Matrix Operations

Midterm

Computer Organization and Assembly Language

Student Name	Nouman Javed
Registration #	21L-1788
Instructor	Hazoor Ahmad
Class	CS3
Section	A, D, H, H
Semester	Fall 2022

Fast School of Computing

FAST-NU, Lahore, Pakistan

Activity 1

Assembly Language Code

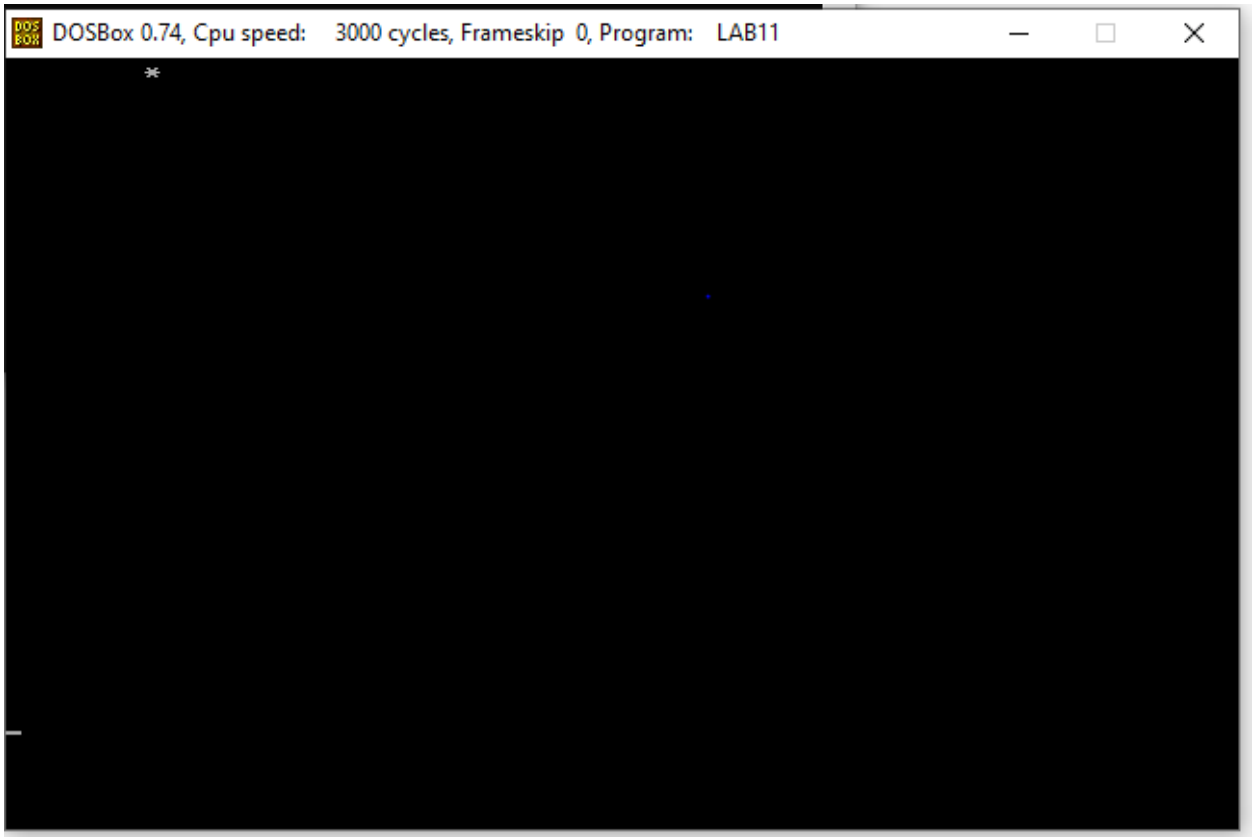
```
[org 0x0100]
jmp start

tickcount:dw 0
left:dw 0
right:dw 1
up:dw 0
down:dw 0
start:
call clrsr

mov ax,0
push ax
pop es

mov di,0

cli

mov word[es:8*4],timmer
mov word[es:8*4+2],cs

sti


l1:

jmp l1


mov ax,0x4c00
int 21h


clrsr:


push di
push cx
push ax
push bx
push ax
push dx
```

```
mov bx,0xb800
mov es,bx
mov di,0

mov cx,2000
mov ax,0x0720
rep stosw


pop dx
pop ax
pop bx
pop ax
pop cx
pop di

ret


timmer:

push ax
push es
mov ax,0xb800
push ax
pop es


add word[tickcount],1

cmp word[tickcount],20
jne d5


mov word[up],0
mov word[down],0
mov word[left],0

cmp word[right],1
jne d1
mov word[es:di],0x0720
cmp di,158
je d1
add di,2

mov word[es:di],0x072a ;asterik printing
```

```
mov word[tickcount],0
jmp exit1

d1:

mov word[right],0
mov word[up],0
mov word[left],0


mov word[down],1

cmp word[down],1
jne d2

mov word[es:di],0x0720
cmp di,3998
je d2
add di,160

mov word[es:di],0x072a
mov word[tickcount],0

jmp exit1

d5:

jmp exit1

d2:

mov word[up],0
mov word[down],0
mov word[right],0

mov word[left],1

cmp word[left],1
jne d3

mov word[es:di],0x0720
cmp di,3840
je d3
sub di,2

mov word[es:di],0x072a
mov word[tickcount],0
```

```
jmp exit1


d3:
mov word[right],0
mov word[down],0
mov word[right],0
mov word[up],1


cmp word[up],1
jne exit1

mov word[es:di],0x0720
cmp di,0
je exit1
sub di,160

mov word[es:di],0x072a
mov word[tickcount],0
jmp exit1




exit1:

mov al,0x20

out 0x20,al

pop es

pop ax

iret
```

