A picture containing text

Description automatically generated

**Name: Saad Rehman  
Student ID: 21F-9640  
Section: 3A**  
**Degree: BS-CS  
Teacher Name: Sir Abdul Qadeer Bilal  
Course: EL-2003 (COAL)  
Lab Number: 12  
……………………………………………………………………………**

**Qno.1**

Write a code for scroll up behavior of the screen. This will require an input of how many

numbers of lines being needed to be scrolled up as a parameter and then you will use Movs

to scroll up

**Answer**

[org 0x0100]

jmp start

scrollup:

push bp

mov bp,sp

push ax

push cx

push si

push di

push es

push ds

mov ax, 80 ; load chars per row in ax

mul byte [bp+4] ; calculate source position

mov si, ax ; load source position in si

push si ; save position for later use

shl si, 1 ; convert to byte offset

mov cx, 2000 ; number of screen locations

sub cx, ax ; count of words to move

mov ax, 0xb800

mov es, ax ; point es to video base

mov ds, ax ; point ds to video base

xor di, di ; point di to top left column

cld ; set auto increment mode

rep movsw ; scroll up

mov ax, 0x0720 ; space in normal attribute

pop cx ; count of positions to clear

rep stosw ; clear the scrolled space

pop ds

pop es

pop di

pop si

pop cx

pop ax

pop bp

ret 2

start:

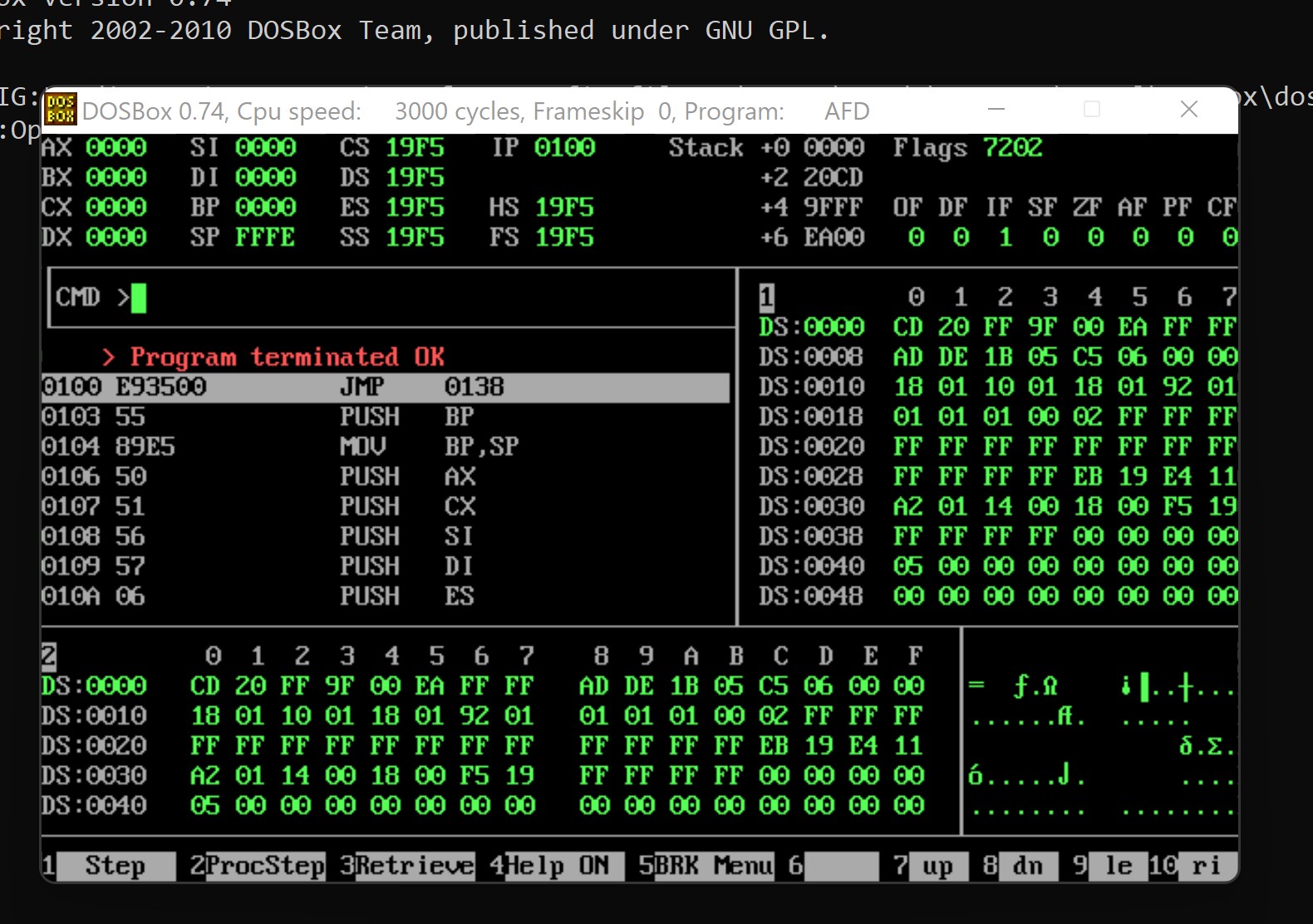
mov ax,5

