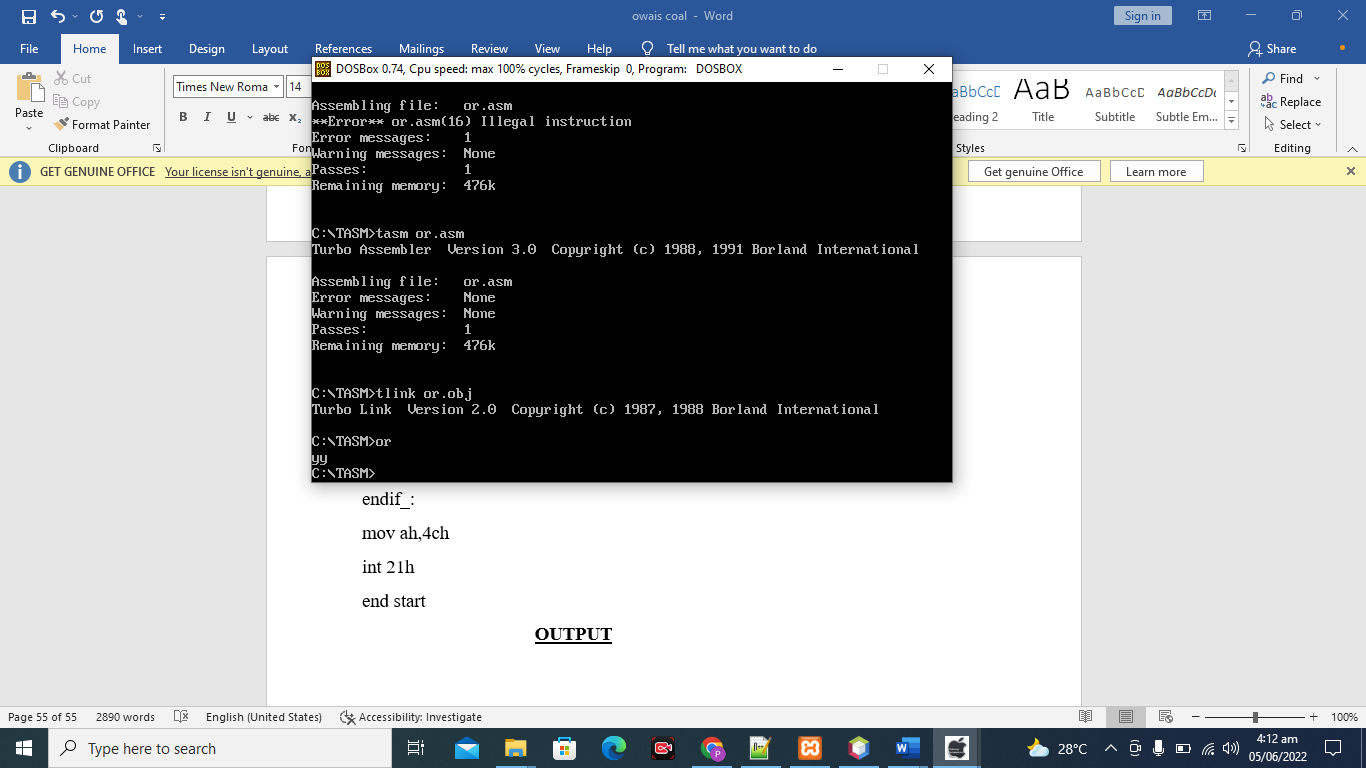
endif\_:

mov ah,4ch

int 21h

end start

**OUTPUT**



**Create a program using And logic**

**SOURCE CODE;-**

.model small

.stack 100h

.code

start:

mov ah,01

int 21h

cmpal,'Y'

je then

cmp al, 'y'

je then

then :

mov dl, al

mov ah ,02

int 21h

jmp endif\_

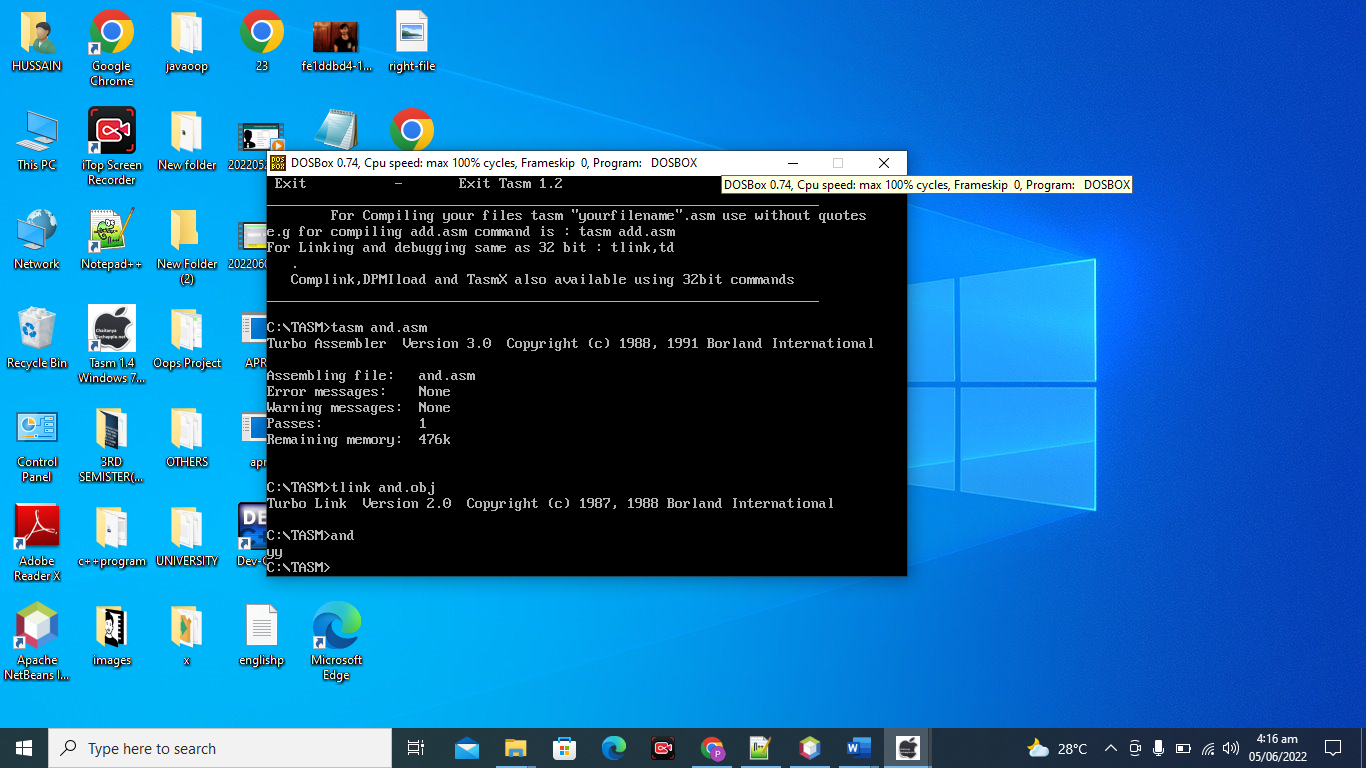
endif\_:

mov ah,4ch

int 21h

end start

**OUTPUT;-**



**Create pattern with help of nested loop**

**SOURCE CODE**

.model small

.stack 100h

.data

.code

start:

MOV CX,3

l1:

mov ah,2

mov dl,97

int 21h

mov bx,cx

mov cx,3

mov dl,48

l2:

mov ah,2

int 21h

inc dl

loop l2

mov cx,bx

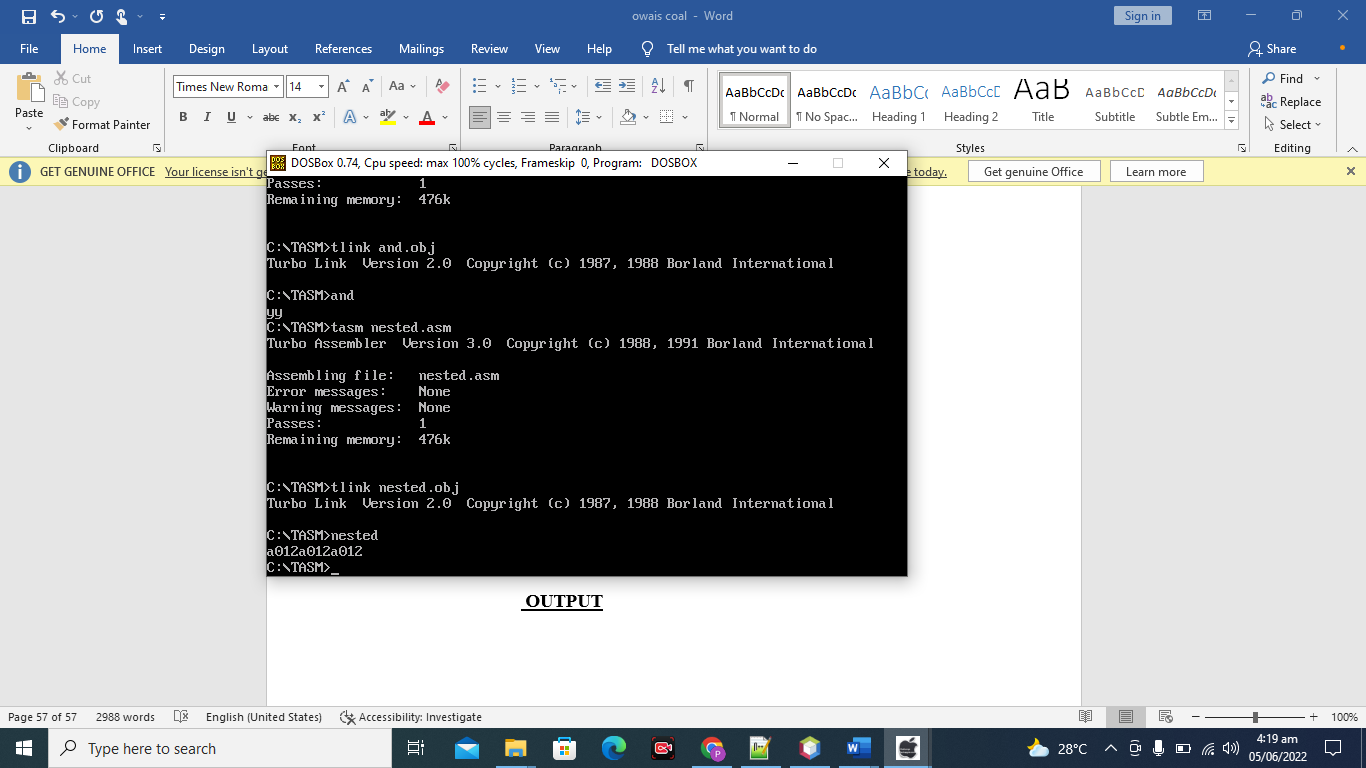
loop l1

mov ah,4ch

int 21h

end start

**OUTPUT**



**LAB 09**

# **STRINGS**

Create a program which compare string and print message accordingly. Hint : use jg,jl

SOURCE CODE;-

.model small

.stack 100h

.data

var1 db 10,"JAMAL$"

var2 db 10,"JAMAL$"

m db 10,"MATCH $"

n db 10,"NOT MATCH$"

.code

start:

mov ax,@data

mov ds,ax

mov es,ax

lea si,var1

lea di,var2

cld

cmpsb

jz match

mov dx,offset n

mov ah,09

int 21h

mov ah,4ch

int 21h

match:

mov dx,offset m

mov ah,09

int 21h

mov ah,4ch

int 21h

end start

OUTPUT



**Create a program for Scan and compare string instruction.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

str1 db "important equation$"

fmsg db 10,"FOUND $"

ntmsg db 10,"NOT FOUND $"

.code

start:

mov ax,@data

mov ds,ax

mov es,ax

mov cx,100

cld

lea di,str1

mov al,'e'

repnescasb

je found

mov dx,offsetntmsg

mov ah,09

int 21h

mov ah,4ch

int 21h

found:

mov dx,offsetfmsg

mov ah,09

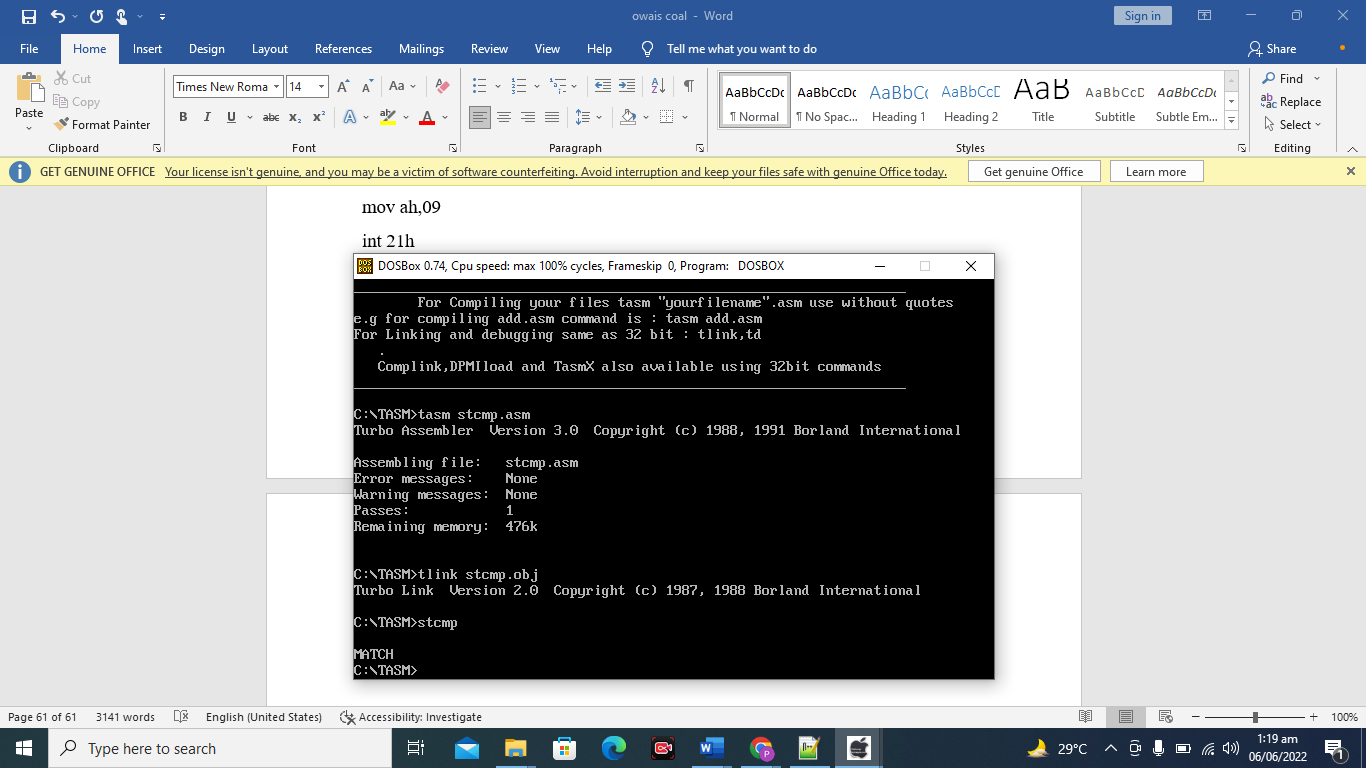
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**Create program using stosb store al data on 2nd location.**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var db "hello$"