Debugging Screenshots



Activity 2

Assembly Language Code

```
[org 0x0100]

   jmp main
; old isr offset and segment
oldisr: dd 0
 ; buffer to save video memory
temp: times 2000 dw 0
```

```
jmp main



; to store video memory in buffer

store_buffer:
push    bp
mov     bp, sp
push    ax
push    cx
push    si
push    di
push    es
push    ds

mov     ax, 0xb800  ; points to video memory
mov     ds, ax
mov     si, 0
mov     ax, cs
mov     es, ax
mov     di, temp
mov     cx, 2000

cld
rep     movsw    ; move data from video memory to buffer

pop     ds
pop     es
pop     di
pop     si
pop     cx
pop     ax
pop     bp
ret

print:
push ax
push bx
push cx
push dx


mov cx,ax
```

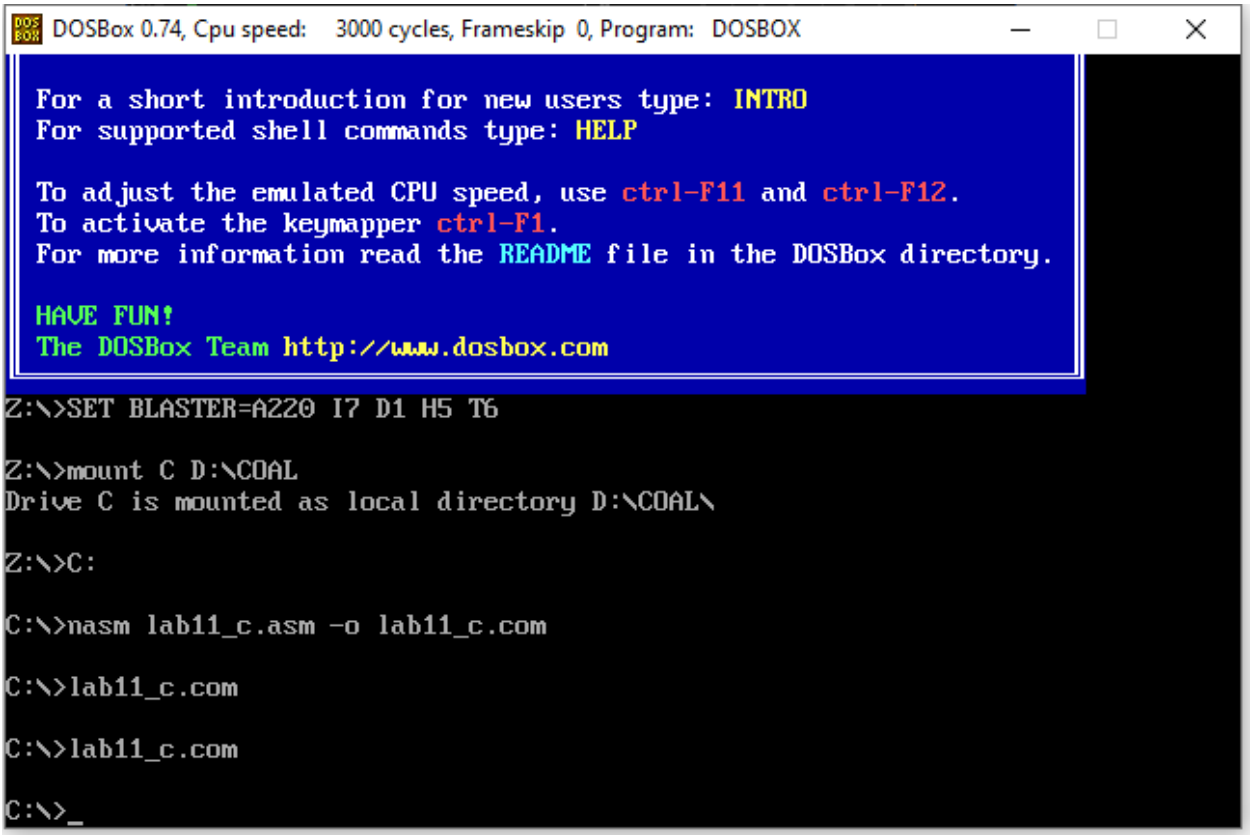
```
mov ax,0xb800
push ax
pop es

mov ah,0x0f

next1:
stosw
lodsb

loop next1


pop dx
pop cx
pop bx
pop ax
ret


; load buffer
load_buffer:
push    bp
mov     bp, sp
push    ax
push    cx
push    si
push    di
push    es
push    ds

mov     ax, 0xb800      ; points to video memory
mov     es, ax
mov     di, 0
; points to temporary memory
mov     ax, cs
mov     ds, ax
mov     si,  temp
mov     cx, 2000

cld
rep     movsw       ; load buffer in video memory

pop     ds
pop     es
pop     di
pop     si
```

