National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Computer Organization and Assembly Language	Course Code:	EE213
Program:	BS(Computer Science)	Semester:	Spring 2018
Duration:	60 Minutes	Total Marks:	35
Paper Date:		Weight	15%
Section:	ALL	Page(s):	
Exam Type:	Mid-2		

Student : Name:_______________________________ Roll No.________________ Section:_______

Instruction/Notes:

1. Exam is Open book, Open notes.
2. Properly comment your code.
3. You **CANNOT** use an instruction **NOT** taught in class.
4. Write your answer in the space provided. You **can take extra sheets BUT they WONT BE ATTACHED WITH THE QUESTION PAPER OR MARKED.**
5. No need to copy code from book. If you need any code/part of code, just mention the line numbers and page no.

Q1. Write code to clear IF (i.e. IF=0) without using cli instruction.

Solution:

```
pushf
pop ax
and ax, 0xFDFF
push ax
popf
```

Q2. Write a SubStr function that extracts a substring from video memory and places it in DS. The subroutine is passed the following parameters: The row number of video screen where string is placed, the column number, the length of string, the starting position (i.e. character no.) of substring on video screen, length of substring and address of substring array in DS. **You have to do this using String Instructions ONLY. No credit for doing it without string instructions. Stack is also made for your ease. Order of parameters should remain the same.**

Return Value	← SP
Address of string in DS	
Length of Substring	
Starting position i.e. char of string on video memory	
Length of string printed on video memory	
Column no.	
Row no.	

```
substr:
                push bp
                mov bp, sp
                push es
                push ds
                push ax
                push cx
                push si
                push di
                push bx


                mov al, 80
                mul byte [bp+14]                ;y position i.e. row
                add ax, [bp+12]                 ;x position      i.e. column
                add ax, [bp+8]                  ;character no
                shl ax, 1                       ;convert to byte
                mov si,ax                       ;store in si

                mov di, [bp+4]                  ;di points to string array in DS

                ;following code makes sure the sub string length never exceeds its total length
                mov bx, [bp+10]                         ;string length
                sub bx, [bp+8]                  ;now bx has remaining no of char from substr position
                cmp bx, [bp+6]                  ;compare with substr length
                jg l1
                mov cx, bx                      ;move the remaining characters of  string in cx
                jmp l2
                l1:
                mov cx, [bp+6]                  ;move substring length in cx
                l2:

                ;now cx has required substring length
                mov bx, 0xb800
                mov ds, bx

                ;lods from video screen, stos in DS
                l3:
                lodsb
                add si, 1
                stosb
                loop l3

                pop bx
                pop di
                pop si
                pop cx
                pop ax
                pop ds
                pop es
                pop bp
                ret 12
```

Q3. Write a software interrupt service for int 0x58 that receives three arguments via registers: a number k in ax register, a segment value in dx register, and an offset value in bx register. The service replaces the offset and segment values for interrupt number k in the IVT with the ones passed in bx and dx registers respectively. Basically int 0x58 will be used to 'hook' the kth interrupt. Note: the service maybe used to hook a software or hardware interrupt. **(5 Marks)**

Here is an example of how int 0x58 may be used:

```
mov bx, myISRX  ;offset of the ISR
mov dx, CS      ;segment of the ISR
mov ax, 0x31        ;hook int 0x31
int 0x58
```

```
int58:
push si
push cx
push es

xor cx, cx
mov es, cx
mov cl, 4
mul cl
mov si, ax

mov [es:si],bx
add si, 2
mov [es:si], dx


pop es
pop cx
pop si
iret
```

Q4. Dry run the following code and write in one line precisely what the code is doing?

```
    mov ax, 0xb800
    mov es, ax
    mov di, 2560
    sub di, 2
    mov ax, 0x0720
    mov cx, 480
    std
    rep stows
```

Solution:

The code is clearing the screen from line 10 to 15. (i.e. from 2558 to 1600) OR 11th line to 16th line (note: the code ends with di at 1598 which means 1598 is not changed location)