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):	3
	Exam Type:	Mid-1		

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 to Q5 carry 5 marks each.

Q1. To address 1-MB of memory we need 20 bits of addressing, then how much megabytes of memory can be accessed using 30 bits of addressing. ___1024_____ MB.

Q2. Given the following jump statement and its opcode, identify the type of jump (near or short) and the offset (logical address in CS) to which the jump will take place.

Offset of Opcode	Opcode	Type of Jump?	Offset ?
0125	EBE9 ;EB is opcode of jump	Short	0110

Q3. Given the following sequential set of instructions of same program, write down the values of CF, PF, ZF after each instruction: (initially all flags are zero.)

	CF	PF	ZF
xor ah, ah	__0_____	_ _ 1_____	__ 1_____
mov al, 0x4A	__0_____	___1_____	___1_____
shl al, 2	__1_____	___1_____	___0____
rcr ah, 3	__0_____	___1_____	___0_____
sub ah, al	__1_____	___0_____	___0_____

Q4. Write an instruction which reads the first byte of its own op-code and stores it into al register.

L1: mov al, [l1]

Q5. Write assembly code to compare two 32-bit numbers such that if num1 is equal to equal to num2 it sets ZF=1 else ZF=0. (Declare two 32-bit numbers in memory to compare)

OR

```
[org 0x0100]
mov ax, [num1]
sub [num2], ax
jnz exit
mov ax, [num1+2]
sub [num2+2], ax

exit:
mov ax, 0x4c00
int 21h
num1: dd 0xABCDEF01
num2: dd 0xEF01ABCD
```

```
[org 0x0100]
mov ax, [num1]
cmp [num2], ax
jnz exit
mov ax, [num1+2]
cmp [num2+2], ax

exit:
mov ax, 0x4c00
int 21h
num1: dd 0xABCDEF01
num2: dd 0xEF01ABCD
```

Q6. Write an assembly program, such that given an array of **ten** integers (each integer is stored as word), your program finds and stores the sum of unique elements of array in a memory label called **"sum"** (defined word).

```
    [org 0x0100]

    jmp start

    data: dw 3,3,3,3,3
    swap: db 0
    sum: dw 0

    start:
        mov bx, 0               ; initialize array index to zero
        mov byte [swap], 0      ; rest swap flag to no swaps

    loop1:
        mov ax, [data+bx]       ; load number in ax
        cmp ax, [data+bx+2]     ; compare with next number
        jbe noswap              ; no swap if already in order
        mov dx, [data+bx+2]     ; load second element in dx
        mov [data+bx+2], ax     ; store first number in second
        mov [data+bx], dx       ; store second number in first
        mov byte [swap], 1      ; flag that a swap has been done
noswap: add bx, 2               ; advance bx to next index
        cmp bx, 8               ; are we at last index
        jne loop1               ; if not compare next two
        cmp byte [swap], 1      ; check if a swap has been done
        je start                ; if yes make another pass
```

```
     mov bx, 0
     mov cx, [data+bx]      ;store first element in sum

l1:  mov ax, [data+bx]      ;store in ax to make it prev. ax =  prev
     cmp ax, [data+bx+2]
     je skip

     add cx, [data+bx+2]
skip:
     add bx, 2
     cmp bx, 8
     jne l1

mov [sum], cx

mov ax, 0x4c00
int 21h
```

💣☠ GOOD LUCK! 💣☠