.code

start:

mov ax,@data

mov ds,ax

mov es,ax

lea di,var+2

mov al,'A'

cld

stosb

mov ah,09

mov dx,offset var

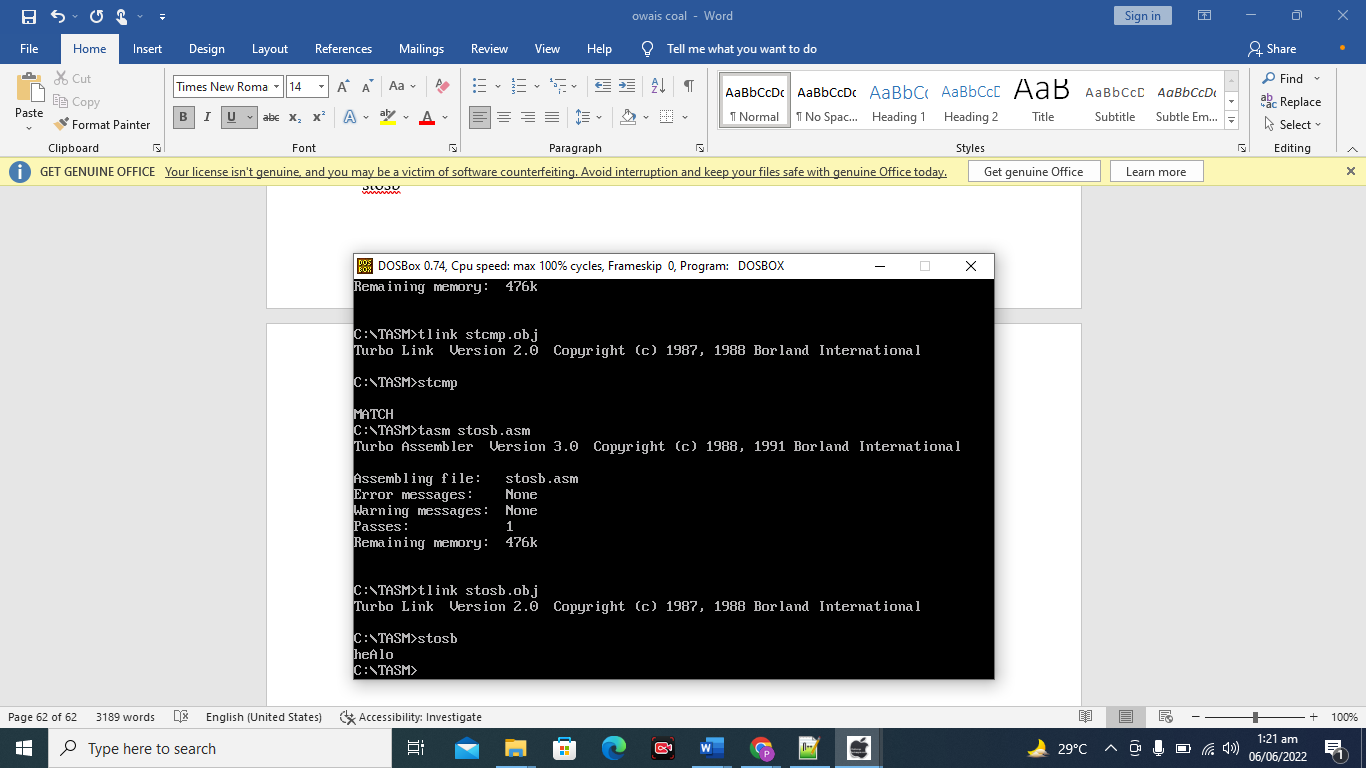
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**LAB 10**

# **BIT MANUPULATION**

**Create a Program for ROR, ROL,RCL,RCR.**

**SOURCE CODE;-**

ROR;-

.model small

.stack 100h

.code

start:

mov dx,8

ror dx,3

add dx,48

mov ah,2

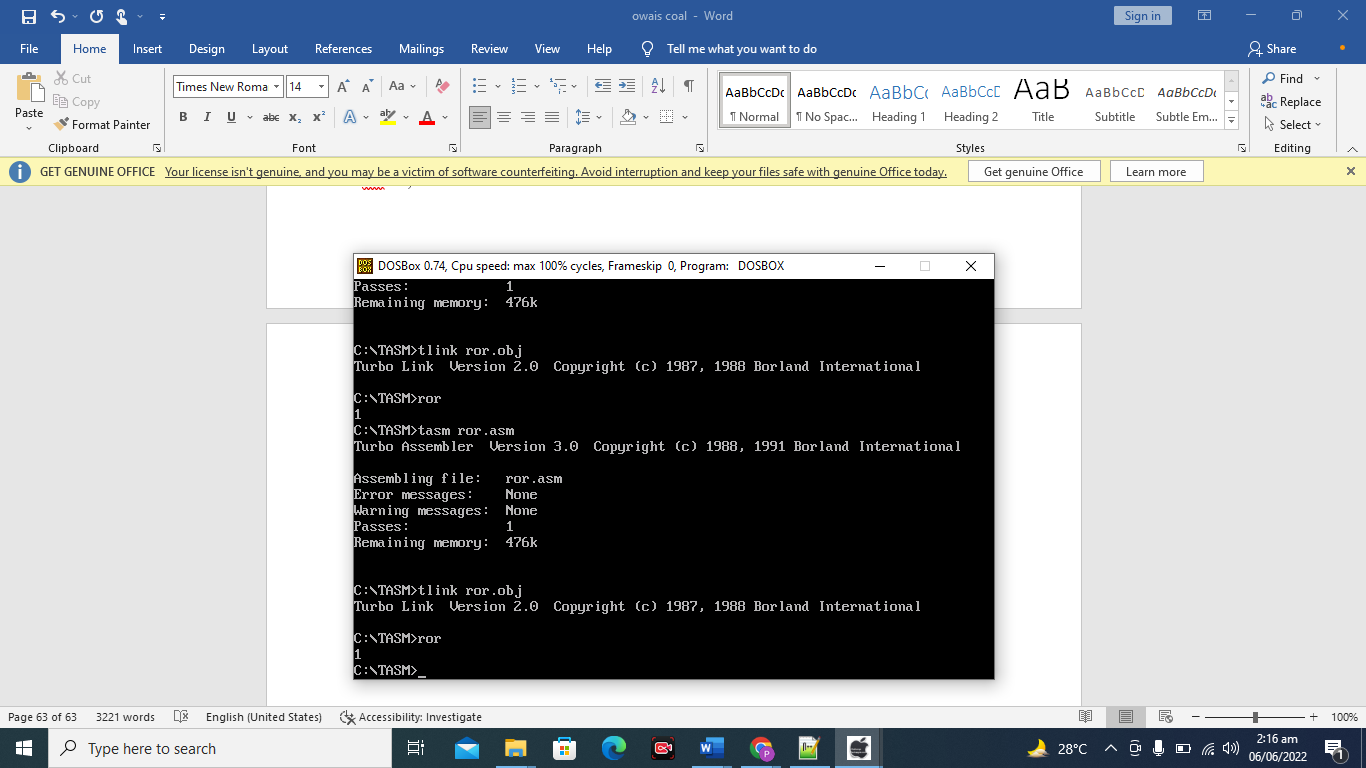
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT;-**



