

**Department of Computer Science**

**Lab Manual for**

**Computer Organization and Assembly language**

**B.S (CS)**

**Semester: III**

**CourseSupervisor*:* Miss Sahar Zafar**

STUDENT NAME: HAFIZ MUDASSIR HUSAIN

STUDENT ID: CSC-22F-145

SECTION: 3C

CERTIFICATE

**Department of Computer Science**

**Computer Organization and Assembly Lanaguage**

This is to certify that Mr/Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ So/Do \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ having Roll No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ has successfully completed Laboratory work during Spring Semester 2023.

Lab Instructor Course Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature Signature

**LAB 1**

**Create program with ASCII Code**

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

. Model Small

. Stack 100H

. Data

. Code

Start:

Mov ah, 02

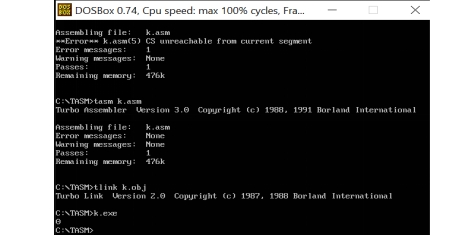
Mov dl, ”00110000b”

Int 21h

Mov ah, 4ch

Int 21h

End Start



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

. Model Small

. Stack 100H

. Data

. Code

Start:

Mov ah, 02

Mov dl, ”01000001b”

Int 21h

Mov ah, 4ch

Int 21h

End Start



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

.model small

.stack 100h

.data

.code

start:

mov ah,02

mov dl,'S'

int 21h

mov ah,02

mov dl,'A'

int 21h

mov ah,02

mov dl,'A’'

int 21h

mov ah,02

mov dl,'D'

int 21h

mov ah,4ch

int 21h

end start



**Lab 2**

**Basic DOS Functions**

* **Create a code for given output**

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

**A**

**A**

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

.model small

.stack 100h

.data

.code

start:

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

Mov ah,02

Mov dl, 0AH

Int 21h

mov ah,02

mov dl, 'A'

int 21h

Mov ah,02

Mov dl, 0AH

Int 21h

mov ah,02

mov dl, 'A'

int 21h

Mov ah,02

Mov dl, 0AH

Int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

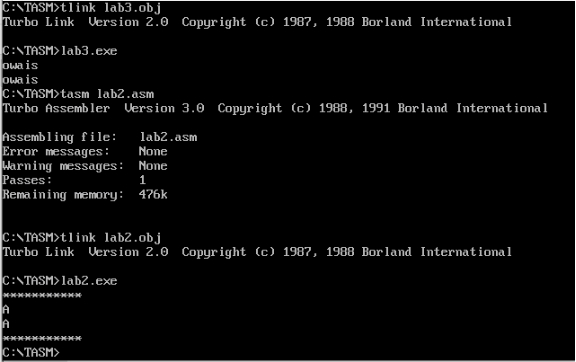
mov dl, '\*'

int 21h

mov ah,4ch

int 21h

end start



**Lab 3**

**Variable**

* **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**

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

.model small

.stack 100h

.data

var db 'Name: $'

var1 db 10,13,'Father Name: $'

var2 db 10,13,'Roll Number: $'

var3 db 10,13,'University: $'

.code

start:

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

Mov ah,02

Mov dl, 0AH

Int 21h

mov ax,@data

mov ds,ax

mov ah,09

lea dx,var ; mov dx,offset var

int 21h

mov ah,3fh

int 21h

mov ah,09

lea dx,var1 ; mov dx,offset var1

int 21h

mov ah,3fh

int 21h

mov ah,09

lea dx,var2 ; mov dx,offset var2

int 21h

mov ah,3fh

int 21h

mov ah,09

lea dx,var3 ; mov dx,offset var3

int 21h

mov ah,3fh

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,02

mov dl, '\*'

int 21h

mov ah,4ch

int 21h

end start



**Lab 4**

**Case conversation**

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

.model small

.stack 100h

.data

Msg1 db "ENTER NAME IN SMALL: $"

Msg2 db 10,13, "YOUR NAME IS CONVERTED INTO CAPITAL: $"

Msg3 db 10,13, "ENTER ID IN Capital: $"

Msg4 db 10,13, "YOUR ID IS CONVERTED INTO CAPITAL: $"

.code

start:

mov ax,@data

mov ds,Ax

mov ah,09

mov dx,offset msg1

int 21h

mov ah,01

int 21h

sub al,20h

mov ah,09

mov dx,offset msg2

int 21h

mov ah,02

mov dl,al

int 21h

mov ah,09

mov dx,offset msg3

int 21h

mov ah,01

int 21h

ADD al,20h

mov ah,09

mov dx,offset msg4

int 21h

mov ah,02

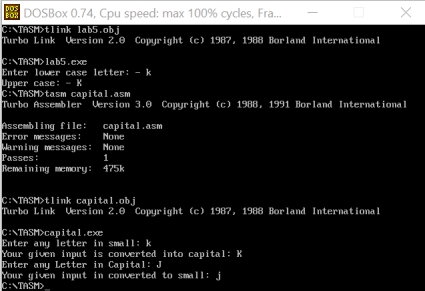
mov dl,al

int 21h

mov ah,4ch

int 21h

end start



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

1. **Create a program using EQU**

.Model Small

.Stack 100h

.data

A equ 41h

Newline equ 0AH

Exit equ 4ch

Print equ 02

.code

Start:

mov ah, Print

mov dl, A

int 21h

mov ah,02

mov dl, Newline

int 21h

mov ah, Print

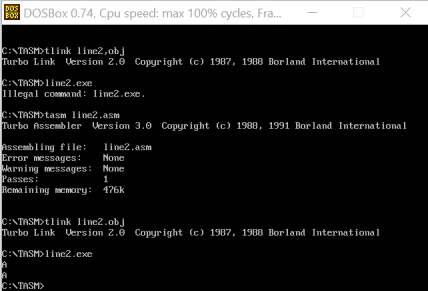
MOV dl, A

int 21h

mov ah, Exit

int 21h

end



**Lab 5**

**Unconditional Loop**

* **Task # 1: Write a Program to Display ASCII characters from A to Z through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,65 ;'A'

l1:

