0032

National University of Computer and Emerging Sciences
Lahore Campus

COAL (EE2003)

Computer organization and assembly language

Date: June 03, 2025

Course Instructor(s)

Sana Fatima, Samin Iftikhar

Final Exam

Total Time (Hrs):	3
Total Marks:	90
Total Questions:	3

16 | 23
11 - 8

Roll No ____________ Section ____________ Student Signature ____________

Instructions: It is an open book exam. Only textbook *"Assembly Language Programming Lecture Notes by Bilal Hashmi"* is allowed. ANY NOTES OR ATTACHMENTS ARE **NOT ALLOWED**.

ATTEMPT Q1 AND 2 ON THIS BOOKLET AND Q3 ON ANSWERSHEET.

CLO # 1: Demonstrate the basic concepts of computer organization including CPU, memories, and input/output and explain their purposes and interactions.

Q1: [30 marks] Short Questions.

I. [10 Marks] A program is trying to write soma data on address [AF00:994A] which location in terms of (row, col) is this in Display Memory? Row = (14)10 Col = (69)10

Show your working to get credit.

(r * 160) + (x * 2) = 2378 16 | 94A

1001 0100 1010

2378

r = 14 c = 69

14 * 160 = 2240

~~B8000~~
- B894A
B8000
94A

AF000
994A
B894A

II. [8 Marks] What should be the values of following registers to print RED on BLUE '*' (ASCII = 0x2A) on 4th cell of 20th row of video memory using Mov [es:di] = AX

0*160
2
4*

ES = 0x B800 , DI = (3208)10, AL = 0x 2A , AH = 0x ~~2A~~ 14

III. [1 Marks] What is the total size (in bytes) of the interrupt vector table? 1KB

IV. [4 Marks] Which interrupt will be hooked after execution of following code?

0x (11)16 or (17)10

```
[org 0x100]
;;;; myISR is written here
xor ax,ax
mov es, ax
mov [es: 0x44 ] , myISR
mov [es : 0x46] , CS
mov ax, 4c00h
Int 21h
```

I * 4

V. [7 Marks] What will be the value of ax in hexadecimal after the execution of the following piece of code?

```
[org 0x100]
mov al, [num1]
mov ah, [num2+1]
add al, ah
num1: dw 0x1213
num2: dw 0x00FF
```

Solution: Ax = 0X13

B F
Bl I RGB RGB
0 0 001 100
0 0 100 001
OC

B F
BIRGB IRGB
0 001 0 100

1 4
0100 0 0 01
4

0X13
+0X00

dec 17

44
4×10¹ + 4×10⁰
40 + 4

17
4 | 68
48
28

17

CLO # 2: Describe the working of important x86 assembly primitives, including arithmetic, branching, bit manipulation, addressing modes and interrupt handling.

Q2: [30 Marks] Short Questions.

I. [8 Marks]

Part (a): What will be the values of following registers such that SCAS successfully finds 1st non-space character in the 1st row of display memory starting from end of the 1st row. Consider whole screen has 0x07 attribute.

ES = B800 1 mark DI = 158 2 marks DS = — SI = —

AX = 0720 1 mark CX = 80 1 marks

1 mark Part (b): std scas w ; write appropriate scan string instruction to complete this task

1 mark Part (c): What is appropriate instruction for this search? CLD or STD STD

II. [8 Marks] The assembly code is provided in method 1 (column 1) to calculate the sum of all the elements of an array? Optimize this program (with respect to number of code lines) by using displacement addressing modes.