**RCL**

**SOURCER CODE;-**

.model small

.stack 100h

.data

.code

start:

mov dl, 4

rcl dl, 1

add dl, 48

mov ah, 02

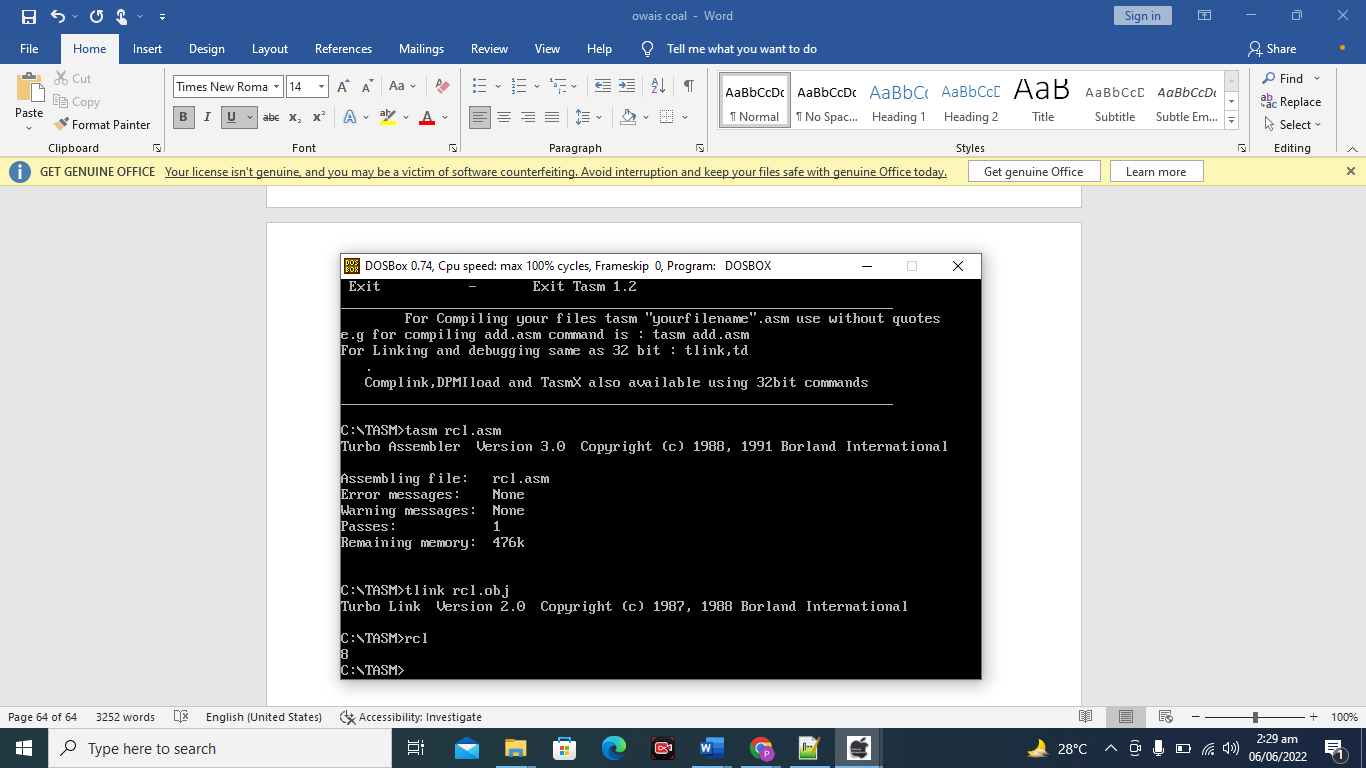
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



**ROL;-**

**SOURCE CODE;-**

.model small

.stack 100h

.data

.code

start:

mov dl, 4

rol dl, 1

add dl, 48

mov ah, 02

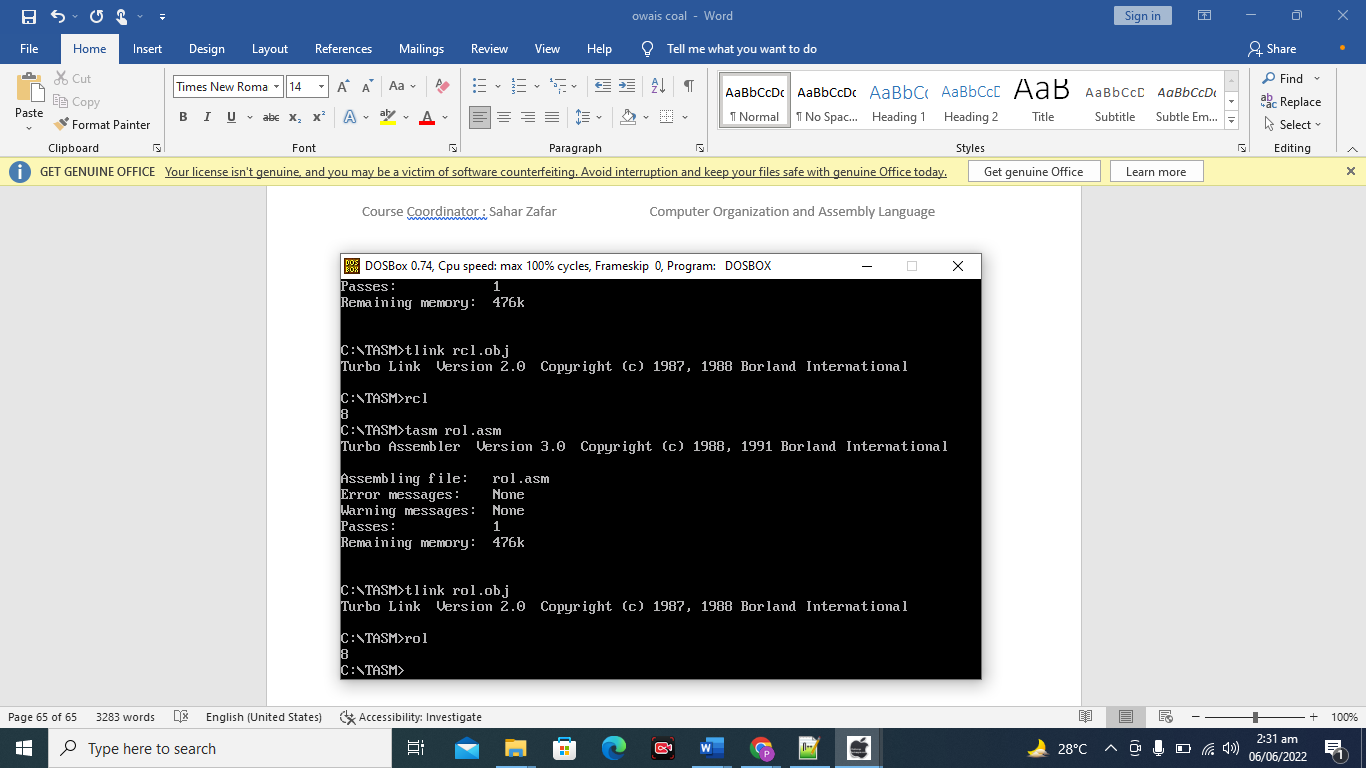
int 21h

mov ah,4ch

int 21h

end start

**OUTPUT**



Create a program for NOT instruction.