mov ah,02

int 21h

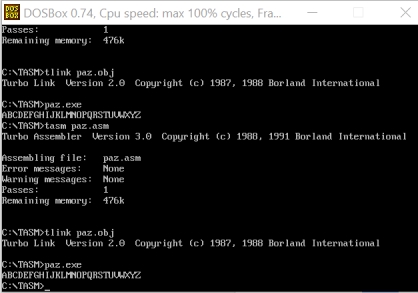
inc dl

loop l1

mov ah,4ch

int 21h

end start



* **Task # 2: Write a Program to Display ASCII characters from a to z through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,97 ;'a'

l1:

mov ah,02

int 21h

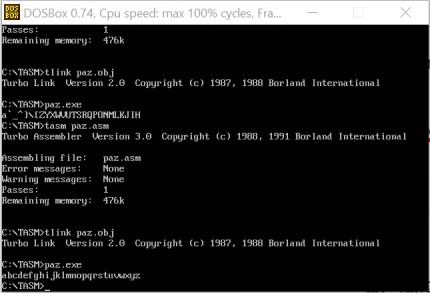
inc dl ;inc dl

loop l1

mov ah,4ch

int 21h

end start



* **Task # 3: Write a Program to Display ASCII characters from Z to A through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,90

l1:

mov ah,02

int 21h

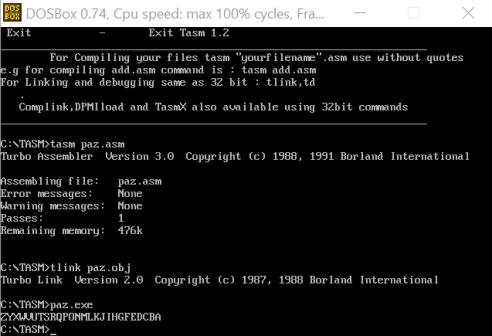
dec dl

loop l1

mov ah,4ch

int 21h

end start



* **Task # 3: Write a Program to Display ASCII characters from 0 to 9 through Looping.**

.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



* **Task # 4: Write a Program to Display ASCII characters from 9 to 0 through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,10

mov dl,57

l1:

mov ah,02

int 21h

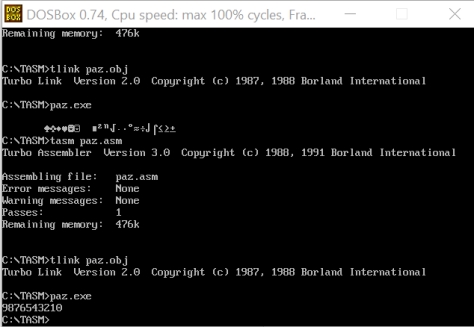
DEC dl

loop l1

mov ah,4ch

int 21h

end start



* **Task # 5: Write a Program to Display ASCII characters from z to a through Looping.**

.model small

.stack 100h

.data

.code

start:

mov cx,26

mov dl,122 ;'z'

l1:

mov ah,02

int 21h

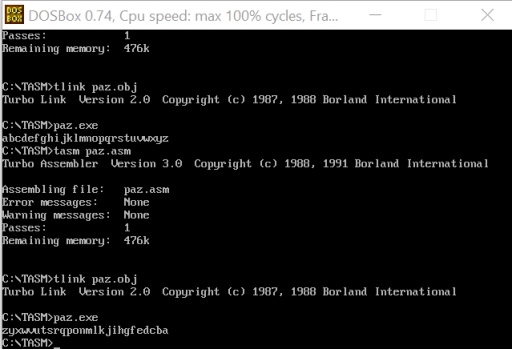
dec dl

loop l1

mov ah,4ch

int 21h

end start



* **Task # 7: Write a Program to Display ASCII characters from A-Z & z-a   
   through Looping.**

.model small

.stack 100h

.data

.code

start:

mov ah,02h

mov cx,26

mov dl,65

l1:

Int 21h

inc dl

loop l1

mov ah,02

mov cx,26

mov dl,122

l2:

Int 21h

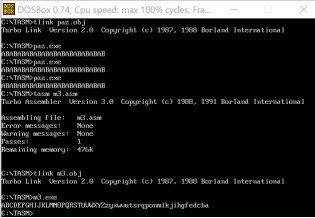
dec dl

loop l2

mov ah,4ch

int 21h

end start



**Task # 8: Write a Program to Display ASCII characters from Z-A & a-z   
 through Looping.**

.model small

.stack 100h

.data

.code

start:

mov ah,02h

mov cx,26

mov dl,90

l1:

Int 21h

dec dl

loop l1

mov ah,02

mov cx,26

mov dl,97

l2:

Int 21h

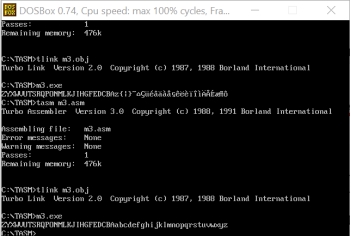
inc dl

loop l2

mov ah,4ch

int 21h

end start



* **Task # 9: Write a Program to print your name 10 times.**

.model small

.stack 100h

.data

msg db 'm.ikramah' ,10,13, '$'

.code

start:

mov ax,@data

mov ds,ax

mov cx,10

print:

mov ah,9

lea dx,msg

int 21h

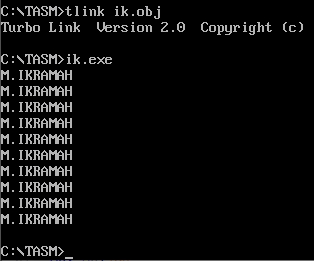
dec cx

jnz print

mov ah,4ch

int 21h

end start



* **Task # 10: Write a program in Assembly Language to print following pattern:0iiiiii0iiiiii0iiiiii0iiiiii0**

.model small

.stack 100h

.data

var db"0iiiiiii$"