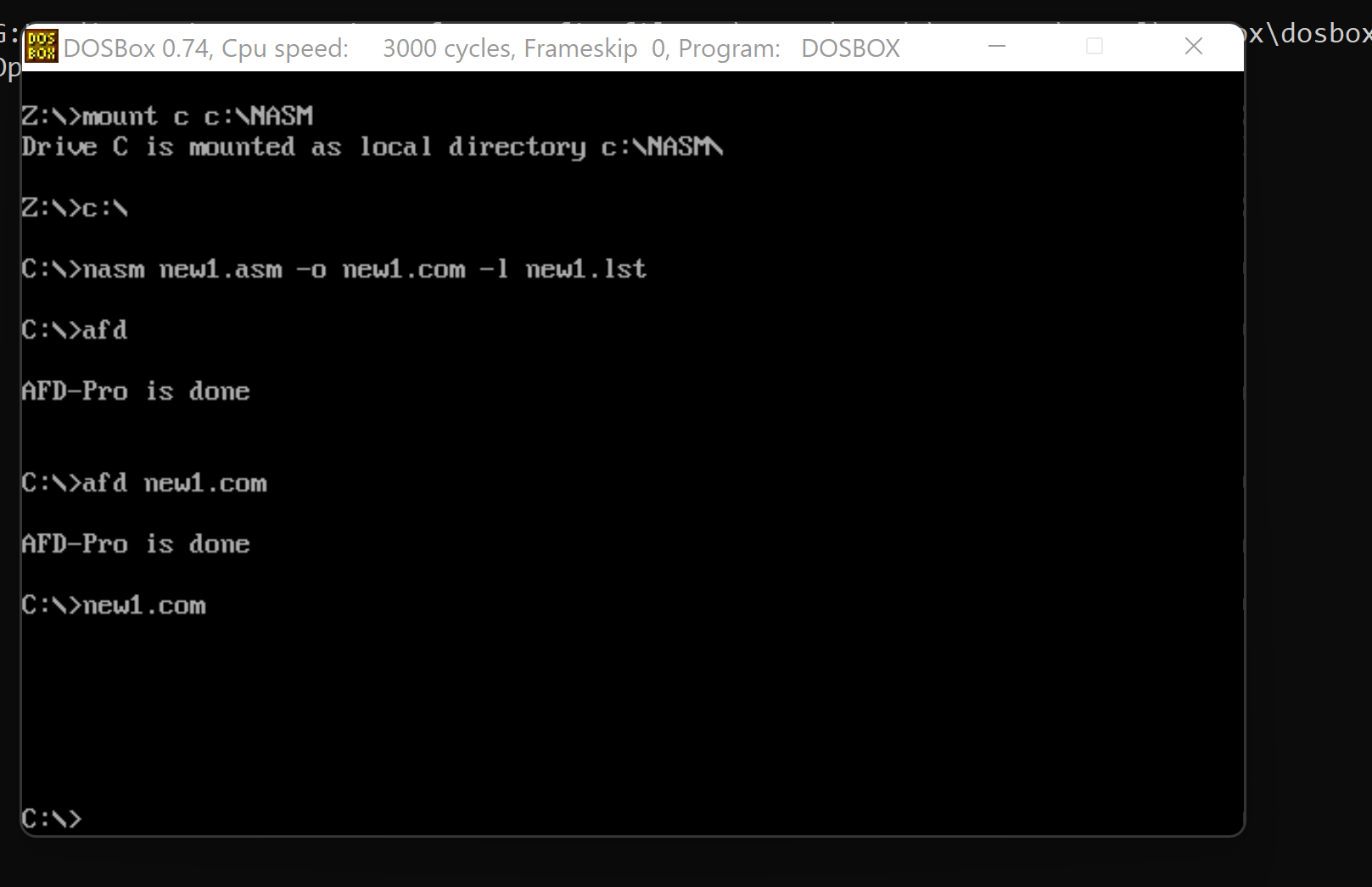
push ax ; push number of lines to scroll

