**National University of Computer and Emerging Sciences**

**Lab Manual**

**Computer Organization and Assembly Language**

****

**Lab 09**

**Instructor** **:** Hazoor Ahmad

**Class**  **:** CS3

**Semester**  **:** Fall 2022

**Name** **:**

**Roll no.** **:**

Fast School of Computing

FAST-NU, Lahore, Pakistan

Objectives

• Subroutines

• Display Memory

• String Instructions

**Contents**

Objectives 2 ACTIVITY 1: [20 Marks] 2 ACTIVITY 2: [20 Marks] 2 ACTIVITY 3: [40+20 Marks] 3 REFERENCES 3

**Note for all questions**: You can make as many memory variables, subroutines as you need. Must read all the manual before starting.

**ACTIVITY 1: [20 Marks]** Write a subroutine which copies contents of string1 into another string but without spaces and punctuation marks. For example, if it is provided the following string

*String1 DB “Mr. Ali, Usman, & Anwar! Doing what???? want to travel????”, ‘0’* It should return

*String2 DB “MrAliUsmanAnwarDoingwhatwanttotravel”, ‘0’*

Note: Your subroutine should be capable of eliminating spaces and punctuation marks from the string of any size.

**SOLUTION:**

[org 0x0100]

jmp start

msg: db 'Mr. Ali, Usman, & Anwar! Doing what???? want to travel????',0

msg2:db ' ', 0

remove:

mov dx,cx

mov si,0

mov di,0

l1:

mov bx,0

mov al,[msg+si]

cmp al,'0'

jb no

cmp al,'9'

jbe inp

cmp al,'A'

jb no

cmp al,'Z'

jbe inp

cmp al,'a'

jb no

cmp al,'z'

jbe inp

ja no

inp:

mov [msg2+di],ax

add di,1

no:

add si,1

dec cx

cmp cx,0

jne l1

len:

push ds

pop es

mov di,msg

mov ax,0

mov cx,0xffff

repne scasb

mov ax,0xffff

sub ax,cx

dec ax

ret

prnt:

mov ax,0xb800

mov es,ax

mov ah,0x30

mov si,msg2

mov cx,cx

dec cx

mov di,500

l2:

lodsb

stosw

loop l2

ret

start:

call len

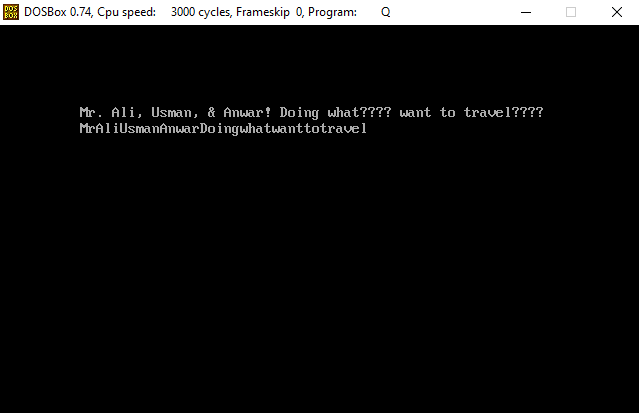
mov cx,ax

call remove

call prnt

mov ax,0x4c00

int 0x21

**SCREENSHOT: **

**ACTIVITY 2: [20 Marks]** Write a subroutine reverses the contents of a given string. For example, if it is provided the following string

*String1 DB “I am Mr X”, ‘0’*

It should return

**SOLUTION:**

[org 0x0100]

jmp start

msg: db 'I am Mr X',0

msg2:db ' ',0

remove:

mov dx,cx

mov si,0

mov di,0

l1:

mov bx,0

mov al,[msg+si]

cmp al,'0'

jb no

cmp al,'9'

jbe inp

cmp al,'A'

jb no

cmp al,'Z'

jbe inp

cmp al,'a'

jb no

cmp al,'z'

jbe inp

ja no

inp:

mov [msg2+di],ax

add di,1

no:

add si,1

dec cx

cmp cx,0

jne l1

ret

reverse:

mov di,0

mov cx,cx

mov si,cx

dec si

l3:

mov bx,[msg+si]

dec si

mov [msg2+di],bx

inc di

cmp cx,0

jne l3

ret

len:

push ds

pop es

mov di,msg

mov ax,0

mov cx,0xffff

repne scasb

mov ax,0xffff

sub ax,cx

dec ax

ret

prnt:

mov ax,0xb800

mov es,ax

mov ah,0x30

mov si,msg2

mov cx,cx

dec cx

mov di,500

l2:

lodsb

stosw

loop l2

ret

start:

call len

mov cx,ax

;call remove

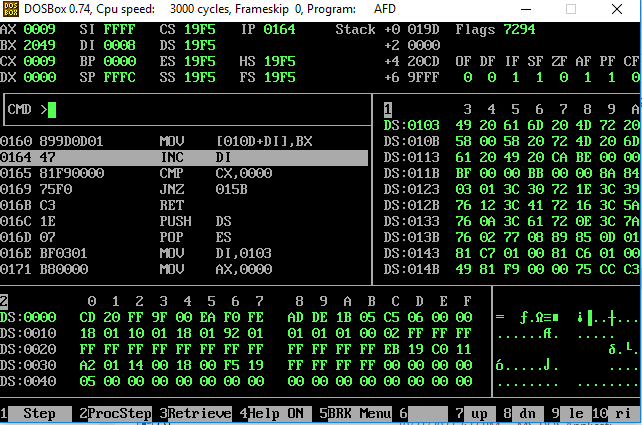
;call prnt

call reverse

mov ax,0x4c00

int 0x21

**SCREENSHOT:**



*String2 DB “X rM ma I”, ‘0’*

Note: Your subroutine should be capable to reverse the string of any size.

**ACTIVITY 3: [20 Marks]** Write a program which

1. First removes punctuation from **String1** and produces **String2**

2. The reverses the **String2** and produces **String3**

3. Compare both Strings (**String2** and **String3)** for equality

4. If both strings are equal, print(“The given string is palindrome”) otherwise print(“The given string is not a palindrome”)

*String1 DB “*A man, a plan, a canal, Panama!!!*”, ‘0’*

*String2 DB “*AmanaplanacanalPanama*”, ‘0’*

*String3 DB “*amanaPlanacanalPanamA*”, ‘0’*

**SOLUTION:**

[org 0x0100]

jmp start

msg: db 'aaabbaaa',0

msg2:db ' ',0

msg3:db 'palindrome'

remove:

mov dx,cx

mov si,0

mov di,0

l1:

mov bx,0

mov al,[msg+si]

cmp al,'0'

jb no

cmp al,'9'

jbe inp

cmp al,'A'

jb no

cmp al,'Z'

jbe inp

cmp al,'a'

jb no

cmp al,'z'

jbe inp

ja no

inp:

mov [msg2+di],ax

add di,1

no:

add si,1

dec cx

cmp cx,0

jne l1

ret

reverse:

mov di,0

mov cx,cx

mov si,cx

dec si

l3:

mov bx,[msg+si]

dec si

mov [msg2+di],bx

inc di

cmp cx,0

jne l3

ret

len:

push ds

pop es

mov di,msg

mov ax,0

mov cx,0xffff

repne scasb

mov ax,0xffff

sub ax,cx

dec ax

ret

prnt:

mov ax,0xb800

mov es,ax

mov ah,0x30

mov si,msg3

mov cx,10

dec cx

mov di,500

l2:

lodsb

stosw

loop l2

ret

compare:

mov si,msg

mov di,msg2

mov cx,cx

repe cmpsb

call true

ret

true:

mov ax,0xb800

mov es,ax

mov ah,0x30

mov si,msg3

mov cx,10

mov di,500

l4:

lodsb

stosw

loop l4

ret

start:

call len

mov cx,ax

;call remove

call prnt

call reverse

call compare

mov ax,0x4c00

int 0x21

**SCREENSHOT:**



**ACTIVITY 4: [20 Marks]** Write a program which prints a moving counter as shown in the attached video file[1].

**SOLUTION:**

[org 0x100]

jmp start

print: push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov ax,0xb800

mov es,ax

mov di,[bp+4]

mov ah,0x04

mov al,0x1

h1: mov [es:di],ax

add al,1

add di,2

call delay

call clear

add si,2

cmp si,4000

jnz h1

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret4

clear: push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov ax,0xb800

mov es,ax

mov di,0

mov cx,2000

mov ax,0x0720

cld

rep stosw

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret

delay:

push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov cx,0xffff

g1: loop g1

mov cx,0xffff

g2: loop g2

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret

start: call clear

mov cx, 0xffff

push cx

mov ax,0

push ax

call print

mov ax, 0x4c00

int 0x21

**SCREENSHOT:**

**ACTIVITY 5: [20 Marks]** Write a subroutine **RANDOMPOS** which (on each call) receives a number as **SEED** and based on that **SEED** generates random position (**X**, **Y** of the DOSBox Screen).

Write a program which displays Counter from **Activity4** on random locations Using the **RANDOMPOS**.

**SOLUTION:**

[org 0x100]

jmp start

print: push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov ax,0xb800

mov es,ax

mov di,[bp+4]

mov ah,0x04

mov al,0x1

h1: mov [es:di],ax

add al,1

add di,2

call delay

call clear

add si,2

cmp si,4000

jnz h1

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret4

clear: push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov ax,0xb800

mov es,ax

mov di,0

mov cx,2000

mov ax,0x0720

cld

rep stosw

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret

delay:

push bp

mov bp,sp

push es

push si

push di

push ax

push cx

mov cx,0xffff

g1: loop g1

mov cx,0xffff

g2: loop g2

pop cx

pop ax

pop di

pop si

pop es

pop bp

ret

start: call clear

mov cx, 0xffff

push cx

mov ax,0

push ax

call print

mov ax, 0x4c00

int 0x21

**SCREENSHOT:**