.code

start:

mov ax,@data

mov ds,ax

mov cx,4

l1:

mov dx,offset var

mov ah,09

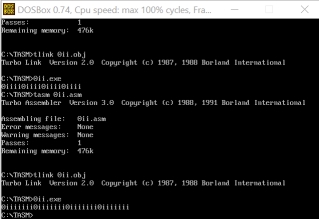
int 21h

loop l1

mov ah,4ch

int 21h

end start



**Lab 6**

**Conditional Jumps**

* **Create program using different jump category given in theory lecture.**

.model small

.stack 100h

.data

.code

start:

mov al,2

mov bl,0

cmp al,bl

jg=greater

mov ah,02

mov dl,'F'

int 21h

mov ah,4ch

int 21h

greater:

mov ah,02

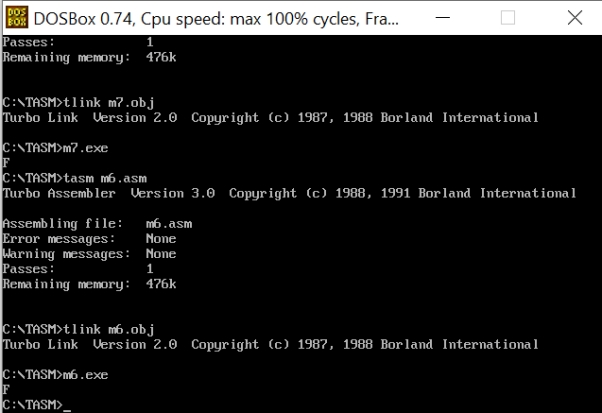
mov dl,'T'

int 21h

mov ah, 4ch

int 21h

end start



**Lab 7**

**Branching , Nested loop, Repeat Until**

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

.model small

.stack 100h

.code

start:

mov ah,01

repeat:

int 21h

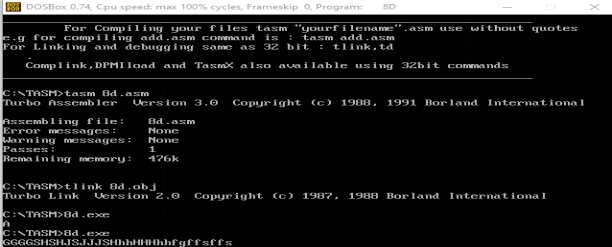
cmp al, 'A'

jne repeat

mov ah,4ch

int 21h

end start



**While loop**

.model small

.stack 100h

.data

.code

start:

mov dx,0

mov ah,01

int 21h

while\_:

cmp al, 0dh

je end\_while

inc dx

int 21h

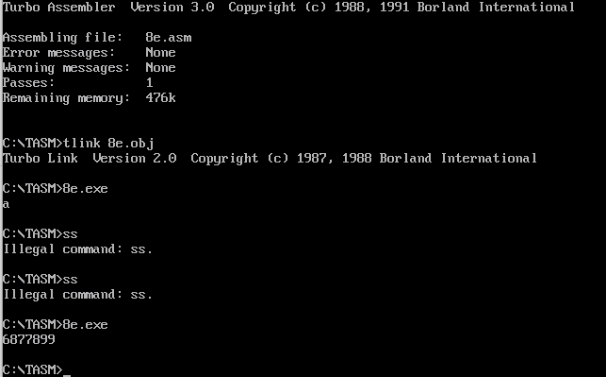
jmp while\_

end\_while:

mov ah, 4ch

int 21h

end start



**IF THEN**

**( if condition is true it print e otherwise it terminate)**

.model small

.stack 100h

.code

start:

mov al,49

CMP al, 49

Je then ; no, exit

NEG AX ; yes, change sign

jmp exit

then:

mov ah, 02

mov dl, 'E'

int 21h

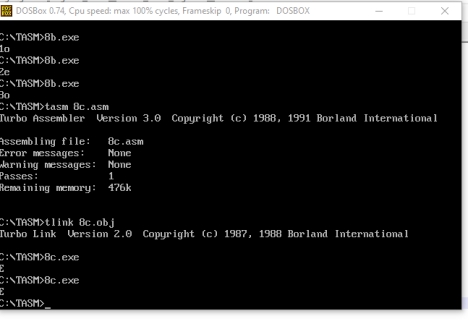
jmp exit

exit:

mov ah,4ch

int 21h

end start



**IF THEN ELSE**

**CASE**

.model small

.stack 100h

.code

start:

mov ah,01

int 21h

cmp al,49

je odd

cmp al,51

je odd

cmp al,50

je even1

cmp al,52

je even1

jmpend\_case

even1:

mov dl, 'e'

jmp display1

odd:

mov dl ,'o'

jmp display1

display1:

mov ah,02

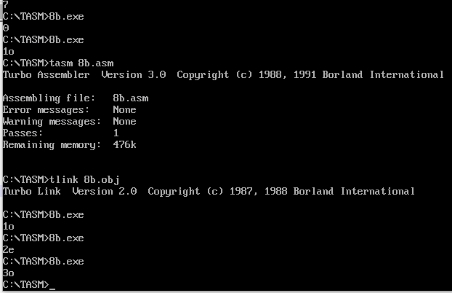
int 21h

end\_case:

mov ah, 4ch

int 21h

end start



* **OR (Create a program using OR condition) OR = when both condition are true**

.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

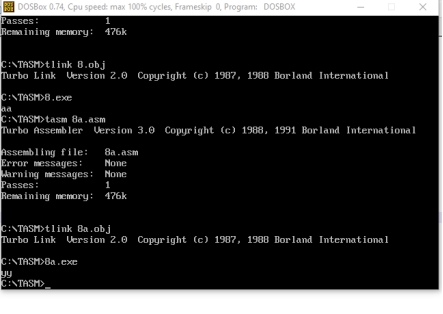
jmpendif\_

endif\_:

mov ah,4ch

int 21h

end start



* **AND (create a program using And logic means when both conditions are true than program will execute otherwise it terminates)**

.model small

.stack 100h

.data

.code

start:

mov ah,01

int 21h

cmpal,'A'

je endif\_

cmp al, 'Z'

je endif\_

mov dl, al

mov ah ,02

int 21h

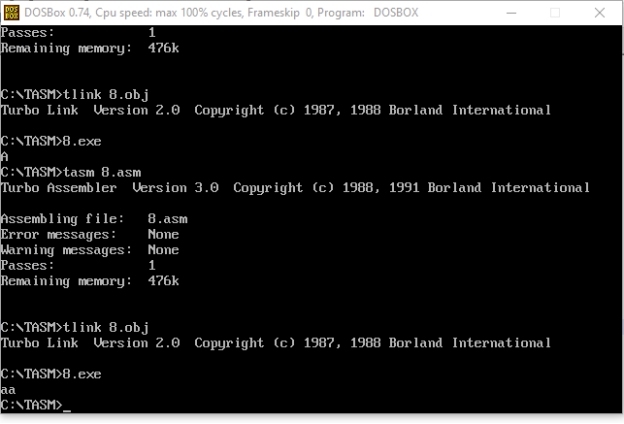
jmpendif\_

endif\_:

mov ah,4ch

int 21h

end start



* Task : create pattern with help of nested loop Program:

.model small

.stack 100h

.code

start:

mov cx, 4

l1:

mov bx,cx

l2:

mov dl,'\*'

mov ah,02

int 21h

loop l2

mov ah, 02

mov dl, 0ah

int 21h

mov ah, 02

mov dl, 0dh

int 21h

mov cx,bx

loop l1

mov ah, 4ch

int 21h

end start



**Lab 8**

**String Instruction**

* **Program 1 : Compare two strings and display message accordingly in Assembly Language.**

.model small

.stack 100h

.data

msg1 db "This is 1st string$"

msg2 db "This is 2nd string$"

cmp1 db 'Both the strings are equal$'

cmp2 db 'both the strings are not equal$'

.code

start:

Mov ax,@data

Mov ds, ax

mov ah,09

lea dx,msg1

int 21h

mov dl,10

mov ah,02

int 21h

mov ah,09

lea dx ,msg2

int 21h

mov dl,10

mov ah,02

int 21h

mov al, msg1

mov bl,msg2

cmp al,bl

je equal

cmp al,bl

jne not\_equal

not\_equal:

mov ah,09

lea dx,cmp2

int 21h

equal:

mov ah,09

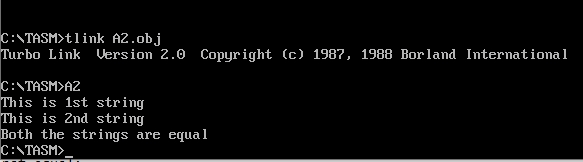
lea dx,cmp1

int 21h

mov ah,4ch

int 21h

end start



* **Program 2:Take input a string and print it on screen.**

.model small

.stack 100h

.data

var1 db 100 dup('$')

.code

begin:

mov ax,@data

mov ds,ax

mov si,offset var1

l1:

mov ah,01

int 21h

cmp al,13 ; Ascii of enter key

je pend

mov [si],al

inc si

jmp l1

pend:

mov dx, offset var1

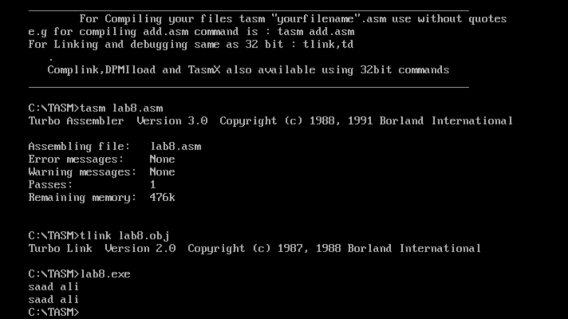
mov ah,09

int 21h

mov ah, 4ch

int 21h

end begin



**Program 3: Reverse the string.**

.model small

.stack 100h

.data

text1 db 'hello$'

text2 db 5 dup('0$')

.code

begin:

mov ax,@data

mov ds,ax

mov es,ax

lea si,text1+4

lea di,text2

std

mov cx,5

move:

movsb

add di,2

loop move

mov dx, offset text2

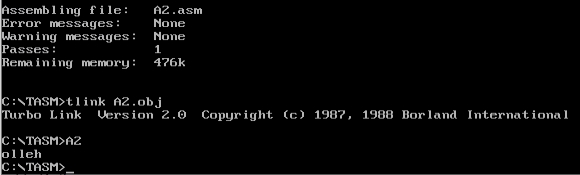
mov ah,09

int 21h

mov ah, 4ch

int 21h

end begin



* **program 4: Copy String using Movesb.**

.model small

.stack 100h

.data

text1 db 'hello$'

text2 db 5 dup('0$')

.code

begin:

mov ax,@data

mov ds,ax

mov es,ax

lea si,text1

lea di,text2

cld

mov cx,5

rep movsb

mov dx, offset text2

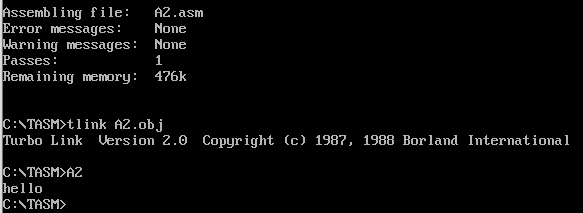
mov ah,09

int 21h

mov ah, 4ch

int 21h

end begin



* **Program 5: Create a program using string instruction stosb to store content in variable from al register.**

.model small

.stack 100h

.data

text1 db 'hello$'

text2 db 5 dup('$')

.code

begin:

Mov ax ,@data

Mov es,ax

lea di, text1

Cld

Mov al,'A'

stosb

mov dx, offset text1

mov ah,09

int 21h

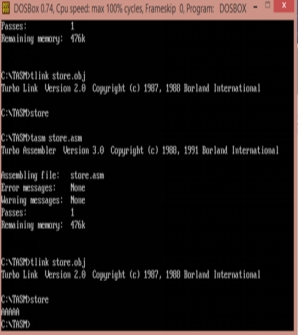
mov ah, 4ch

int 21h

end begin

* **Program 6: Create code which store A 5 times in variable which already contain any string using stosb string instruction.**

.model small

.stack 100h

.data

str1 db "hello$"

str2 db 5 dup('0$')

.code

start:

mov ax,@data

mov es,ax

mov ds,ax

cld

mov cx,5

mov al,'A'

lea di, str1

repne stosb

mov ah,09

lea dx, str1

int 21h

mov ah,4ch

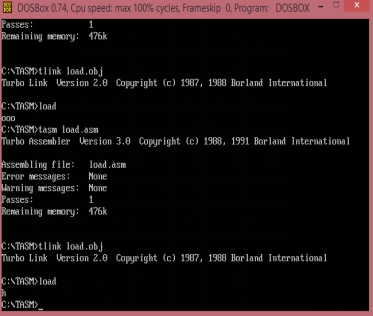
int 21h

end start

* **Program 7 : Create program using load instruction.**

.model small

.stack 100h

.data

str1 db "hello$"

.code

start:

mov ax,@data

mov es,ax

mov ds,ax

lea si,str1

cld

lodsb

mov ah,02

mov dl, al

int 21h

mov ah,4ch

int 21h

end start

* **Program 8 : Create program using Scan string operation**.