Method 1: increment bx to advance to each value

```
List  db        10h, 20h, 30h, 40h
sum   db        0

mov   bx, List
mov   al, [bx]          ; AL = 10h
inc   bx                ; BX points to 20h
add   al, [bx]          ; AL = 30h
inc   bx                ; BX points to 30h
add   al, [bx]          ; AL = 60h
inc   bx                ; BX points to 40h
add   al, [bx]          ; AL = 0A0h
mov   si, sum           ; SI points to sum
mov   [si], al          ; SUM = 0A0h
```

Method 2: use bx with displacements to access each value

Solution:

```
mov cx,4
mov al,0
mov ~~bx~~ si, 0
l1: add ~~mov~~ al, [bx + si]
loop l1
```

loop 4 marks 2 of value

III. [2 Marks] What will be the value of IP after execution of following statements. Consider, initial value of CS:IP is 0CCD:0007. IP = 0X000F 2 or zero

```
BB0000          Mov BX,0
B80000          Mov AX,0
CD18            Add AX,BX
```

IV. [2 Marks] Interrupt 0x 08 works as scheduler in multitasking. 2 or zero

V. [4 Marks] An Elaborate Multitasking example 10.2 is provided in the book. Make the following change to the example. *In your solution only following is required:*

Lines of code which need to be removed
Lines of code which need to be modified along with modification
Lines of code which need to be added.

No credit will be given for anything else.

0

Decrease the number of processes from 32 to 16.

Pcb : times 16 x 16 dw 0

Stack : times 16 x 256 dw 0

init pcb : cmp bx, 16

3 + ① - if al ans corr

b. [6 Marks] Answer the following questions for the code segment given below:

```
1.  [org 0x100]
2.  Jmp start
3.  Abc:
4.  Push bp,
5.  Mov bp,sp
6.  Pop ax
7.  Pop bx
8.  Sub  bx ,[bp+10]
9.  Sub ax,[bp+8]
10. Add bx,ax
11. Ret 4
12.     Start:
13.     Mov ax,2
14.     Mov bx,3
15.     Push ax
16.     Push bx
17.     Mov ax,4
18.     Mov bx,5
19.     Push ax
20.     Push bx
21.     Call abc
22.     Mov ax,0x4c00
23.     Int 0x21
```

a. Is this program clearing the stack properly?

- Yes
- [✓ No]

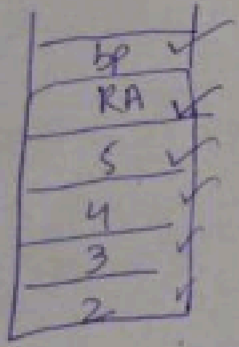
2

x

4

b. What is the value of bx after execution of **line #10.**

BX= (RA − 2) + (bp − 3)



ax → bp

bx → RA

bx = (RA − 2)

ax = (bp + 3)