call scrollup ; call the scroll up subroutine

mov ax, 0x4c00 ; terminate program

int 0x21





**Q no. 2:**

Write a code for scroll down behavior of the screen. This will require an input of how many

numbers of lines being needed to be scrolled up as a parameter and then you will use Movs

to scroll down

**Answer**

[org 0x0100]

jmp start

scrolldown:

push bp

mov bp,sp

push ax

push cx

push si

push di

push es

push ds

mov ax, 80 ; load chars per row in ax

mul byte [bp+4] ; calculate source position

push ax ; save position for later use

shl ax, 1 ; convert to byte offset

mov si, 3998 ; last location on the screen

sub si, ax ; load source position in si

mov cx, 2000 ; number of screen locations

sub cx, ax ; count of words to move

mov ax, 0xb800

mov es, ax ; point es to video base

mov ds, ax ; point ds to video base

mov di, 3998 ; point di to lower right column

std ; set auto decrement mode

rep movsw ; scroll up

mov ax, 0x0720 ; space in normal attribute

pop cx ; count of positions to clear

rep stosw ; clear the scrolled space

pop ds

pop es

pop di

pop si

pop cx

pop ax

pop bp

ret 2

start:

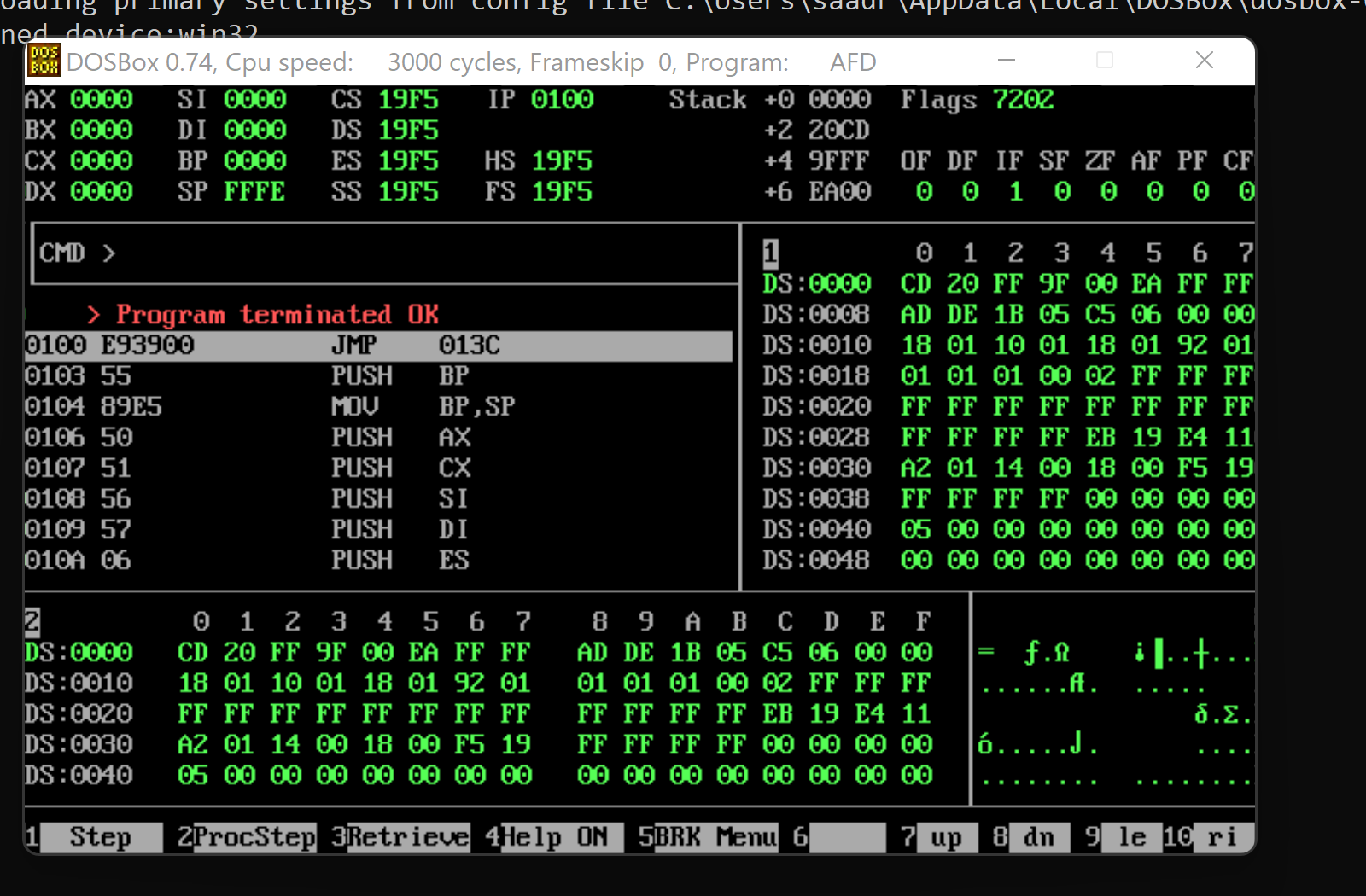
mov ax,5

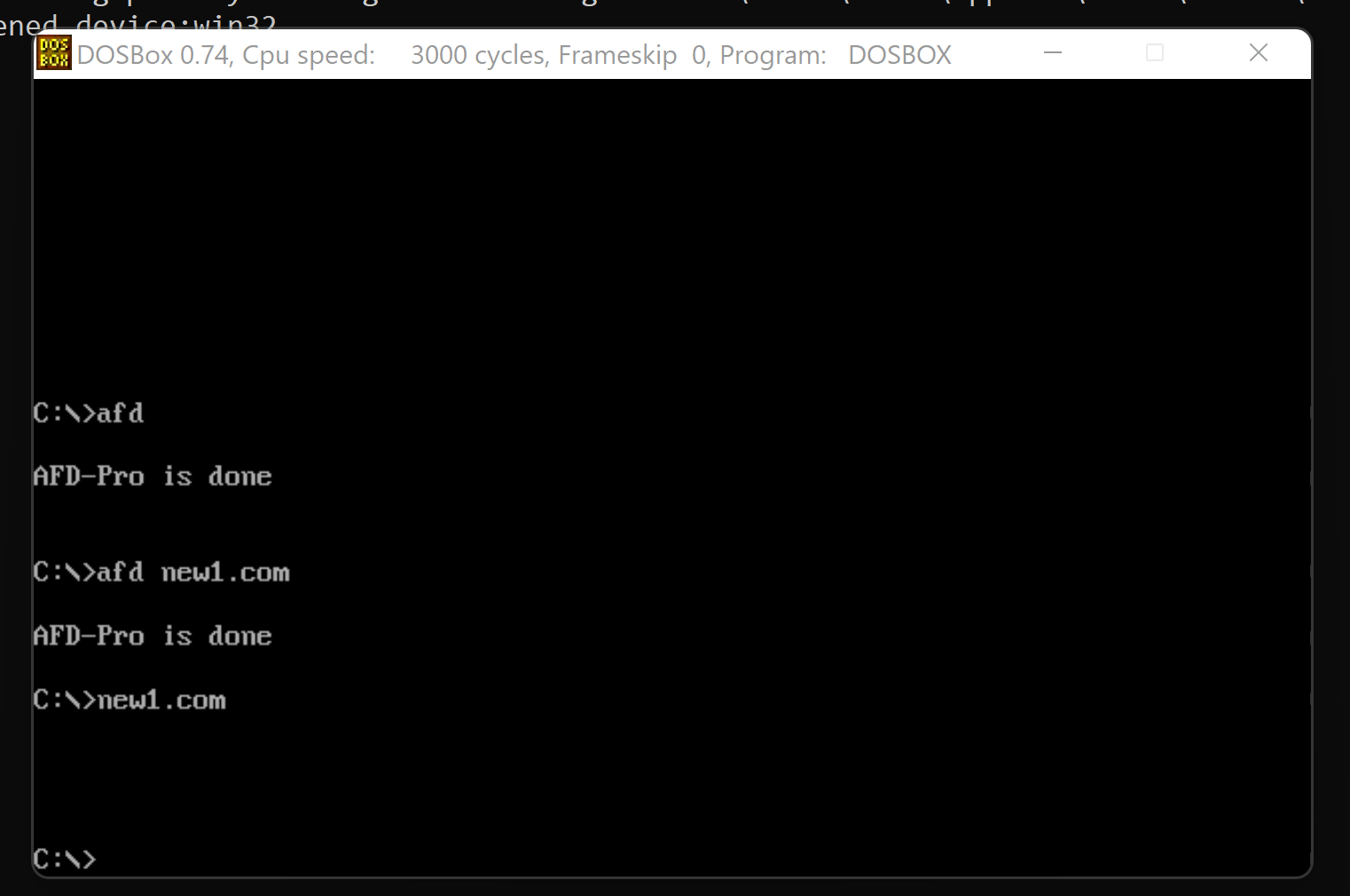
push ax ; push number of lines to scroll

call scrolldown ; call scroll down subroutine

mov ax, 0x4c00 ; terminate program

int 0x21





**Q no. 3(i & ii):**

String1 “Coal Lab”

String2 “Coal Lab”

String3 “PF Lab”

i. Find lengths of all three strings.

ii. Find which of the following strings are equal

iii. Find “Coal” in stirng1, “Lab” in string2 and “PF” in string3

You will CMPS statements for these tasks.

**Answer**

[org 0x0100]