.model small

.stack 100h

.data

var db "A$"

.code

start:

mov ax,@data

mov ds, ax

mov es, ax

lea di, var

cld

mov al,'A'

scasb

jz equal

mov dl,49

mov ah,02

int 21h

jmp close

equal:

mov dl, 48

mov ah,02

int 21h

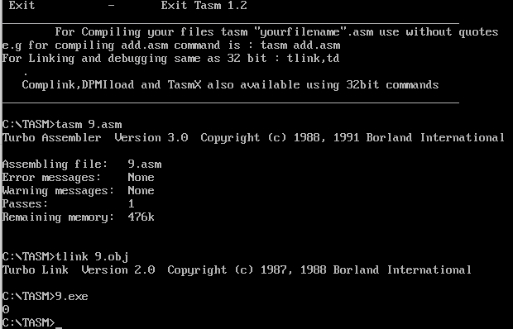
jmp close

close:

mov ah, 4ch

int 21h

end start



**Lab 9**

**Bit Manipulation**

* **Program for SHL**

.model small

.stack 100h

.code

main proc

mov dx,4

shl dx,1

add dx,48

mov ah,2

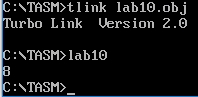
int 21h

mov ah,4ch

int 21h

main endp

end main



* **Program for SHR**

.model small

.stack 100h

.code

main proc

mov dx,4

shr dx,1

add dx,48

mov ah,2

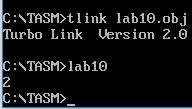
int 21h

mov ah,4ch

int 21h

main endp

end main



* **Logical Instruction Program for AND**

.model small

.stack 100h

.code

start:

mov ax,2

mov bx,4

and ax,bx

add ax,48

mov ah,02

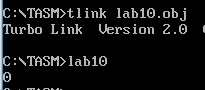
mov dx,ax

int 21h

mov ah,4ch

int 21h

end start



**Program for OR**

.model small

.stack 100h

.code

start:

mov bl,101b

or bl,110B

add bl,48

mov ah,02

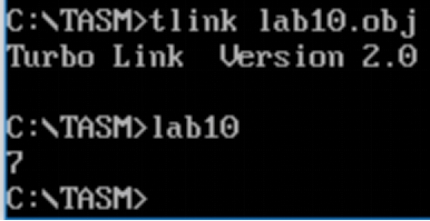
mov dl,bl

int 21h

mov ah,4ch

int 21h

end start



* **Program for XOR**

.model small

.stack 100h

.code

start:

mov bl,101b

xor bl,110B

add bl,48

mov ah,02

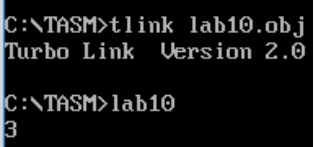
mov dl,bl

int 21h

mov ah,4ch

int 21h

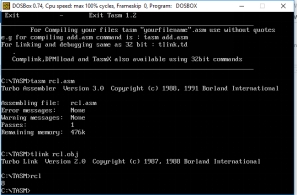
end start



* **RCL /RCR**

.model small

.stack 100h

.code

start:

clc

mov ax,08

rcl ax,1

add bl,48

mov ah,02

mov dl,bl

int 21h

mov ah,4ch

int 21h

end start

* **ROL/ROR**

.model small

.stack 100h

.code

main proc

mov dx,4

ror dx,1

add dx,48

mov ah,2

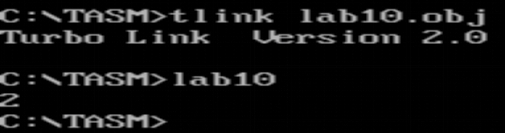
int 21h

mov ah,4ch

int 21h

main endp

end main



**Lab 10**

**Graphics**

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

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

**Create Parallel line on screen.**

**Create any shape on graphics mode. (Square, Triangle, rectangle)**

**TASK 01**

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

.model small

.stack 100h

.data

.code

start:

mov ah,0

mov al,6

int 10h

mov ah,0ch

mov al,1

mov cx,50

mov dx,50

l1:

int 10h

inc cx

cmp cx,280

jle l1

mov ah,0

int 16h

mov ah,0

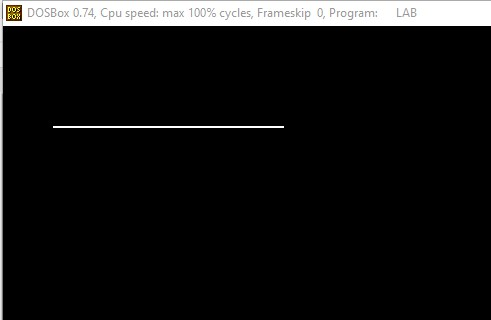
mov al,3

int 10h

mov ah,4ch

int 21h

end start



**TASK 2**

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

.model small

.stack 100h

.data

.code

start:

mov ah,0

mov al,6

int 10h

mov ah,0ch

mov al,1

mov cx,80

mov dx,80

l1:

int 10h

inc cx

cmp cx,150

jle l1

mov ah,0

int 16h

mov ah,0

mov al,3

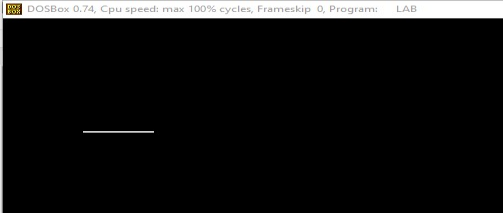
int 10h

mov ah,4ch

int 21h

end start

**output**



**TASK 3**

* **Create Parallel line on screen**.

.model small

.stack 100h

.data

.code

start:

mov ah,0

mov al,6

int 10h

mov ah,0ch

mov al,1

mov cx,90

mov dx,50

l1:

int 10h

inc Dx

cmp Dx,150

jle l1

mov ah,0ch

mov al,1

mov cx,120

mov dx,50

l2:

int 10h

inc Dx

cmp Dx,150

jle l2

mov ah,0

int 16h

mov ah,0

mov al,3

int 10h

mov ah,4ch

int 21h

end start



**TASK 4**

* **Create any shape on graphics mode. (Square, Triangle, rectangle)**

**SQUARE:**

.model small

.stack 100h

.data

.code

start:

mov ah,0

mov al,6

int 10h

mov ah,0ch

mov al,1

mov cx,90

mov dx,50

l1:

int 10h

inc Dx

cmp Dx,150

jle l1

mov ah,0ch

mov al,1

mov cx,400

mov dx,50

l2:

int 10h

inc Dx

cmp Dx,150

jle l2

mov ah,0ch

mov al,1

mov cx,90

mov dx,5

l3:

int 10h

inc Cx

cmp Cx,400

jle l3

mov ah,0ch

mov al,1

mov cx,90

mov dx,150

l4:

int 10h

inc Cx

cmp Cx,400

jle l4

mov ah,0

int 16h

mov ah,0

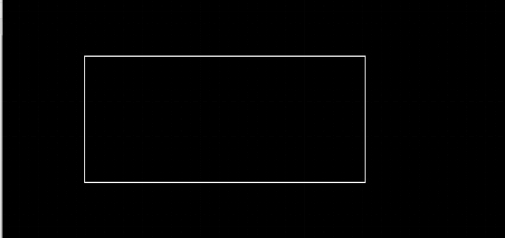
mov al,3

int 10h

mov ah,4ch

int 21h

end start



* **TRIANGLE:**

.model small

.stack 100h

.data

.code

start:

mov ah,0

mov al,6

int 10h

mov ah,0ch

mov al,1

mov cx,301

mov dx,150

l1:

int 10h

inc cx

cmp cx,400

jle l1

mov ah,0ch

mov al,1

mov cx,350

mov dx,100

l2:

int 10h

inc dx

cmp dx,150

inc cx

cmp cx,400

jle l2

mov ah,0ch

mov al,1

mov cx,350

mov dx,100

l3:

int 10h

dec cx

cmp cx,280

inc dx

cmp dx,150

jle l3

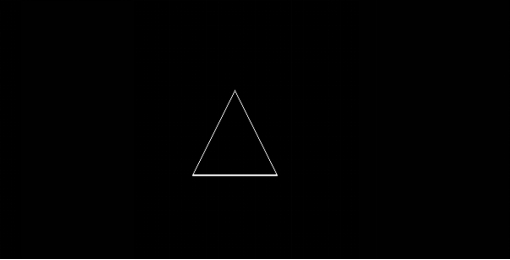
mov ah,0

int 16h

mov ah,4ch

int 21h

end start



**LAB 12**

**Graphics**

**Program#1 : Create a program to print a line on screen**

.model small

.stack 100h

.code

mainproc

;set graphic mode

Mov ax,6

int 10h ; int 10h used for screen manipulation

;draw line

Mov ah,0ch

Mov al,1

Mov cx,301

mov dx,100

L1:

int 10h

inc cx

cmp cx,600

jle l1

; read keyboard

Mov ah,0

int 16h

Mov ax,3

int 10h

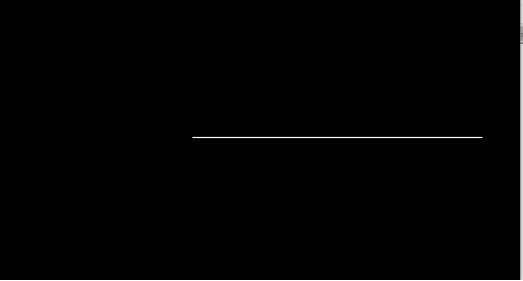
return dos

Mov ah,4ch

int 21h

mainendp

end main



* **Program 2: Create a program to print a character in color on screen**

.model small

.stack 100h

.code

start:

Mov ah, 0 ; set mode

mov al,04h ; mode 4

int 10h

mov ah,0bh ; function 0BH for background

mov bh,00h ; select background color

mov bl,3 ; blue

int 10h

mov ah,02 ; set cursor

mov bh,0 ; page 0

mov dh,0 ; row 0

mov dl,30 col 39

int 10h

mov ah,09 ; write character function

moval,'A'

mov bl,2 ; red color

mov cx,1 ; write 1 character

int 10h

mov ah,4ch

int 21h

end start



* **Program #3 create a program which display 256 color**

.model small

.stack 100h

.code

start:

; set mode

mov ah,0

mov al, 13h

int 10h

; display 256 pixel in row 100

mov ah ,0ch ; write pixel function

mov al, 0 ; start with pixel color 0

mov bh, 0 ; page 0

mov cx, 0 ; column 0

mov dx, 100h ; row 100

l1:

int 10h ; write pixel

inc al ; next color

inc cx ; next column

cmp cx, 256 ; finished?

jl l1 ; no repeat

mov ah, 4ch

int 21h

end start



* **TITLE: "Program to clear screen"**

.model small ; small memory model

.stack 100h ; 256 Byte memory is allocated for stack memory

.code ; start of code memory

main proc

mov ah,06h ;window scroll up

mov al,0 ;scroll entire window

mov ch,00 ;upper left row window

mov cl,00 ;upper left column window

mov dh,24 ;lower right row window

mov dl,79 ;lower right column window

mov bh,07h ;normal video attribute (bh=attribute for blank lines)

int 10h ;clear screen

mov ah,4ch

int 21h

main endp

end main

\

* **TITLE: "Program to set cursor size"**

.model small ; small memory model

.stack 100h ; 256 Byte memory is allocated for stack memory

.code ; start of code memory

main proc

mov ah,01h ;set cursor size

mov ch,00h ;set cursor top line

mov cl,07h ;set cursor bottom line

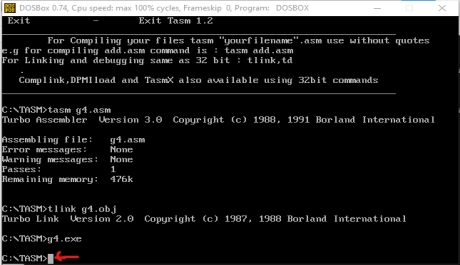
int 10h ;display cursor

mov ah,4ch

int 21h

main endp

end main



* **Program : Set the position of the cursor.**
* **input:**
* **DH = row.**
* **DL = column.**
* **BH = page number (0..7).**

