Name: ____________________ Roll Number: ____________________ Section: ____________________

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Computer Organization and Assembly Language**	**Course Code:**	**EE2003**
		Semester:	**Fall 2021**
Program:	**BS (CS, DS)**	**Total Marks:**	**30**
Duration:	**60 Minutes**	**Weightage:**	**15**
Paper Date:	**2-Nov-2021**	**Page(s):**	**8**
Section(s):	**All**	**Section:**	_________
Exam:	**Midterm II**	**Roll No:**	_________

Instruction/Notes:

- Exam is Open book, Open notes.
- Properly comment your code.
- You **CANNOT** use an instruction **NOT** taught in class.
- If there is any ambiguity, make a reasonable assumption. Questions during the exam are not allowed.
- Write your answer in the space provided. You **can take extra sheets BUT they WON'T BE ATTACHED WITH THE QUESTION PAPER OR MARKED.**
- All other rules pertaining to examinations as per NUCES policy apply.

Question 1 [15 Marks]: Short Questions

i. **[2 marks]** Consider a subroutine TempSBR that uses the stack to return three output values *(each of size 1 word)* through the stack. Write a statement that will create the space for these three output variables before calling this TempSBR.

```
sub sp, 6
```

Name: ____________________ Roll Number: ____________________ Section: ____________________

ii. **[6 marks]** Consider the following subroutine, which calculates the factorial of a number *(size = 1 word)* placed at the stack as a parameter and outputs the answer on the stack *(size = 1 word).* However, the code has some logical errors. Correct those errors so that the required functionality can be achieved. You can ADD or MODIFY existing lines, but you cannot REMOVE any line.

```
factorial:
  push bp
  mov bp, sp
  push ax
  push bx
  push dx

  mov ax, [bp+8]; copying the input
  cmp ax, 0
  ja L1

  mov word [bp+10], 1; returning the result
  jmp L2

L1:
  sub sp, 2
  dec bp
  push bp; passing parameter for recursive subroutine
  call factorial; recursive subroutine call

returnFact:
  pop bx
  mov dx, 0
  inc ax
  mul bx

  mov [bp+10], ax; returning the result

L2:
  pop dx
  pop bx
  pop ax
  pop bp
  ret 6
```

```
; Rewrite your code here

factorial:
  push bp
  mov bp, sp
  push ax
  push bx
  push dx

  mov ax, [bp+4]
  cmp ax, 0
  ja L1

  mov word [bp+6], 1
  jmp L2

L1:
  sub sp, 2
  dec ax
  push ax
  call factorial

returnFact:
  pop bx
  mov dx, 0
  inc ax
  mul bx

  mov [bp+6], ax

L2:
  pop dx
  pop bx
  pop ax
  pop bp
  ret 2
```

Name: ____________________ Roll Number: ____________________ Section: ____________________

iii. **[3 Marks]** Consider the code given below, write out the sequence in which the instructions are executed. Each executable instruction in code is numbered so your answer should be as follows:

Sample answer:

Instructions executed in following order

I11

I6

I10

….

You also have to briefly explain the working of this program.

	[org 0x0100]	Order of execution:
I1	jmp start	I1
		I10
	my_rout:	I2-I8
I2	mov ax, 0x8434	
I3	mov bl, 0x85	While executing I8, the division instruction generates a quotient that doesn't fit into the destination register. Hence, the program calls the ISR for INT0. However, after executing the corresponding ISR, the IRET of ISR will take control back to the DIV instruction in I8 which will again generate the same interrupt. Thus the program will be stuck in this infinite back and forth call to the ISR.
I4	div bl	
I5	mov ax, 0xffff	
I6	mov dx, 0x0100	
I7	mov bl, 0x3	
I8	div bl	
I9	ret	
	start:	
I10	call my_rout	
I11	mov ax, 0x4c00	
I12	int 0x21	

Name: ____________________ Roll Number: ____________________ Section: ____________________

iv. **[4 Marks]** In the code given below, we are copying the data of video memory from one location to another using string instructions. As a result of the execution of this code, what will be the changes on the screen?

```
[org 0x0100]
        jmp start

movepixels:
        push ax
        push bx
        push cx
        push si
        push di
        push es
        push ds

        mov ax, 0xb800
        mov es, ax
        mov ds, ax
        mov si, 0
        mov di, 80
        mov bx, 0

        ; (code is continued in the second column)
```

```
loop1:
        mov cx, 80
        cld
        rep movsb
        add si, 80
        add di, 80

        add bx, 1
        cmp bx, 25
        jne loop1

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

start:
        call movepixels

        mov ax, 0x4c00
        int 0x21
```

Solution: The left half of the screen is copied onto the right half.

