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 2019
Duration:	60 Minutes	Total Marks:	35
Paper Date:	12th April, 2019	Weight	15%
Section:	ALL	Page(s):	4
Exam Type:	Mid-2		

Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes:

1. Exam is Open book, Open notes.
2. Properly comment your code.
3. Syntax error will result in **negative** marking.
4. Write your answer in the space provided. You **can take extra sheets BUT they WONT BE ATTACHED WITH THE QUESTION PAPER OR MARKED.**

Q1. Short questions.

Part A) MCQs. Tick one answer only. NO CUTTING/OVER WRITING. AMBIGOUS ANSWERS WILL NOT BE CONSIDERED. **[5X1 Marks]**

1. Which of the following is not a valid jump instruction?
   a. Jcxz
   b. Jne
   c. Jncxz
2. When we set direction flag to 1, it will decrease the indexes for:
   a. Only the immediate one string instruction after it
   b. For all string instructions after it
   c. For all string instructions before and after it
3. When an interrupt occurs, the following are push on the stack in this order:
   a. Flags, CS, IP
   b. IP, CS, Flags
   c. IP, Flags, CS
4. Ret 4 results in the following
   a. Decreases sp by 4 bytes
   b. Increases sp by 4 bytes
   c. None of the above
5. A "Division by Zero" interrupt is generated only:
   a. when the operand of the "DIV" instruction has a value of zero.
   b. when the quotient cannot fit in the destination register(s).
   c. when the "DIV" instruction produces a quotient equal to zero.

Part B) True/False. **[1x2 Marks]**

1. loop l1 is equivalent to the these two instructions:
   DEC CX
   JNZ L1

True	False

2. Total size of IVT is 1MB.

True	False

Part C) Short questions. **[8+5 Marks]**

1. A memory location has an address 0xB8B7C. It represents a location on video screen. By showing complete calculation, determine, the row and column number on video memory that this location represents?

0xB8B7C= 0xB8B7C-0xB8000
= 0xB7C
=2940
=2940/160
=18th row
=remainder: 2940- (18*160)
=60
=60/2=30th column

So 18th Row and 30th Column

2. Write a fragment of code to hook interrupt 0xA1 with your service myISR, which is in your current CS.

```
Mov [es:0xA1*4], myISR
Mov [es:0xA1*4+2], CS
```

Q2. [15 Marks] Write a subroutine **Compress Data** that takes (row, col) coordinates of a cell of video memory as parameters. The function reads character from that cell, removes all the consecutive occurrences of that character horizontally, and shifts the remaining data left (leaving spaces at the end). Assume that attribute byte is identical throughout the video memory. **You have to solve it using string instructions only.**
Sample run on a video memory of 5x5 cells:

(row,col) = (2,1) Character at (Row 2, Col 1) = 'a'	After removing consecutive occurrences of 'a' and shifting remaining data
b l n g l h E l l o b **A** a b z a P p l e m A n g o	b i n g l h e l l o b **b** **z** a p p l e m a n g o

4.5/6

FAST NATIONAL UNIVERSITY
School of Computing
Fall 2022

Course Title:	Computer Organization and Assembly Language
Task:	Quiz 1
Section:	BCS-3A
Date:	14th Sep, 2022

21L-5160

Q1. For each of the following words, identify the byte that is stored at lower memory address and the byte that is stored at higher memory address in big and little endian format:

(3)

0x5688

Big Endian
Lower Memory 56
Higher Memory 88

Little Endia
Lowe Memory : 88
Higher Memory : 56

Q2. Calculate the physical address that is generated by the following segment offset pairs:

(3)

1DEF:0001

1DEF0
0001
1DEF1
F

1.5

National University of Computer and Emerging Sciences

COAL Lab Midterm

Computer Organization and Assembly Language

Time Allowed	90 Minutes	Student Name	
Maximum Marks	100	Roll Number	
Lab Instructors		Date	26th

Before you start make sure:

1. **Fill word Format on GCR & submit PDF [Only PDF formats accepted]**
2. **PLAGIARISM WILL BE MARKED ZERO WITH NO RETAKE**

Activity 1: [20 Marks]

Initialize a memory array with last 4 digits of **Your Own Roll Number** (for example, if your roll number is **16L-4195** then memory array should be initialized with **{4,1,9,5}**).
Then write a subroutine **LoadMN** which stores the matrices M and N using the values as described in the Code section below:

Activity 2: [50 Marks]

Use the subroutine from **Activity 1**, to initialize matrices **M** and **N.**

Theory:
If M and N are two matrices of order 2×2 as shown below

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$
$$N = \begin{bmatrix} d & c \\ b & a \end{bmatrix}$$

Convolution of two matrices can be calculated by padding first matrix and scrolling the second matrix on it across the rows and columns of A and performing dot product:

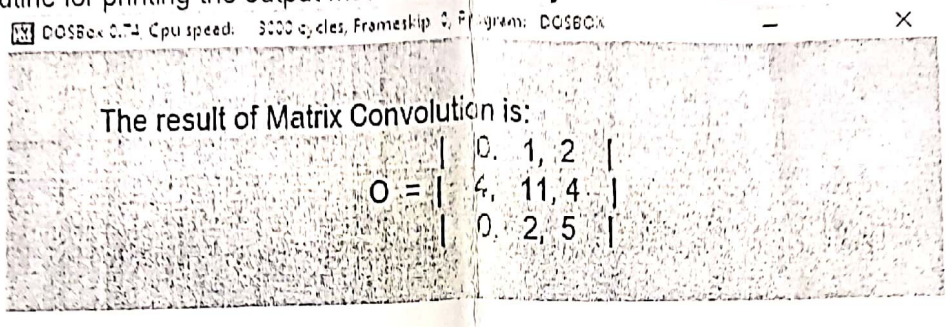
$$O = MN = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & a & b & 0 \\ 0 & c & d & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} d & c \\ b & a \end{bmatrix} = \begin{bmatrix} a^2 & 2ab & b^2 \\ 2ac & 2(ad+bc) & 2bd \\ c^2 & 2cd & d^2 \end{bmatrix}$$

As shown above, convolution of M and N is the matrix O.

Write a subroutine, **MatConv,** which performs 2x2 matrix **convolution** using the values of M and N initialized in the **Activity 1.**

Activity 3: [30 Marks]

Make a subroutine for printing the output matrix from Activity 2 on the console as shown below



Code:

Your final program structure should look like:

```
[org 0x100]
roll: db 4,1,9,5          ; Change to last 4 digits of your roll number
a: db 0                   ; store roll
b: db 0                   ; store roll+1
c: db 0                   ; store roll+2
d: db 0                   ; store roll+3
; rows of M matrix of order 4X4
Mr1: db 0, 0, 0, 0        ;
Mr2: db 0, 0, 0, 0        ; store 0, a, b, 0
Mr3: db 0, 0, 0, 0        ; store 0, c, d, 0
Mr4: db 0, 0, 0, 0        ;
; rows of N matrix of order 2X2
Nr1: db 0, 0              ; store d, c
Nr2: db 0, 0              ; store b, a
; rows of O matrix of order 3X3
Or1: dw 0, 0, 0
Or2: dw 0, 0, 0
Or3: dw 0, 0, 0
LoadMN:
; Write code for LoadMN
MatConv:
; Write code for Matrix Convolution
PrintMat:
; Write code for Printing Matrix
start:
;_____
; Write Calls to the subroutines & other codes here
;_____
mov ax, 0x4c00
int 0x21
```