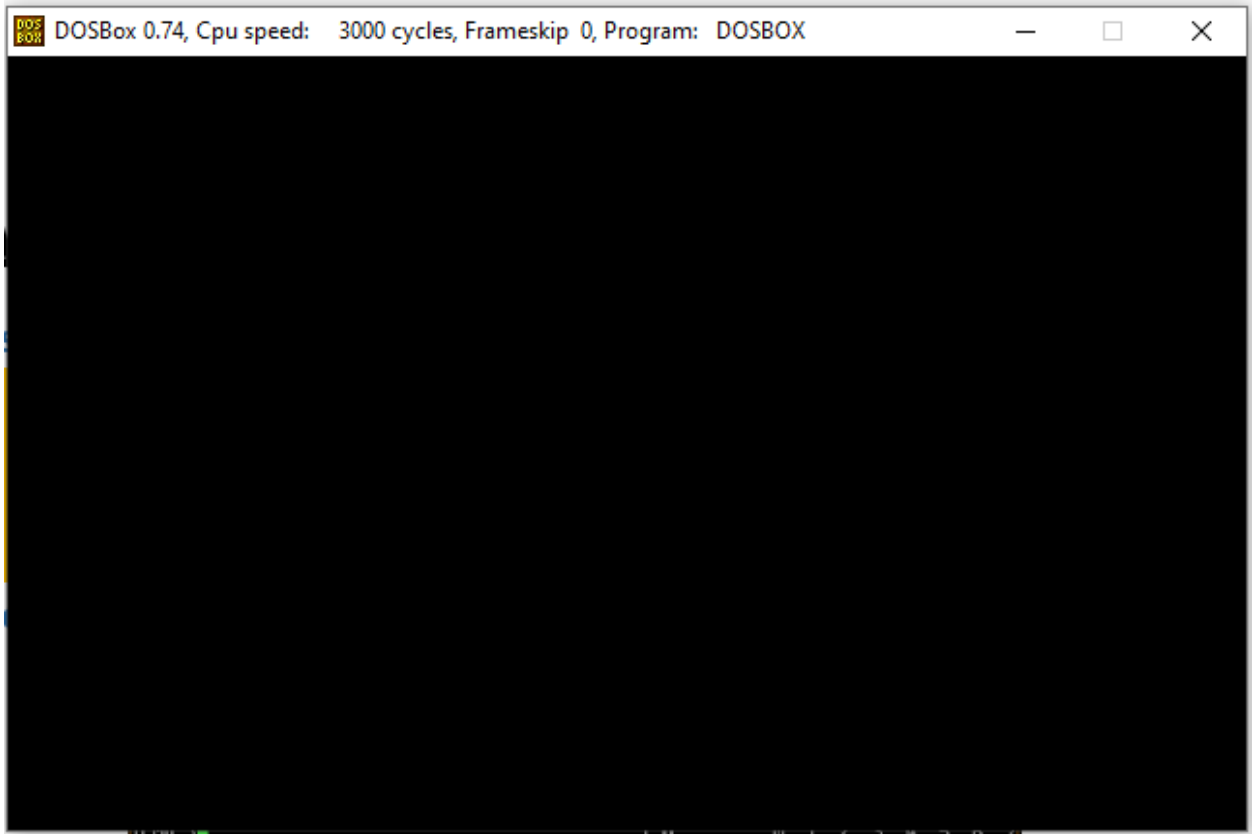
```
pop     cx
pop     ax
pop     bp
ret

movement:
push ds
push es
pop ds


rep movsw

pop ds

ret


; hook key board interrupt with interrupt chaining
kbISR:
push    ax
in      al, 0x60     ; read a char from keyboard

cmp     al, 00011101b  ;code for ctrl comparison
JNE     nextCmp

CALL    store_buffer   ; store video memory in a buffer
CALL    clrscr         ; clear screen
jmp     exit

nextCmp:
cmp     al, 10011101b
JNE     noMatch

CALL    delay          ; add some delay
CALL    load_buffer    ; load buffer in video memory
jmp     exit

noMatch:
pop     ax
jmp     far [cs:oldisr] ; CALL the original ISR

exit:
; send EOI
mov     al, 0x20
out     0x20, al
pop     ax
```

```
iret


;Write a TSR to clear the screen when CTRL key is pressed and restore it when it is released
main:
xor     ax, ax
mov     es, ax
; save old keyboard isr
mov     ax, [es:9*4]
mov     [oldisr], ax
mov     ax, [es:9*4+2]
mov     [oldisr+2], ax

; hook keyboard interrupt
cli
mov     word [es:9*4], kbISR
mov     [es:9*4+2], cs
sti


l1:

jmp l1

mov ax,0x4c00
int 21h



; to clear video screen
clrscr:
push    es
push    ax
push    di

mov     ax, 0xb800
mov     es, ax
mov     di, 0
nextchar:
mov     word [es:di], 0x0720
add     di, 2
cmp     di, 4000
jne     nextchar

pop     di
pop     ax
pop     es
```

```
ret

; to add some delay

delay:
push ax
push bx
push cx
push dx
mov ax,10
lo2:
mov cx,0xff
cmp ax,0
jz exit1
sub ax,1
lo1:


cmp cx,0
jz lo2

sub cx,1

jmp lo1


exit1:

pop dx
pop cx
pop bx
pop ax

ret
```