SOURCE CODE;-

.model small

.stack 100h

.data

.code

start:

mov dl, 165

; not dl

mov ah, 02

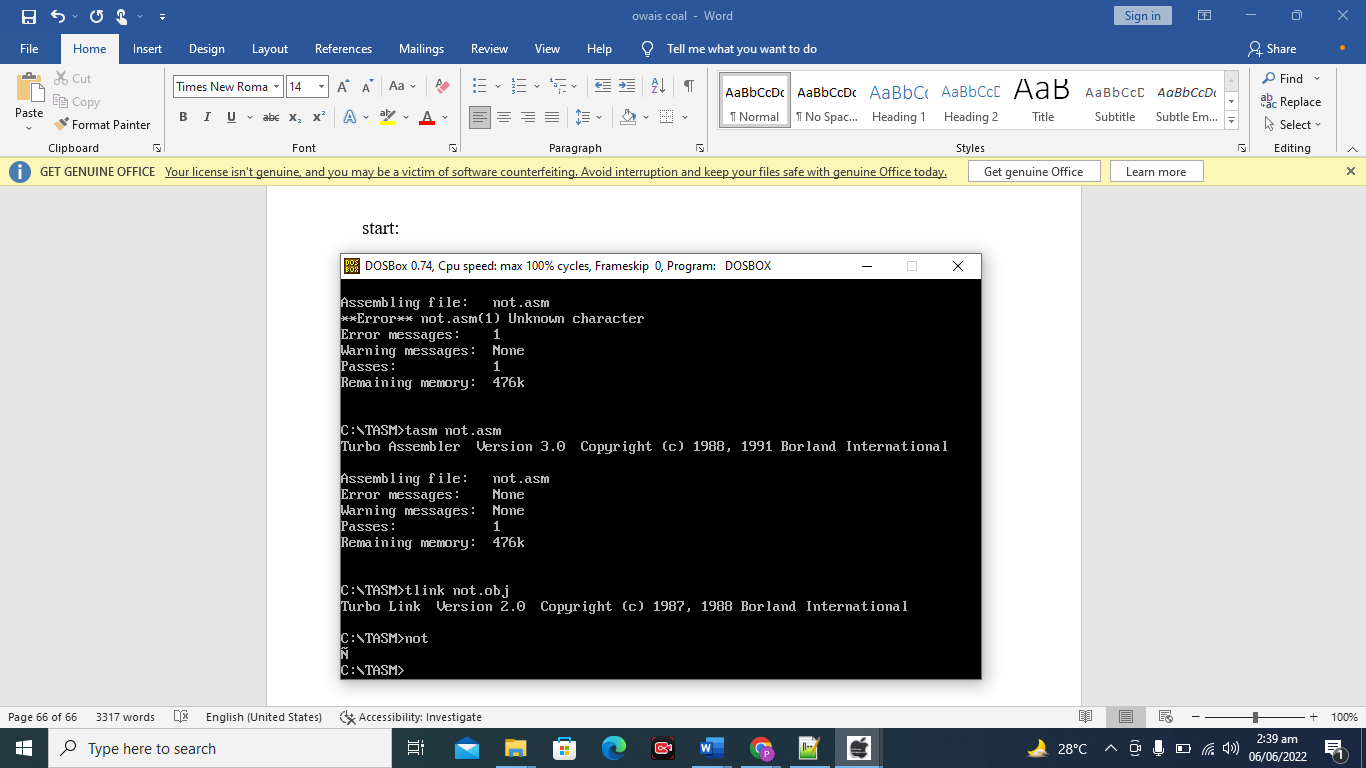
int 21h

mov ah, 4ch

int 21h

end start

OUTPUT;-



Create a program which do MUL and DIV using shlshr in one program.

.model small

.stack 100h

.data

.code

start:

mov dl,4

shl dl,1

add dl, 48

mov ah, 02

int 21h

mov bl, dl

MOV dl, 10

MOV ah, 02h

INT 21h

MOV dl, 13

MOV ah, 02h

INT 21h

mov dl, bl

sub dl, 48

shr dl, 1

add dl, 48

mov ah, 02

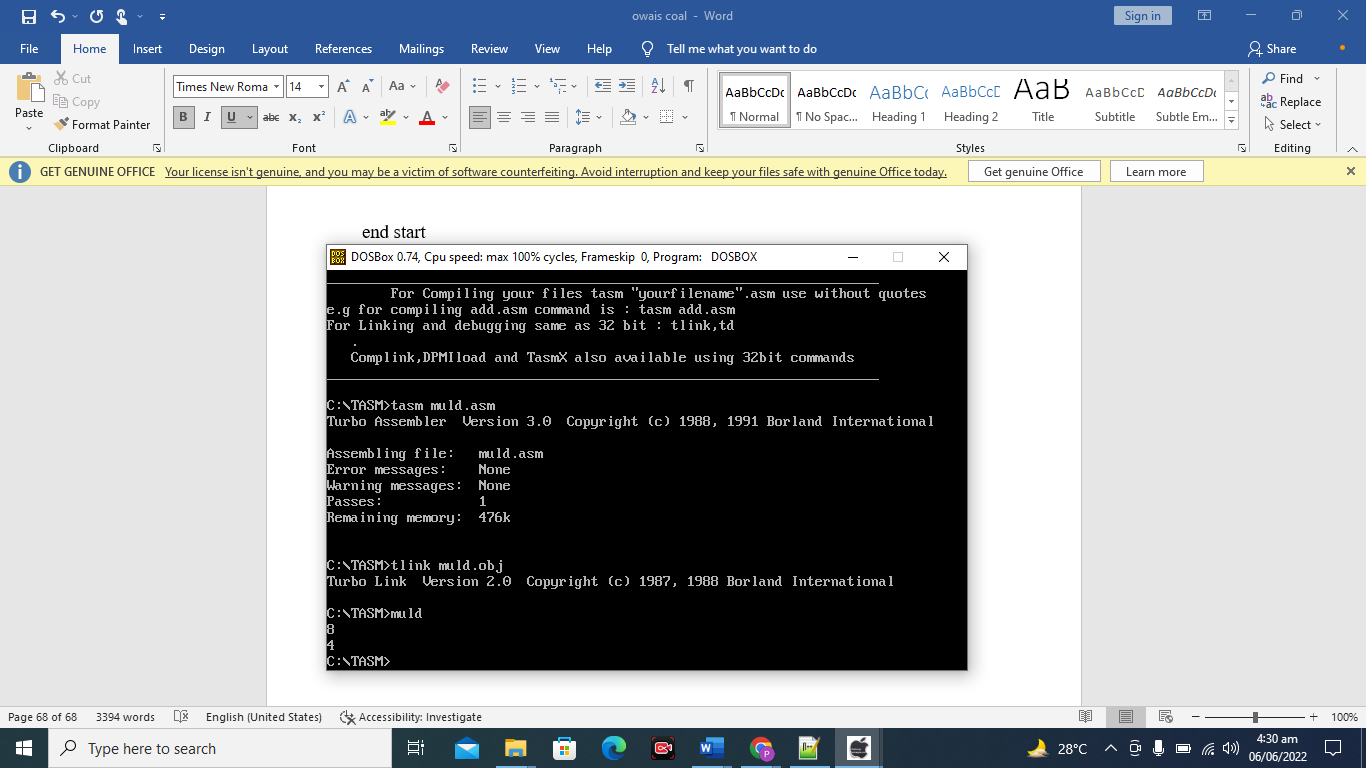
int 21h

mov ah, 4ch

int 21h

end start

OUTPUT



**LAB 11**

# **STACK, MACRO AND SUBROUTINES**

**Program 1 : Create a program using PUSH or POP.**

.model small

.stack 100h

.code

start:

mov ax,49

mov bx 50

mov cx,51

push ax

push bx

push cx

mov cx,3

l1:

pop dx

mov ah,02

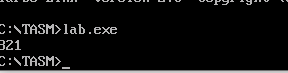
int 21h

loop l1

mov ah,4ch

int 21h

end start



**program 2: Reverse String with the help of stack**

.model small

.stack 100h

.data

var db "smiu$"

.code

start:

mov ax,@data

mov ds,ax

mov si, offset var

mov cx,4

l1:

mov bx,[si]

push bx

inc si

loop l1

mov cx , 4

l2:

pop dx

mov ah,02

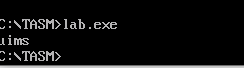
int 21h

loop l2

mov ah,4ch

int 21h

end start



**Program 3 : Create a program using Subroutine.**

.model small

.stack 100h

.code

start:

Mov al,49

mov bl,50

call proc1

call display1

mov ah,4ch

int 21h

proc1 proc

mov al,50

mov bl,49

ret

proc1 endp

display1 proc

mov ah,02

mov dl,al

int 21h

mov ah,02

mov dl,bl

int 21h

ret

display1 endp

end start



**Program 4: Create a program using nested subroutine.**

.model small

.stack 100h

.code

start:

Mov al,49

mov bl,50

call proc1

call display1

mov ah,4ch

int 21h

proc1 proc

mov al,50

mov bl,49

ret

proc1 endp

display1 proc

mov ah,02

mov dl,al

int 21h

mov ah,02

mov dl,bl

int 21h

ret

display1 endp

end start

program 5: create a program using Macro

.model small

.stack 100h

.code

start:

Dos\_rtn macro

mov ah,4ch

int 21h

endm dos\_rtn

newline macro

mov ah,02

mov dl,10

int 21h

endm newline

input macro

mov ah,01

int 21h

endm input

output macro

mov ah,02

mov dl,al

int 21h

endm output

mov ah,01

int 21h

newline

mov ah,02

mov dl,al

int 21h

newline

dos\_rtn

end start

**Task:**

Create a program using subroutine.

Create a program using macro.

Create a program using stack.

**Create a program using subroutine.**

.model small

.stack 100h

.data

.code

start:

mov al,'2'

mov bl,'3'

call overwrite

call display1

mov ah,4ch

int 21h

overwrite proc

mov al,'3'

mov bl,'2'

ret

overwrite endp

display1 proc

mov ah,02

mov dl,al

int 21h

mov ah,02

mov dl,bl

int 21h

ret

display1 endp

end start



**Create a program using stack.**

.model small

.stack 100h

.data

.code

start:

mov ax,49

push ax

pop bx

mov ah,02

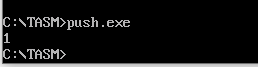
mov dx,bx

int 21h

mov ah,4ch

int 21h

end start



**Create a program using macro.**

.model small

.stack 100h

.DATA

msg1 db "this is macro program by jamal$"

.code

start:

mov ax,@data

mov ds,ax

print macro p

mov ah,09

lea dx,p

int 21h

endm print

print msg1

end start



**LAB 12**

# **GRAPH STRAIGHT LINES**

**Change the position of the line in the above program.**

.model small

.stack 100h

.data

.code

start:

mov ax, 13 ; set graphics mode

int 10h ; set graphics mode

mov ah, 0ch ;write pixel on screen

mov al, 13

mov cx, 10

mov dx, 30 ; position of line changes here

l1:

int 10h

inc cx

cmp cx,200

jle l1

mov ah, 0

int 16h ; keyboard interrupt

mov ax, 3 ; set text mode

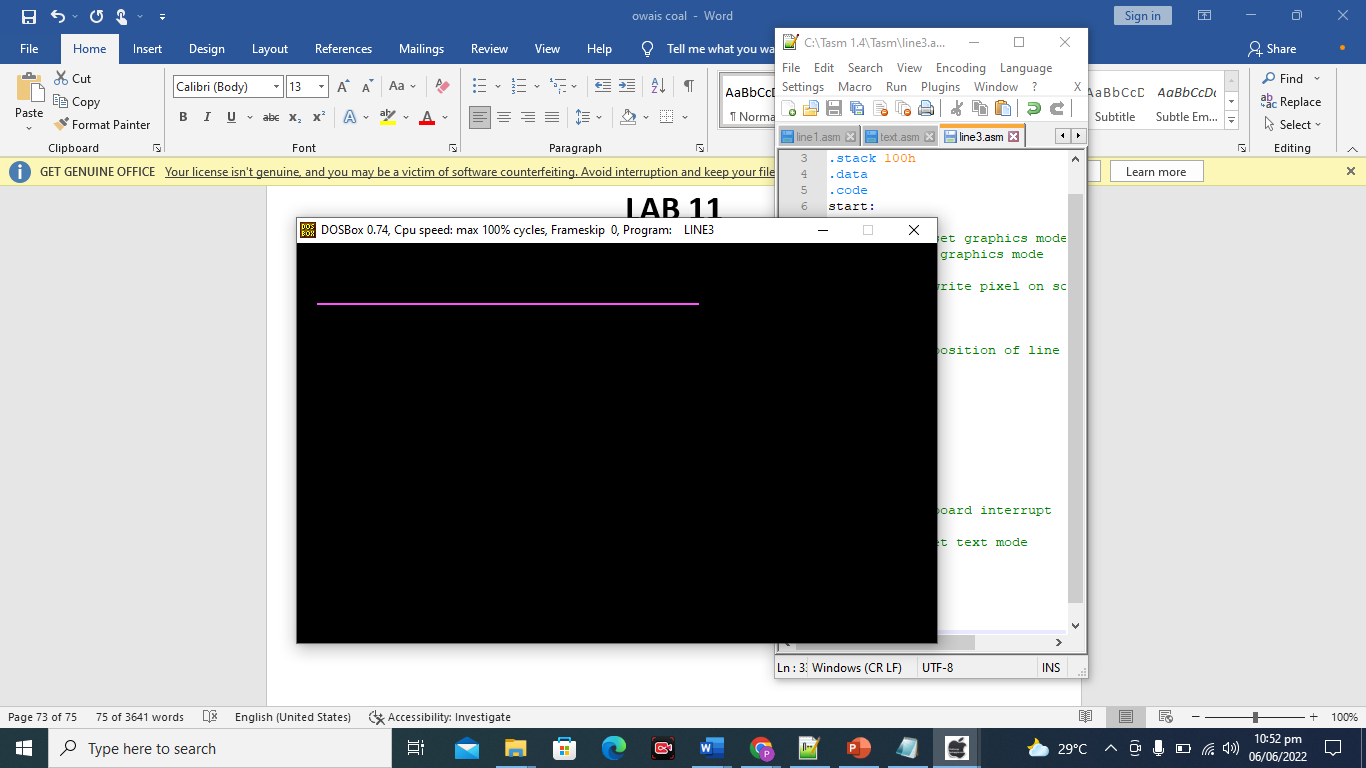
int 10h

mov ah, 4ch

int 21h

end start

OUTPUT



Change length of the line in above program.

.model small

.stack 100h

.data

.code

start:

mov ax, 13 ; set graphics mode

int 10h ; set graphics mode

mov ah, 0ch ;write pixel on screen

mov al, 13

mov cx, 10

mov dx, 20

l1:

int 10h

inc cx

cmp cx,150 ; length changes here

jle l1

mov ah, 0

int 16h ; keyboard interrupt

mov ax, 3 ; set text mode

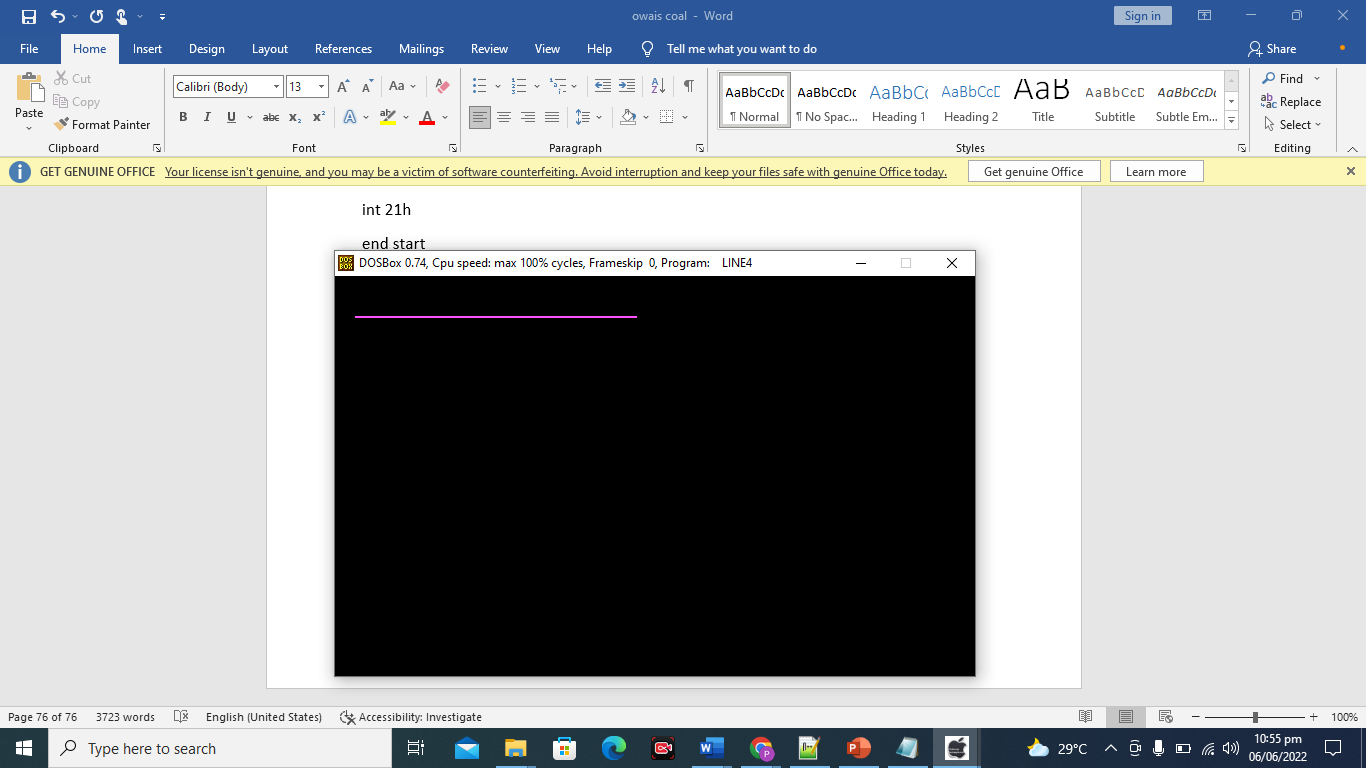
int 10h

mov ah, 4ch

int 21h

end start

OUTPUT



**Create any shape on graphics mode**

. .model small

.stack 100h

.data

.code

start:

mov ax,6

int 10h

mov ah,0ch

mov al,1

mov cx,100

mov dx,50

l1:

int 10h

inc dx

cmp dx,150

jle l1

mov ah,0ch

mov al,1

mov cx,100

mov dx,50

l2:

int 10h

inc cx

cmp cx,550

jle l2

mov ah,0ch

mov al,1

mov cx,550

mov dx,50

l3:

int 10h

inc dx

cmp dx,150

jle l3

mov ah,0ch ; nechay wali line left to right

mov al,1

mov cx,100

mov dx,150

l4:

int 10h

inc cx

cmp cx,550

jle l4

mov ah,0ch ; andhar wali line

mov al,1

mov cx,150

mov dx,70

l5:

int 10h

inc cx

cmp cx,550

jle l5

mov ah,0ch ; andhar wali line

mov al,1

mov cx,150

mov dx,60

l6:

int 10h

inc cx

cmp cx,550

jle l6

mov ah,0ch ; left side par jo line h

mov al,1

mov cx,150

mov dx,50

l7:

int 10h

inc dx

cmp dx,150

jle l7

mov ah,0ch ; j ki top wali line

mov al,1

mov cx,250

mov dx,90

l8:

int 10h

inc cx

cmp cx,425

jle l8

mov ah,0ch ; j right side wali line

mov al,1

mov cx,350

mov dx,90

l9:

int 10h

inc dx

cmp dx,140

jle l9

mov ah,0ch ; j ki nechay wali line

mov al,1

mov cx,300

mov dx,140

l10:

int 10h

inc cx

cmp cx,350

jle l10

mov ah,0ch ; j ki last ki choti wali line

mov al,1

mov cx,300

mov dx,120

l11:

int 10h

inc dx

cmp dx,140

jle l11

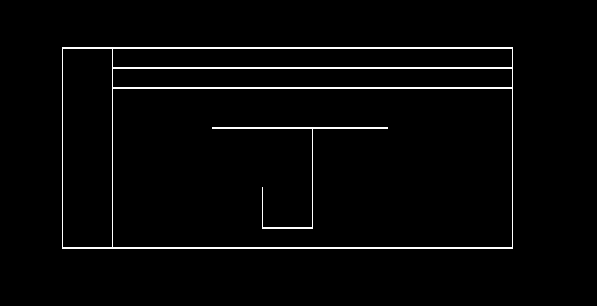
mov ah,0

int 16h

mov ah,4ch

int 21h

end start



**LAB 13**

# **GRAPHIC MODE (text & background)**

**Change position of character to center.**

.model small

.stack 100h

.data

.code

start:

mov ah, 00

mov al, 03

int 10H

mov dh, 12

mov dl, 40

mov ah, 2

mov bh, 0

int 10h

mov dl, 'A'

mov ah, 02

int 21h

mov ah, 0

int 16h

mov ah, 4ch

int 21h

end start

**Change background color with changing palate.**

.model small

.stack 100h

.data

.code

start:

mov ah, 6

mov al, 0

mov cx, 0

mov dx, 184fh

mov bh, 00110000b

int 10h

mov ah, 0

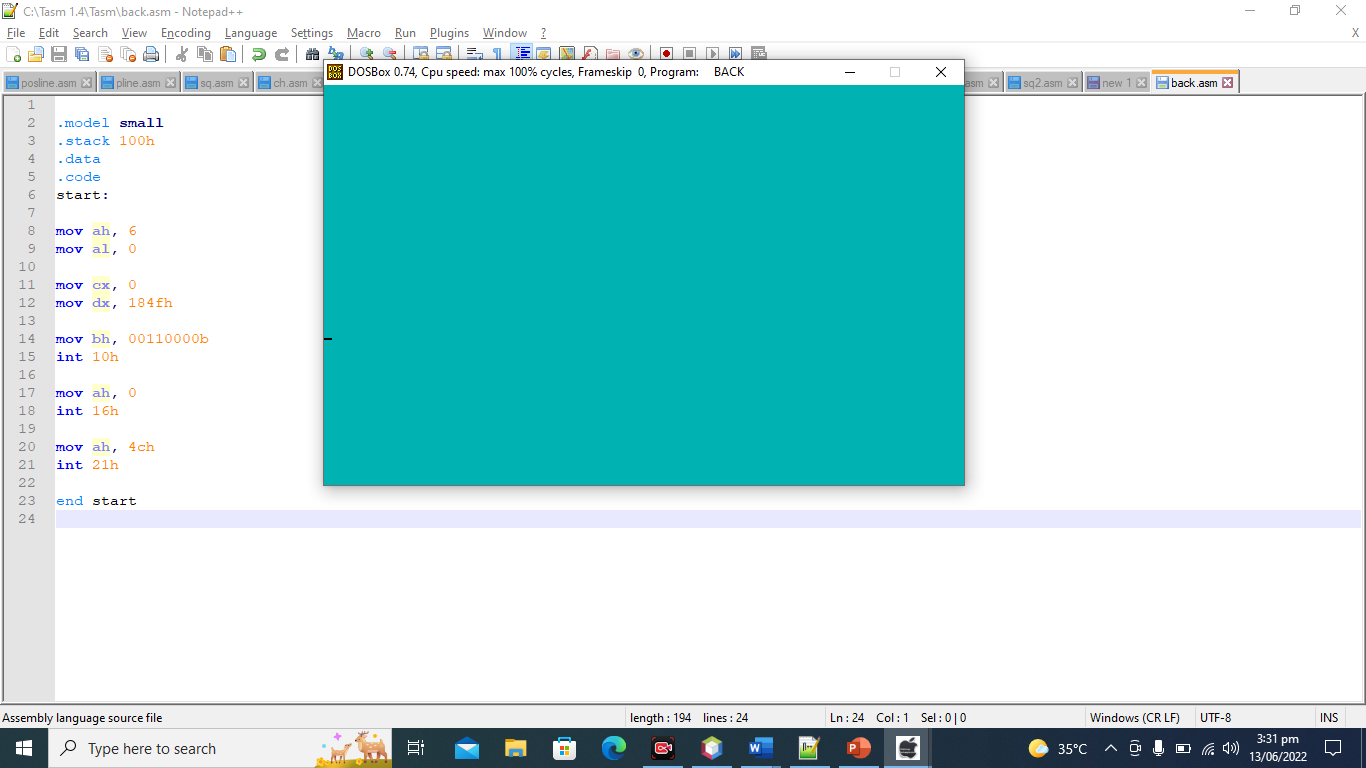
int 16h

mov ah, 4ch

int 21h

end start

**OUTPUT**



**Change font color.**

SOURCE CODE;-

.model small

.stack 100h

.data

var db 'OWAIS AMIR$'

.code

start:

mov ax, @data

mov ds, ax

mov ah, 09 ; write character and attribute at cursor position.

mov cx, 15 ; number of times to write character

mov bl, 00001011b ;color attribute - 1000 1111 (bright blinking)

int 10h

lea dx, var

mov ah, 09

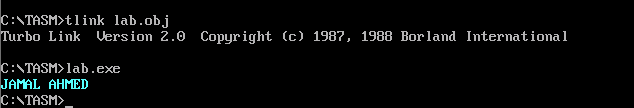
int 21h

mov ah, 4ch

int 21h

end start

**OUTPUT;-**

****

**Display your name**

**SOURCE CODE;-**

.model small

.stack 100h

.data

var db 'OWAIS AMIR$'

.code

start:

mov ax, @data

mov ds, ax

mov ah, 09 ; write character and attribute at cursor position.

mov cx, 15 ; number of times to write character

mov bl, 00001011b ;color attribute - 1000 1111 (bright blinking)

int 10h

lea dx, var

mov ah, 09

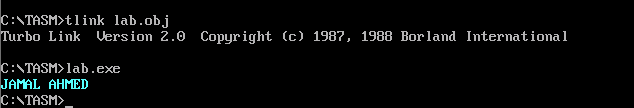
int 21h

mov ah, 4ch

int 21h

end start

**OUTPUT;-**



**LAB 14**

# **ARITHMATICS(MUL-DIV)**

**Program: 1 Create a program of using Mul mnemonic.**

.model small

.stack 100h

.data

msg1 db 10, 13 , " Enter Multiplicand: $"

msg2 db 10,13 , " Enter Multipiler: $"

msg3 db 10,13, " Result:$"

num1 db ?

num2 db ?

result db ?

.code

start:

mov ax,@data

mov ds,ax

mov ah,09

lea dx, msg1

int 21h

mov ah,01

int 21h

sub al, 30h

mov num1,al

mov ah,09

lea dx,msg2

int 21h

mov ah,01

int 21h

sub al,30h

mov num2,al

mul num1

mov result,al

aam

add ah,30h

add al,30h

mov bx,ax

mov ah,09

lea dx,msg3

int 21h

mov ah,02

mov dl, bh

int 21h

mov ah,02

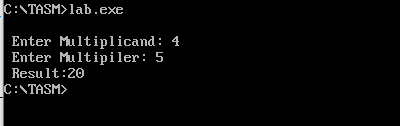
mov dl, bl

int 21h

mov ah, 4ch

int 21h

end start



**Program: 2 Create a program of using DIV mnemonic.**

.MODEL SMALL

.STACK 2000

.DATA

MSGA DB 13,10,"Input first number: ","$"

MSGB DB 13,10,"Input second number: ","$"

MSGC DB 13,10,"The quotient is: ","$"

MSGD DB 13,10,"The modulo is: ","$"

NUM1 db ?

NUM2 db ?

.CODE

MAIN PROC NEAR

MOV AX, @DATA

MOV DS, AX

; get first number

LEA DX, MSGA

MOV AH, 09h

INT 21h

MOV AH, 01

INT 21H

SUB AL, '0'

MOV BL, AL

; get second number

LEA DX, MSGB

MOV AH, 09h

INT 21h

MOV AH, 01

INT 21H

SUB AL, '0'

MOV CL, AL

; divide

MOV AH, 0 ; prepare dividend

MOV AL, BL

DIV CL

MOV NUM1, AL

ADD NUM1, '0'

MOV NUM2, AH

ADD NUM2, '0'

; output quotient

LEA DX, MSGC

MOV AH, 09h

INT 21h

MOV DL, NUM1

MOV AH, 02H

INT 21h

; output remainder/modulo

LEA DX, MSGD

MOV AH, 09h

INT 21h

MOV DL, NUM2

MOV AH, 02H

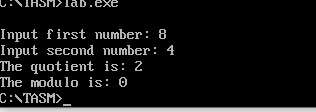
INT 21h

MOV AH, 4Ch

INT 21h

MAIN ENDP

END MAIN



**LAB 15**

# **FILE HANDLING**

**CREATE A FILE WHICH IS CREATE IN TASM**

**CODE:**

.model small

.stack 100h

.data

f\_name db 'jamal.txt',0

address dw ?

.code

start:

mov ax,@data

mov ds,ax

mov ah,3ch

lea dx,f\_name

mov cl,0

int 21h

mov address,ax

mov ah,3ch

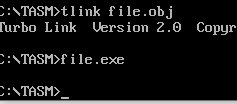
lea dx,address

int 21h

mov ah,4ch

int 21h

end start

****

**WRITE IN A FILE WHICH IS CREATE IN TASM**

**CODE:**

.model small

.stack 100h

.data

f\_name db 'JAMAL.txt',0

address dw ?

msg db "jamal$"

.code

start:

mov ax,@data

mov ds,ax

mov ah,3dh

lea dx,f\_name

mov al,2

int 21h

mov address,ax

mov ah,40h

lea bx,address

mov cx,5

lea dx,msg

int 21h

mov ah,3eh

mov bx,address

mov ah,4ch

int 21h

end start

**WRITE IN STRING TO A FILE WHICH IS CREATE IN TASM**

**CODE**

.model small

.stack 100h

.data

f\_name db 'JAMAL.txt',0

address dw ?

msg db "ENTER THE DATA $"

buffer db 100 dup('$')

.code

start:

mov ax,@data

mov ds,ax

mov ah,3dh

lea dx,f\_name

mov al,2

int 21h

mov address,ax

mov ah,09

lea dx,msg

int 21h

mov si,0

mov cx,0

again:

mov ah,01

int 21h

cmp al,13

je exit

mov buffer[si],al

inc si

inc cx

jmp again

exit:

mov ah,40h

lea bx,address

lea dx,buffer

int 21h

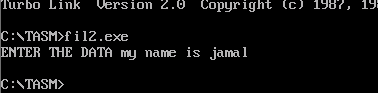
mov ah,3eh

mov bx,address

mov ah,4ch

int 21h

end start



**DELETE A FILE WHICH IS CREATE IN TASM**

**CODE**

.model small

.stack 100h

.data

fname db 'C:\tasm 1.4\tasm\JAMAL.txt',0

msg db "FILE DELETED$"

.code

start:

mov ax,@data

mov ds,ax

lea dx,fname

mov ah,41h

jc exit

mov ah,09

lea dx,msg

int 21h

exit:

mov ah,4ch

int 21h

end start