.model small

.stack 100h

.code

start:

mov dh, 10

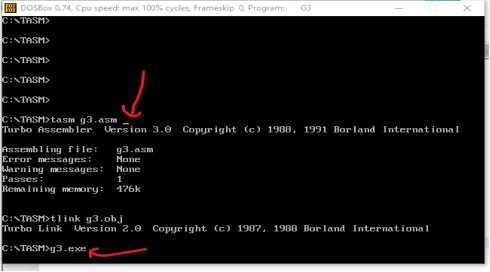
mov dl, 20

mov bh, 0

mov ah, 2

int 10h

end start



* **TITLE "Program to move the cursor on the screen"**

.model small ; this defines the memory model

.stack 100 ; define a stack segment of 100 bytes

.data ; this is the data segment

row db 12 ; define initial row number

col db 39 ; define initial column number

.code

main proc

mov ax,@data ; get the address of the data segment

mov ds, ax ; and store it in register ds

call clearscreen ; clear the screen

call setcursor ; set the cursor position

mov ax, 4c00h ; exit to dos

int 21h

main endp

clearscreen proc

mov ah, 00 ; set video mode

mov al, 03 ; for text 80 x 25

int 10h ; call the dos interrupt

ret ; return to main procedure

clearscreen endp

setcursor proc

mov dh, row ; load row number

mov dl, col ; load column number

mov ah, 2 ; use dos interrupt service for positioning screen

mov bh, 0 ; video page (usually 0)

int 10h ; call the dos interrupt

ret ; return to main procedure

setcursor endp

end main



* **TITLE "Program to enter characters from keyboard"**

.model small ; this defines the memory model

.stack 100 ; define a stack segment of 100 bytes

.data ; this is the data segment

lf equ 10 ; line feed character (0a in hex)

cr equ 13 ; carriage return character (0d in hex)

msg1 db "welcome !!! ", lf, cr, "$"

msg2 db " CS Department, SMIU ", lf, cr, "$"

.code

main proc

mov ax,@data ; get the address of the data segment

mov ds, ax ; and store it in register ds

call clearscreen ; clear the screen

mov dh, 10 ; row 10

mov dl, 13 ; column 13

call setcursor ; set cursor position

lea dx, msg1 ; load the address offset of message to be displayed

mov ah, 09h ; use dos interrupt service for string display

int 21h ; call the dos interrupt

mov dh, 20 ; row 20

mov dl, 13 ; column 13

call setcursor ; set cursor position

lea dx, msg2 ; load the address offset of message to be displayed

mov ah, 09h ; use dos interrupt service for string display

int 21h ; call the dos interrupt

mov ax, 4c00h ; exit to dos

int 21h

main endp

clearscreen proc

mov ah, 00 ; set video mode

mov al, 03 ; for text 80 x 25

int 10h ; call the dos interrupt

ret ; return to main procedure

clearscreen endp

setcursor proc

mov ah, 2 ; use dos interrupt service for positioning screen

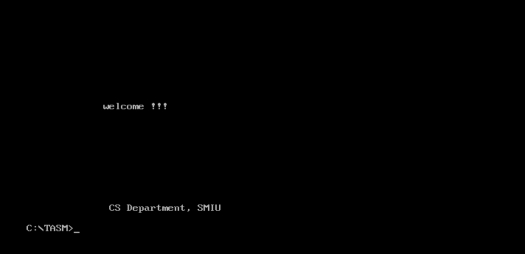
mov bh, 0 ; video page (usually 0)

int 10h ;call the dos interrupt

ret ; return to main procedure

setcursor endp

end main



**Lab 12**

**Push Or Pop**

* **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

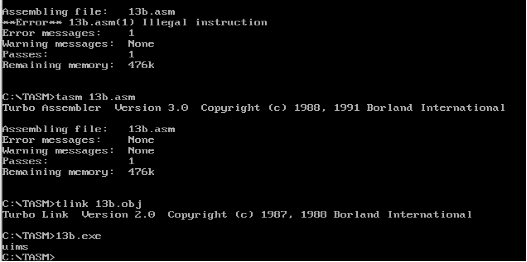
int 21h

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 display

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

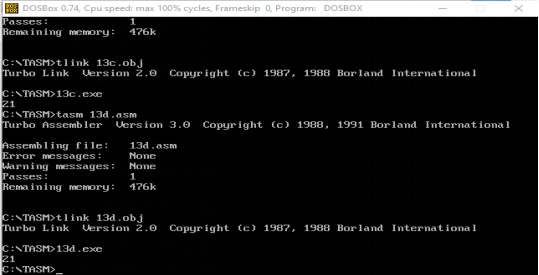
mov dl,bl

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

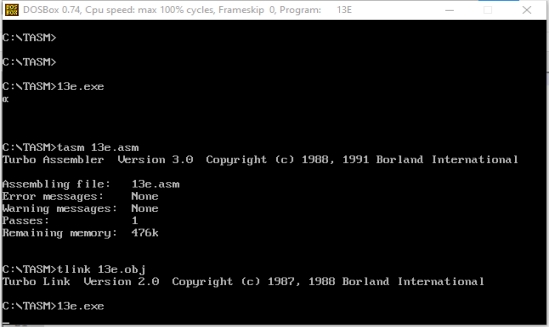
mov dl,al

int 21h

newline

dos\_rtn

end start



**Lab 13**

**One-digit/two-digit addition/subtraction**

* **Program 1: create a program of one-digit addition.**

.model small

.stack 100h

.data

a db ?

b db ?

.code

start:

mov ah, 01

int 21h

sub al,30h

mov a,al

mov ah, 01

int 21h

sub al ,30h

add a,al

add a,48

aaa

Mov ah, 02

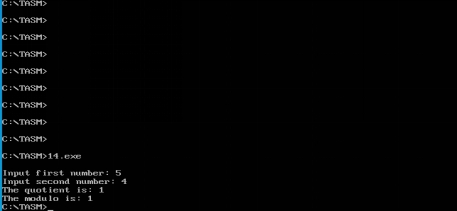
mov dl,a

int 21h

mov ah, 4ch

int 21h

end start



* **Program 2: create a program of two-digit addition.**

.model small

.stack 100h

.data

msg1 db " Enter First number$"

msg2 db 0ah, 0dh," Enter second number$"

msg3 db 0ah,0dh," Result$"

a db 0

b db 0

.code

start:

mov ax, @data

mov ds, ax

mov ah, 09

lea dx, msg1

int 21h

mov ah,01

int 21h

mov a,al

mov ah, 09

lea dx, msg2

int 21h

mov ah,01

int 21h

mov b,al

add al,a

mov ah,0

aaa

add al,48

add ah,48

mov bx,ax

mov ah, 09

lea dx, msg3

int 21h

mov ah,02

mov dl, bh

int 21h

mov ah, 02

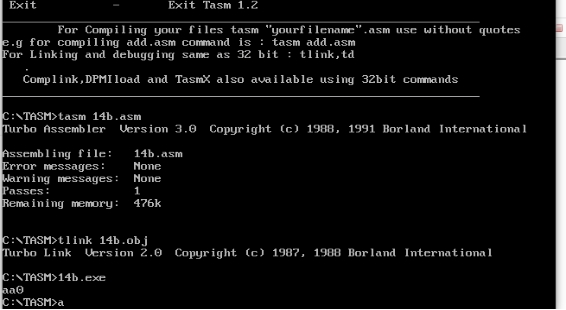
mov dl, bl

int 21h

mov ah, 4ch

int 21h

end start



* **Program 3: create a program of subtraction of one-digit.**

.model small

.stack 100h

.data

a db ?

b db ?