Debugging Screenshots





Activity 3

Assembly Language Code

```
[org 0x0100]

        jmp start


ms:             dw 0                                    ;Milli seconds

count:  dw 0, 0, 0, 0, 0                ;Count of the characters typed

tCount: dw -1                   ;Note: The tCount of first second is initialized to -1 for one time because when
you type the command and press ENTER
                                        ;       then the program gets loaded. And it takes you a
few milliseconds to release the ENTER key
                                        ;       and since the program was loaded before, it counts
this release of ENTER key as one. So this release count
                                        ;   is ignored by initializing the count to -1

iNo : dw 0
```

```
location: db 0                          ;Location where the next star is to be printed


string1:db 0
string2:db 0
string3:db 0
string4:db 0
string5:db 0
string6:db 0
string7:db 0
string8:db 0



jmp start

;Clear Screen
clrscr:
                                        push es

                                        mov ax, 0xb800
                                        mov es, ax
                                        xor di,di
                                        mov ax,0x0720
                                        mov cx,2000

                                        cld
                                        rep stosw

                                        pop es

                                        ret


;Program to print the stars
printStars:                     pusha
                                        push es

                                        mov ax, 0xb800
                                        mov es, ax

                                        mov al, 80
                                        mul byte [cs:location]
                                        add ax, 159
                                        shl ax, 1
```

```
                    mov di, ax

                    mov cx, [cs:tCount]

                    cmp cx, 0
                    jle return


l1:                 mov byte [es:di], '*'
                    inc byte [cs:location]
                    add di, 160
                    loop l1

return:             pop es
                    popa
                    ret
Jmpzuser:
call clrsr
call delay

mov di,string1
call length
mov di,1960
mov si,string1
call print

call delay

mov si,1920
mov di,1760
mov cx,160
call movement


call delay
call delay
mov di,string2
call length
mov di,1960
mov si,string2
call print

call delay

mov si,1760
mov di,1600
mov cx,320
```

```
call movement


call delay
call delay
mov di,string3
call length
mov di,1960
mov si,string3
call print

call delay

mov si,1600
mov di,1440
mov cx,480
call movement

call delay
call delay
mov di,string4
call length
mov di,1960
mov si,string4
call print

call delay

mov si,1440
mov di,1280
mov cx,640
call movement


call delay
call delay
mov di,string5
call length
mov di,1960
mov si,string5
call print

call delay

mov si,1280
mov di,1120
mov cx,800
```

```
call movement


call delay
call delay
mov di,string6
call length
mov di,1960
mov si,string6
call print

call delay

mov si,1120
mov di,960
mov cx,960
call movement


call delay
call delay
mov di,string7
call length
mov di,1960
mov si,string7
call print

call delay

mov si,960
mov di,800
mov cx,1120
call movement


call delay
call delay
mov di,string8
call length
mov di,1960
mov si,string8
call print

call delay

mov si,800
mov di,640
```

```
mov cx,1120
call movement




CTS:                    pusha

                        ;These lines will execute for the very first five seconds
                        cmp word [cs:iNo], 10
                        jz l2

                        add word [cs:ms], 55
                        cmp word [cs:ms], 1000
                        jl EOI2

                        mov word [cs:ms], 0                 ;Resetting the MilliSeconds to zero
                        call printStars                     ;Because the count is to be updated
every second i.e