Name: ____________________ Roll Number: ____________________ Section: ____________________

Question 2 [15 Marks]: Draw a triangle with two given points i.e. A (x1, y1) and B (x2, y2).

i. [3 Marks] Triangle must be isosceles (two sides equal) and right (one 90-degree angle), for that purpose check two conditions given below.

a) y1 must be less than y2 and x1 must be less than x2.

b) (x2-x1) must be equal to (y2-y1).

No need to check other conditions as these two conditions are enough.

ii. [2 Marks] Clear screen with white background.

iii. [7 Marks] Only print the boundary of the triangle with red color and asterisk character (ASCII= 2A-Hex,42-Decimal).

Hint: Write a generic subroutine to print an asterisk on a single point. Use loops to print borders.

iv. [3 Marks] Write a program with proper subroutine names and stack implementation is compulsory for parameter passing.

Note: You can't use software interrupts. You should use hard code inputs but functions should be generic. It should run properly on any inputs.

Example 1:

Input: A (7, 8) and B (10, 11)

Output :

```
(7,8)
  *
  *    *
  *         *
  *    *    *    *
                (10,11)
```

Example 2:

Input: A (10, 11) and B (7, 8)

Output: No printing on screen

Example 3:

Input: A (7, 8) and B (10, 10)

Output: No printing on screen

Name: ____________________ Roll Number: ____________________ Section: ____________________

Write your code below

```
[org 0x0100]
                jmp start
asterisk:   db  '*'

ClearScreen:    push es
                push ax
                push cx
                push di
                mov ax, 0xb800
                mov es, ax
                xor di, di
                mov ax, 0x7020 ; white background clear screen
                mov cx, 2000
                cld
                rep stosw
                pop di
                pop cx
                pop ax
                pop es
                ret

PrintAsterisk:  push bp             ; print asterisk at a specific coordinate
                mov bp, sp
                push es
                push ax
                push si
                push di

                mov ax, 0xb800
                mov es, ax
                mov al, 80
                mul byte [bp+8]
                add ax, [bp+10]
                shl ax, 1
                mov di, ax
                mov si, [bp+4]
                mov ah, [bp+6]
                mov al, [si]
                mov [es:di], ax

                pop di
                pop si
                pop ax
                pop es
                pop bp
                ret 8

TrianglePrint:  push bp
```

```
                mov bp,sp
                sub sp,2
                push ax
                push bx
                push cx
                push dx

                mov ax, [bp+6]
                cmp ax, [bp+10]                 ; compare x1, x2
                jbe end
                sub ax, [bp+10]         ; sub x1 from x2
                mov bx, [bp+4]
                cmp bx, [bp+8]          ; compare y1, y2
                jbe end
                sub bx, [bp+8]          ; sub y1 from y2
                cmp ax, bx              ; compare subtracted result
                jne end

                mov [bp-2], ax          ; local variable for difference used in printing
                add ax, 1               ; print first side
                mov cx, ax
                mov dx, -1
Disp1:          mov ax, [bp+10]
                mov bx, [bp+8]
                inc dx
                add bx, dx
                push ax
                push  bx
                mov ax, [bp+14]                 ; print red asterisk
                push ax
                mov ax, [bp+12]
                push ax
                call PrintAsterisk
                loop Disp1

                mov cx, [bp-2]          ; print 2nd side
                mov dx, -1
Disp2:          mov ax, [bp+6]
                mov bx, [bp+4]
                inc dx
                sub ax, dx
                push ax
                push  bx
                mov ax, [bp+14]                 ; print red asterisk
                push ax
                mov ax, [bp+12]
                push ax
                call PrintAsterisk
                loop Disp2
```

Name: ____________________ Roll Number: ____________________ Section: ____________________

```
            mov cx, [bp-2]          ; print 3rd side
            sub cx, 1
            ;dec cx, 1
            mov dx, 0
Disp3:      mov ax, [bp+10]
            mov bx, [bp+8]
            inc dx
            add ax, dx
            add bx, dx

            push ax
            push  bx
            mov ax, [bp+14]         ; print red asterisk
            push ax
            mov ax, [bp+12]
            push ax
            call PrintAsterisk
            loop Disp3

end:        pop dx
            pop cx
            pop bx
            pop ax
            mov sp, bp
            pop bp
            ret 12

start:      call ClearScreen
            mov ax, 74              ; red asterisk
            push ax
            mov ax, asterisk        ; passing character asterisk
            push ax
            mov ax, 7               ; x1 coordinate of point 1
            push ax
            mov ax, 8               ; y1 coordinate of point 1
            push ax
            mov ax, 10              ; x2 coordinate of point 2
            push ax
            mov ax, 11              ; y2 coordinate of point 2
            push ax
            call TrianglePrint
            mov ax, 0x4c00
            int 0x21
```