bx = (RA − 2) + (bp − 3)

```
; multitasking and dynamic thread registration
[org 0x0100]
jmp start
message1: db 10, 13, 'Exiting... $'
message2: db 10, 13, 'ESC Pressed... $'
rowno: dw 0
attribute: dw 0x0720
tickcount: dw 0
exitFlag: dw 0
oldkbisr: dd 0
oldtimer: dd 0
kbisr:
                        push ax
                        push dx

                        in al, 0x60 ; read char from keyboard port

                        cmp al, 0x01
                        jne exitkbisr
                      mov word [cs:exitFlag], 1; set flag to start printing
exitkbisr: mov al, 0x20
                        out 0x20, al ; send EOI to PIC
                        pop dx
                        pop ax
                        iret ; return from interrupt
StartScreen:
push es
push ax
push cx
push di

mov ax, 0xb800
mov es, ax ; point es to video base

xor di, di ; point di to top left column mov

cx, 2000 ; number of screen locations


loop1:

mov ax, [es:di] ; space char in normal

attribute mov ah, 0x70

mov [es:di], ax

add di,2

loop loop1

pop di

pop cx

pop ax

pop es

ret

ChangeAttributeByRowNo:

push bp

mov bp, sp

pusha
mov ax, 0xb800

mov es, ax ; point es to video base

mov al,80

mul byte [bp+6] ; point di to top left

column shl ax, 1

mov cx, 80

mov bx, [bp+4]

mov di, ax

loop2:

mov ax, [es:di] ; space char in normal

attribute mov ah, bh

mov [es:di], ax
```

```
add di,2

loop loop2

popa

pop bp

ret 4

;---------------------------------------------------- ;

subroutine to print a number at top left of

screen ; takes the number to be printed as its

parameter

;----------------------------------------------------

printnum: push bp

mov bp, sp

push es

push ax

push bx
push cx

push dx

push di

mov ax, 0xb800

mov es, ax ; point es to video base

mov ax, [bp+4] ; load number in ax

mov bx, 10 ; use base 10 for

division mov cx, 0 ; initialize count

of digits

nextdigit: mov dx, 0 ; zero upper half of

dividend div bx ; divide by 10

add dl, 0x30 ; convert digit into ascii

value push dx ; save ascii value on

stack

inc cx ; increment count of values

cmp ax, 0 ; is the quotient zero
```

```
jnz nextdigit ; if no divide it again

mov di, 140 ; point di to 70th column

nextpos: pop dx ; remove a digit from the

stack mov dh, 0x07 ; use normal attribute

mov [es:di], dx ; print char on screen

add di, 2 ; move to next screen location

loop nextpos ; repeat for all digits on

stack pop di

pop dx

pop cx

pop bx

pop ax
pop es

pop bp

ret 2

;-----------------------------------------------------

timer: push ax

                push bx


                cmp word[cs:exitFlag], 1

                je exit


                inc word [cs:tickcount]; increment tick count


                push word [cs:tickcount]

                call printnum ; print tick count


                cmp word [cs:tickcount], 1

                jne exit

                mov word [cs:tickcount], 0
```

```
            mov bx, [cs:attribute]
            ror bh, 4
            mov [cs:attribute], bx


            push word [cs:rowno]
            push word [cs:attribute]
            call ChangeAttributeByRowNo
            inc word [cs:rowno]
            cmp word [cs:rowno] , 25
            jne skip2
            mov word [cs:rowno], 0
    skip2: mov bx, [cs:attribute]
            ror bh, 4
            mov [cs:attribute], bx


            push word [cs:rowno]
            push word [cs:attribute]
            call ChangeAttributeByRowNo
exit:
            mov al, 0x20
            out 0x20, al ; end of interrupt


            pop bx
            pop ax
            iret ; return from interrupt
start:
call StartScreen
push word [cs:rowno]
push word [cs:attribute]
call ChangeAttributeByRowNo
```

```
xor ax, ax
                    mov es, ax
          ; point es to IVT base


                              mov ax, [es:9*4]

                              mov [oldkbisr], ax
          ; save offset of old routine

                              mov ax, [es:9*4+2]

                              mov [oldkbisr+2], ax


                              mov ax, [es:8*4]

                              mov [oldtimer], ax
          ; save offset of old routine

                              mov ax, [es:8*4+2]

                              mov [oldtimer+2], ax


                              cli ; disable interrupts

                              mov word [es:8*4], timer; store offset at n*4

                              mov [es:8*4+2], cs ; store segment at n*4+2

                              mov word [es:9*4], kbisr; store offset at n*4

                              mov [es:9*4+2], cs ; store segment at n*4+2

                              sti ; enable interrupts


myloop: cmp word[cs:exitFlag], 1

                              jne myloop


                              xor ax, ax

                              mov es, ax ; point es to IVT base
                              mov ax, [cs:oldkbisr]
          ; read old offset in ax

                              mov bx, [cs:oldkbisr+2]
          ; read old segment in bx
```

```
                        mov cx, [cs:oldtimer]
        ; read old offset in ax
                        mov dx, [cs:oldtimer+2]
        ; read old segment in bx


                        cli
                ; disable interrupts
                        mov [es:9*4], ax
        ; restore old offset from ax
                        mov [es:9*4+2], bx
        ; restore old segment from bx
                        mov [es:8*4], cx
        ; restore old offset from ax
                        mov [es:8*4+2], dx
        ; restore old segment from bx


                        sti



MOV AX, 0x4C00
        INT 0x21
```