jmp start

msg1: db 'Coal Lab', 0

msg2: db 'Coal Lab', 0

msg3: db 'PF Lab', 0

; subroutine to calculate the length of a string

; takes the segment and offset of a string as parameters

strlen: push bp

mov bp,sp

push es

push cx

push di

les di, [bp+4] ; point es:di to string

mov cx, 0xffff ; load maximum number in cx

xor al, al ; load a zero in al

repne scasb ; find zero in the string

mov ax, 0xffff ; load maximum number in ax

sub ax, cx ; find change in cx

dec ax ; exclude null from length

pop di

pop cx

pop es

pop bp

ret 4

; subroutine to print a string

; subroutine to compare two strings

; takes segment and offset pairs of two strings to compare

; returns 1 in ax if they match and 0 other wise

strcmp: push bp

mov bp,sp

push cx

push si

push di

push es

push ds

lds si, [bp+4] ; point ds:si to first string

les di, [bp+8] ; point es:di to second string

push ds ; push segment of first string

push si ; push offset of first string

call strlen ; calculate string length

mov cx, ax ; save length in cx

push es ; push segment of second string

push di ; push offset of second string

call strlen ; calculate string lenth

cmp cx, ax ; compare length of both strings

jne exitfalse ; return 0 if they are unequal

mov ax, 1 ; store 1 in ax to be returned

repe cmpsb ; compare both strings

jcxz exitsimple ; are they successfully compared

exitfalse: mov ax, 0 ; store 0 to mark unequal

exitsimple: pop ds

pop es

pop di

pop si

pop cx

pop bp

ret 8

start: push ds ; push segment of first string

mov ax, msg1

push ax ; push offset of first string

push ds ; push segment of second string

mov ax, msg2

push ax ; push offset of second string

call strcmp ; call strcmp subroutine

push ds ; push segment of first string

mov ax, msg1

push ax ; push offset of first string

push ds ; push segment of third string

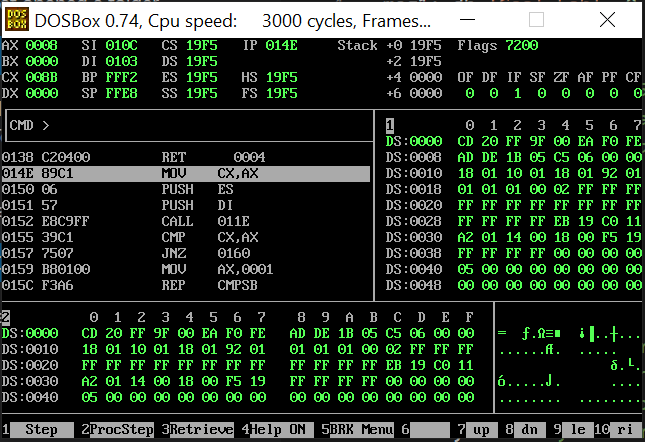
mov ax, msg3

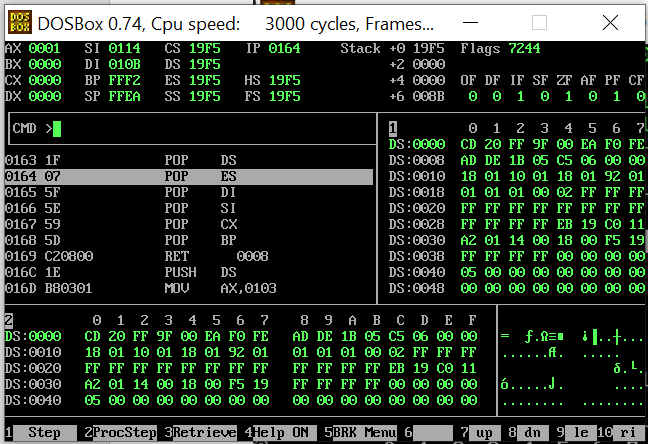
push ax ; push offset of third string

call strcmp ; call strcmp subroutine

mov ax, 0x4c00 ; terminate program

int 0x21





**Q no. 3(iii):**

[org 0x0100]

jmp start

msg1: db 'coal lab', 0

msg2: db 'coal lab', 0

msg3: db 'pf   lab', 0

ms1: db 'coal',0

ms2: db 'lab',0

ms3: db 'pf',0

strlen: push bp

mov bp,sp

push es

push cx

push di

les di, [bp+4]

mov cx, 0xffff

xor al, al

repne scasb

mov ax, 0xffff

sub ax, cx

dec ax

pop di

pop cx

pop es

pop bp

ret 4

strcmp: push bp

mov bp,sp

push cx

push si

push di

push es

push ds

lds si, [bp+4]