.code

start:

mov ah, 01

int 21h

sub al,30h

mov a,al

mov ah, 01

int 21h

sub al ,30h

sub a,al

add a,48

aas

Mov ah, 02

mov dl,a

int 21h

mov ah, 4ch

int 21h

end start



**Lab 14**

* **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

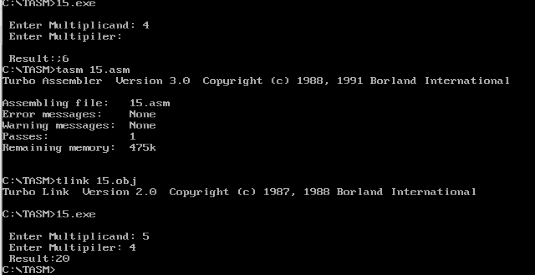
mov dl, bl

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**

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

.model small

.stack 100h

.data

.code

start:

mov al,3

mov bl,1

mul bl

AAM

mov ch,ah

mov cl,al

mov dl,ch

add dl,48

mov ah,2

int 21h

mov dl,cl

add dl,48

mov ah,2

int 21h

mov ah,4ch

int 21h

end start



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

.model small

.stack 100h

.data

q db ?

r db ?

.code

start:

mov ax,@data

mov ds ,ax

mov ax,26

mov bl,5

div bl

mov q,al

mov r,ah

mov dl,q

add dl,48

mov ah,2

int 21h

mov dl,r

add dl,48

mov ah,2

int 21h

mov ah,4ch

int 21h

end start



**LAB 16**

* **Program: 1 Create a file**

.model small

.stack 100h

.data

fname db "thefile.txt",0

address dw ? ;w two bytes leta hai

.code

start:

mov ax,@data

mov ds,ax

mov ah,3ch ;function for create a file

lea dx,fname

mov cl,0 ;0 for create file

int 21h

mov address,ax ;ax for handling file

mov ah,3eh ;function for close a file

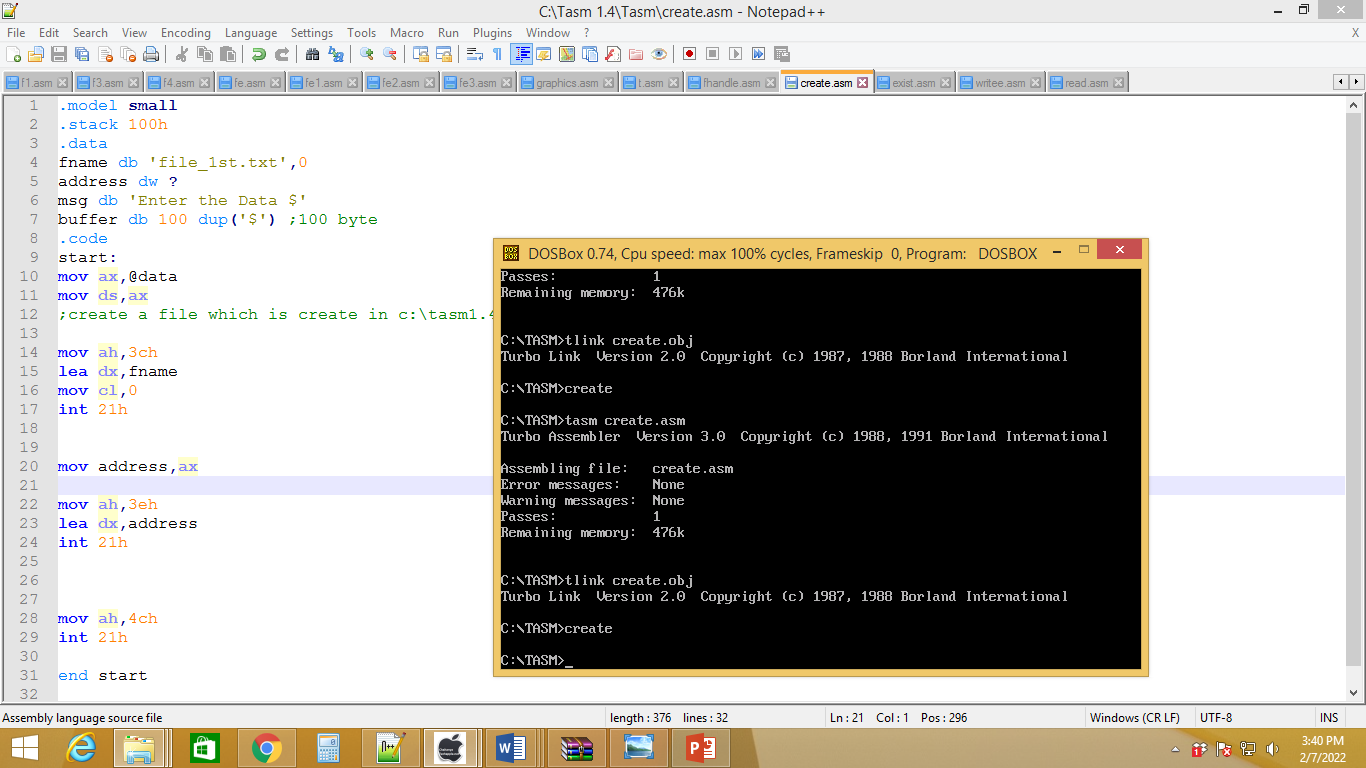
lea dx,address

int 21h

mov ah,4ch

int 21h

end start



* **Program: 2 Write on file**

.model small

.stack 100h

.data

fname db 'yys.txt',0

address dw ?

msg db 'sahar $' ;count how many character in text sahar and write that number in cx line no 25

.code

start:

mov ax,@data

mov ds,ax

;3dh open existing file

mov ah,3dh

lea dx,fname

mov al,2 ;mode 2 for read and write note: be careful for register al or cl it cause error

int 21h

mov address,ax

;write on created file

mov ah ,40h ; 40h will write on file ,which file ans address is a variable which has address of the file on which 40h will write

mov bx,address

mov cx,5

lea dx,msg

int 21h

mov ah,3eh ; 3eh is use to close a file

mov bx,address

int 21h

mov ah,4ch

int 21h

end start

