FAST- National University of Computer & Emerging Sciences, Karachi.
Department of Computer Science,
Mid Term II Examinations, Fall 2019.
6th November, 2019, 1:00 pm – 2:00 pm

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 (X^2) into a variable named SUM.

```
MOV     SUM, 0
MOV     ECX, LENGTH X1
MOV     ESI, OFFSET X1
L1:     MOV   AL, [ESI]
        MUL   AL
        ADD   SUM, AX
        ADD   ESI, TYPE X1
LOOP    L1
```

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

```
Factorial     PROC
   MOV        EBX, [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
SAR AL, 2
;AL = E1
;CF = 0
```

ii.
```
MOV AL, 0D4H
ROL AL, 4
;AL = 4D
;CF = 1
```

iii.
```
MOV AL, 0D1H
SHR AL, 1
RCR AL, 3
;AL = 2D
;CF = 0
```

iv.
```
MOV AL, 00001111B
SHR AL, 1
TEST AL, 00000001B
;AL = 07
;CF = 0
;ZF = 0
```

v.
```
MOV AX, 0D4FCh
MOV CX, 770Ch
SHRD AX,CX, CL
;AX = 70CD
;CF = 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]

```
void func(int);

void main()
{
      int i = 0;
      func(i);
}
```

```
void func(int i)
{
      cout<<i * 2;
      if(i< 5)
      {
            ++i;
            func(i);
      }
}
```

```
Func  PROTO, mem32: DWORD

main  PROC
      PUSH  EBP
      MOV   EBP, ESP
      SUB   ESP, 4
      MOV   [EBP-4], 0
      PUSH  [EBP-4]
      CALL  func
      MOV   ESP, EBP
      ADD   ESP, 4
      POP   EBP
      RET
main  ENDP
```

```
func  PROC, mem32:DWORD
      push  ebp
      mov   ebp, esp
      mov   ebx, 2
      mov   eax, mem32
      mul   ebx
      cmp   eax, 5
      jb    l1
      pop   ebp
      add   esp, 4
      ret
      l1:   inc   mem32
      invoke func,mem32
func  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)	Main's Stack Frame
0FFE8	EBP	
0FFE4	0 (int i)	

0FFE0	0 (mem32)	Func's Stack Frame#1
0FFDC	Return Address (MAIN)	
0FFD8	EBP	

0FFD4	1 (mem32)	Func's Stack Frame#2
0FFD0	Return Address (FUNC#1)	
0FFCC	EBP	

0FFC8	2 (mem32)	Func's Stack Frame#3
0FFC4	Return Address (FUNC#2)	
0FFC0	EBP	

0FFBC	3 (mem32)	Func's Stack Frame#4
0FFB8	Return Address (FUNC#3)	
0FFB4	EBP	

0FFB0	4 (mem32)	Func's Stack Frame#5
0FFAC	Return Address (FUNC#4)	
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_ Count	Sequence_Number	Status

```
MOV   EBX, EAX
AND   EBX, 01
MOV   Status, EBX

MOV   EBX, EAX
AND   EBX, 1FFEh              ; (0000 0000 0000 0000 0001 1111 1111 1110)
SHR   EBX, 1
MOV   SEQ_NUMBER, EBX


MOV   EBX, EAX
AND   EBX, E000h              ; (0000 0000 0000 0000 1110 0000 0000 0000)
SHR   EBX, 13
MOV   REVISION_COUNT, EBX

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]

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

STAY BRIGHT