les di, [bp+8]

push ds

push si

call strlen

mov cx, ax

push es

push di

call strlen

cmp cx, ax

jne exitfalse

mov ax, 1

repe cmpsb

jcxz exitsimple

exitfalse: mov ax, 0

exitsimple: pop ds

pop es

pop di

pop si

pop cx

pop bp

ret 8

start:

push ds

mov ax,msg1

push ax

call strlen

push ds

mov ax,msg2

push ax

call strlen

push ds

mov ax,msg3

push ax

call strlen

push ds

mov ax,msg1

push ax

push ds

mov ax,msg2

push ax

call cmpstr

push ds

mov ax,msg1

push ax

push ds

mov ax,msg3

push ax

call cmpstr

push ds

mov ax,msg2

push ax

push ds

mov ax,msg3

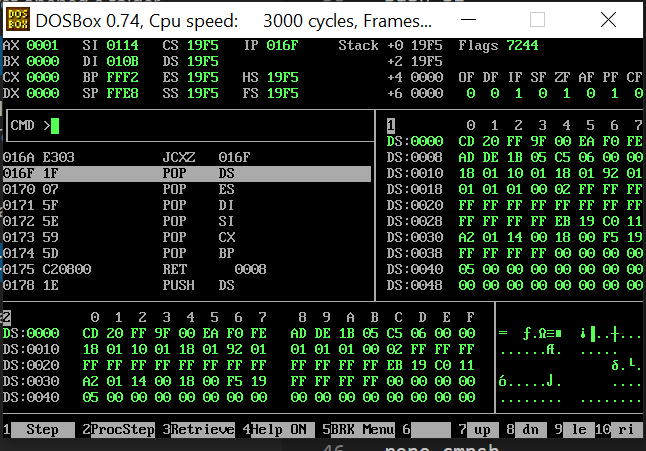
push ax

call cmpstr

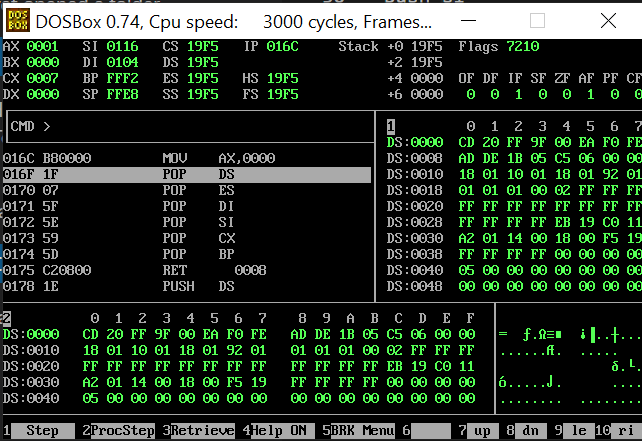
mov ax, 0x4c00

int 0x21

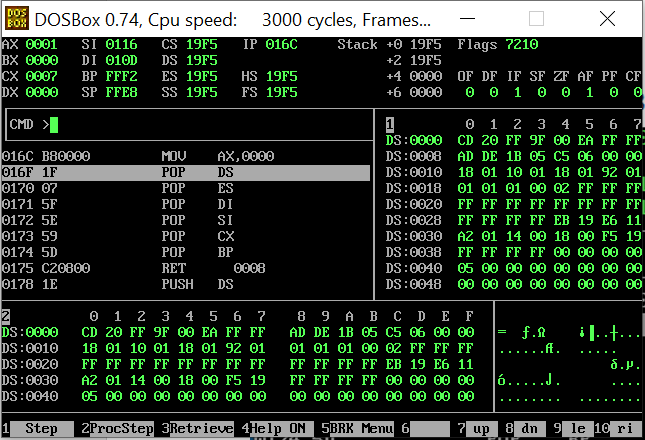
**String 1 found:**



**String 2 found:**



**String 3 found:**



**The End**