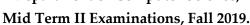
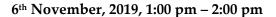


FAST- National University of Computer & Emerging Sciences, Karachi.

Department of Computer Science,







Course Code: EE 213 Course Name: Computer Organization and Assembly Language
Instructors: Dr. Nouman M Durrani, Muhammad Danish Khan, and Syed Zain-ul-Hassan

SOLUTION

Instructions:

- Except your Roll No and Section, DO NOT SOLVE anything on this paper.
- Return the question paper.
- Read each question completely before answering it. There are **3 questions on 2 pages**.
- In case of any ambiguity, you may make assumption but your assumption must not contradict any statement in the question paper.
- All the answers must be solved according to the SEQUENCE given in the question paper, otherwise points will be deducted.
- This paper is subjective.
- Where asked for values, only provide the hex-decimal values.
- Problems needing iterations should be coded using iterative instructions. No points will be awarded otherwise.

Time Allowed: 60 minutes.

Maximum Points: 30 points

Q No. 1(a) Write X86 assembly code snippet for each of the following set of operations:

[2 + 2 + 1 = 5 Points]

i. Exchange the two most recent elements on the stack segment. Do not use PUSH and POP instructions

```
MOV EAX, [ESP]
XCHG EAX, [ESP+4]
MOV [ESP], EAX
```

ii. Loop through a BYTE array X1, and calculate sum of all its elements' squares (X2) into a variable named SUM.

```
MOV
          SUM, 0
           ECX, LENGTH X1
MOV
           ESI, OFFSET X1
MOV
L1:
          MOV
                 AL, [ESI]
          MUL
                  ΑL
          ADD
                  SUM, AX
          ADD
                  ESI, TYPE X1
L<sub>0</sub>OP
          L1
```

iii. How you'll copy the return address of a procedure Factorial into a register EBX?

```
Factorial PROC BBX, [ESP]
```

(b) Show the hex-decimal values of the required registers/flags after execution of the following independent instructions:

[1X5 = 5 Points]

```
i. MOV AL, 084H
                         ii. MOV AL, 0D4H
                                                   iii. MOV AL, OD1H
                             ROL AL, 4
  SAR AL, 2
                                                        SHR AL, 1
                                                        RCR AL, 3
   ; AL = E1
                              ; AL = 4D
                                                        ;AL = 2D
   ; CF = 0
                              ; CF = \underline{1}
                                                         ; CF = 0
      iv. MOV AL, 00001111B
                                            v. MOV AX, 0D4FCh
          SHR AL, 1
                                                 MOV CX, 770Ch
          TEST AL, 0000001B
                                                 SHRD AX,CX, CL
          ;AL = 07
                                                 ; AX = 70CD
                                                 ; CF = 0
          ; CF = 0
          ; ZF = 0
```

Q No. 2 (a) Translate the following code into assembly language equivalent code. Your code must clean the stack. Also don't use LOCAL directive. [4 Points]

```
Func PROTO, mem32: DWORD
                                 func PROC, mem32:DWORD
                                       push
                                              ebp
                                              ebp, esp
main
      PROC
                                       mov
                                              ebx, 2
      PUSH EBP
                                       mov
            EBP, ESP
                                              eax, mem32
      MOV
                                       mov
            ESP, 4
      SUB
                                       mul
                                              ebx
            [EBP-4], 0
                                              eax, 5
      MOV
                                       cmp
            [EBP-4]
      PUSH
                                              11
                                       jb
      CALL
            func
                                       pop
                                              ebp
            ESP, EBP
      MOV
                                       add
                                              esp, 4
            ESP, 4
      ADD
                                       ret
            EBP
      POP
                                       11:
                                                    mem32
                                              inc
      RET
                                       invoke func, mem32
                                 func
main ENDP
                                       ENDP
```

(b) Assuming ESP = 0FFF0h initially, for the procedure in Part A, draw stack frame for each function /recursive function call. [4 Points]

0FFEC	Ret address(system)		
0FFE8	EBP	Main's Stack Frame	
0FFE4	0 (int i)		
0FFE0	0 (mem32)		
0FFDC	Return Address (MAIN)	Func's Stack Frame#1	
0FFD8	EBP		
0FFD4	1 (mem32)		
0FFD0	Return Address (FUNC#1)	Func's Stack Frame#2	
ØFFCC	EBP		
0FFC8	2 (mem32)		
0FFC4	Return Address (FUNC#2)	Func's Stack Frame#3	
0FFC0	EBP		
0FFBC	3 (mem32)		
0FFB8	Return Address (FUNC#3)	Func's Stack Frame#4	
0FFB4	EBP		
_			
0FFB0	4 (mem32)		
0FFAC	Return Address (FUNC#4)	Func's Stack Frame#5	
0FFA8	EBP		

- Q. No. 3 (a) Suppose the following data is received from a wireless sensor node operating in a smart building and is stored in EAX register, as shown in Figure 1. You are required to write an assembly language program with the corresponding data definition directives that would extract the data items and store them at memory locations Status, Sequence_Number, Revision_Count, and Sensor_Data. [4 Points]
 - i) Bit 0 is the Status of the sensor flag (0 Forwarded Data and 1 Sensed Data)
 - ii) Bits 1 to 12 reflect an integer Sequence_Number of the packet being sent.
 - iii) Bits 13 15 show an integer Revision_Count of the packet.
 - iv) Bits 16 31 contain the Sensor_Data.

16 bits	3 bits	12 bits	1 bit
Sensor_Data	Revision_	Sequence_Nu	Status
	Count	mber	

```
EBX, EAX
MOV
    EBX, 01
AND
MOV
    Status, EBX
    EBX, EAX
MOV
AND
    EBX, 1FFEh
                      EBX, 1
SHR
MOV
    SEQ_NUMBER, EBX
MOV
    EBX, EAX
                      EBX, E000h
AND
SHR
    EBX, 13
    REVISION_COUNT, EBX
MOV
MOV
    EBX, EAX
SHR
    EBX, 16
MOVZX SENSOR_DATA, BX
```

(b) Assuming the microprocessor uses some string primitive instruction to transfer data from one memory location to another. If the source data is located at the address C181 00C0H, read 100 WORD from this location and save them to a memory location starting at the address 100F 0778h. You must use string primitive instruction(s) to transfer the data.

[4 Points]

```
MOV ESI, C181 00C0H
MOV EDI, 100F 0778h
MOV ECX, 100
REP MOVSW
```

(c) Consider the following array declarations:

```
array1 WORD 23, 34, 45, 56, 67, 78, 89 array2 WORD 7 DUP (?)
```

Write a program that uses string handling instructions to fetch each element of array1 and save it to array2 after multiplying it by 3. [4 Points]

```
multiplier,3
mov
      esi,OFFSET array1
mov
      edi,esi
mov
      ecx, LENGTHOF array1
mov
cld
      lodsw
                              ; load [ESI] into AX
L1:
      mul multiplier
                              ; multiply by a value
      stosw
                               ; store AX into [EDI]
loop L1
```

STAY BRIGHT