                                                                                                ;the
stars are to be printed after every second

                        mov ax, [cs:tCount]
                        mov bx, [cs:iNo]

                        mov word [cs:count + bx], ax

                        mov word [cs:tCount], 0
                        add word [cs:iNo], 2

                        jmp EOI2



l2:                     add word [cs:ms], 55
                        cmp word [cs:ms], 1000
                        jl EOI2

                        mov word [cs:ms], 0                 ;Resetting the MilliSeconds to zero

                        ;Shifting the counts towards the right, to create a space for this current second

                        mov dx, 0

                        mov ax, [cs:count + 2]
                        add dx, ax
```

```
                    mov [cs:count], ax


                    mov ax, [cs:count + 4]
                    add dx, ax
                    mov [cs:count + 2], ax


                    mov ax, [cs:count + 6]
                    add dx, ax
                    mov [cs:count + 4], ax

                    mov ax, [cs:count + 8]
                    add dx, ax
                    mov [cs:count + 6], ax

                    mov ax, [cs:tCount]
                    add dx, ax
                    mov [cs:count + 8], ax

                    jmp a1


EOI2:               jmp EOI                                         ;Intermediate
Jump


                    ;Now dx contains the count of the last five seconds

a1:                 mov [cs:tCount], dx

                    call clrscr

                    mov byte [cs:location], 0
                    call printStars

                    mov word [cs:tCount], 0


EOI:                mov al, 0x20
                    out 0x20, al



exit:               popa
                    iret
```

```
delay:
push ax
push bx
push cx
push dx
mov ax,10
lo2:
mov cx,0xffff
cmp ax,0
jz exit1
sub ax,1
lo1:


cmp cx,0
jz lo2

sub cx,1

jmp lo1


exit1:

pop dx
pop cx
pop bx
pop ax

ret


length:
push bx
push cx
push dx

push ds  ;data segment mov in es
pop es

mov cx,0xffff
mov al,0

repne scasb
```

```
mov ax,0xffff
sub ax,cx    ;ax has string length



pop dx
pop cx
pop bx

ret


print:
push ax
push bx
push cx
push dx


mov cx,ax
mov ax,0xb800
push ax
pop es

mov ah,0x0f

next1:
stosw
lodsb

loop next1


pop dx
pop cx
pop bx
pop ax
ret



movement:
push ds
push es
pop ds
```

```
rep movsw

pop ds

ret


clrsr:


push di
push cx
push ax
push bx
push ax
push dx

mov bx,0xb800
mov es,bx
mov di,0

mov cx,2000
mov ax,0x0720
rep stosw


pop dx
pop ax
pop bx
pop ax
pop cx
pop di

ret



;Keyboard ISR

kbisr:      push ax

                        in al, 0x60

                        shl al, 1
                        jnc EOI1

                        inc word [cs:tCount]                    ;If a key is released, only then increase the count
```

```
EOI1:     mov al, 0x20
                    out 0x20, al

                    pop ax

                    iret





start:                    mov ax, 0
                              mov es, ax

                              mov bx, 0

                              call clrscr

                              ;Hooking the interrupts
                              cli

                              mov word [es: 9*4], kbisr
                              mov [es:9*4+2], cs

                              mov word [es:8*4], CTS
                              mov [es:8*4+2], cs

                              sti


                              ;Code for making it TSR
                              mov dx, start                                   ;End of resident portion
                              add dx, 15                                            ;round up to next
para
                              mov cl, 4
                              shr dx, cl                                            ;number of paras

end:                      mov ax, 0x3100                            ;terminate and stay resident
                              int 21h
```

Debugging Screenshots