FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
KARACHI CAMPUS

Grand Assignment Fall 2022

Computer Organization & Assembly Language

Total Points: **155**

Solve on this paper, and attached the program results

Roll No: Section : Signature: _____

Part I

Question No. 1: Programming Basics [10*02 = 20 Points]

Machine Language

(i) The following bytes are found in order somewhere in memory. Assuming they are machine codes, decode the values into meaningful assembly language mnemonics. [Solve this when Machine Language is covered in the class]

```
B9 00 12,8C 85 DC 01
```

(ii) Convert the following independent Assembly Language instructions to Machine Language code – give your answers in hexadecimal:

```
MOV [SI+490], SP

ADD   AL, [BX + SI]

JNZ NEXT                ; NEXT is a label at offset 0008H and

PUSH AX

MOV AX, VAR + 6   ; OFFSET of VAR is 0002H

SUB CX, VAR2      ; OFFSET of VAR2 is 0008H

INC DX
```

(iii) In the following instruction sequence, show the resulting value of AL where indicated, in hexadecimal:

```
MOV AL,7AH
NOT AL            ; a.  AL =
MOV AL,3DH
AND AL, 74H       ; b.  AL =
MOV AL,9BH
OR AL,35H         ; c.  AL =
MOV AL,72H
XOR AL,0DCH       ; d.  AL =
```

(iv) Differentiate between the following Assembly Language instructions:

```
MOV EAX, OFFSET VAR1
MOV EAX, VAR1
```

(v) List *four* important uses of the runtime stacks in programs.

(vi) Suppose EAX=1234H, EBX=5678H, ECX=9ABCH, and ESP=100H, Give the contents of EAX, EBX, ECX, and ESP after the execution of the following instructions:

```
PUSH        EAX
PUSH        EBX
XCHG        EAX, ECX
POP         ECX
PUSH        EAX
POP         EBX
```

a) EAX : b) EBX : ECX : c) ESP :

(vii) What additional instructions are generated by the assembler as a result of assembling the following procedure?

```
MYSUM PROC USES ESI ECX
      MOV ECX, 10
      L1:
      ADD EAX, [ESI]
      SUB ESI, 4
      LOOP L1
      ret
MYSUM ENDP
```

(viii) Generate a Map file for an assembly language program that has a code size of 100h bytes, data size of 50h bytes and a stack of 200h bytes. Using this map file, give the contents of CS, DS, and SS registers if this program is loaded at address of 508A0h.

(ix) The shown program sets AH to a value depending on the comparison result of unsigned integers V1 and V2. For each condition in the table below, use "√" sign to indicate which value AH will have after the program is executed. If there are more than one possibility, use "?" sign to indicate which value of AH is possible.

```
.DATA
            V1 DB(?)
            V2 DB(?)
.CODE
Start:
            .
            .
            MOV AL, V1
            CMP AL, V2
            JZ Label1
            JS Label1
            MOV AH, 1
            JMP Continue
Label1:
            JE Label2
            MOV AH, 2
            JMP Continue
Label2:
            MOV AH, 3
Continue:
            .
            .
            .
```

	AH =1	AH=2	AH=3
If V1=V2 then			
If V1<V2 then			
If V1>V2 then			

(x) Give the contents of the following registers, along with the run-time stack, when the following instructions are executed. Initially, consider ESP = 00001FF8h.

Note: SOLVE THIS PART HERE. No Marks will be awarded without proper working using the stack diagrams.

```
X1 DWORD 25H
X2 DWORD 27H
MAIN PROC
                PUSH 6H
                PUSH 5H
                CALL P1
    11500000H   MOV RESULT, EAX              ; ESP:      _____________
MAIN ENDP
    P1 PROC
    115000A4H   PUSH EBP
                MOV EBP, ESP                 ; EBP:      _____________
                MOV EAX, [EBP+8]
                ADD EAX, [EBP+12]            ; EAX:      _____________
                PUSH OFFSET X1
                PUSH OFFSET X2               ; ESP:      _____________
                POP ESI
                POP EBX
                ADD [ESI], EAX               ; X2:       _____________
                ADD [EBX], EAX               ; X1:       _____________
                MOV ESP, EBP
                POP EBP
                RET 8                        ; EIP:      _____________
    P1 ENDP
```

Part II

Q. No 2 Answer all the questions in this section. [2x22=44]

```
.DATA
BARRAY BYTE 10H, 20H, 30H, 6 DUP (0AH)
ALIGN 4
WARRAY WORD 5 DUP(1000H)
PRESSKEY EQU <"PRESS ANY KEY TO CONTINUE ...",0>
DARRAY DWORD 5 DUP(56789ABH),7 DUP(12345678H)
PROMPT BYTE PRESSKEY
```

What will be the value of EAX, and AL after executing each of the following instructions? Assume that the address of barray is 404000h.

i. `MOV EAX, TYPE WARRAY ; EAX =`

ii. `MOV EAX, LENGTHOF BARRAY ; EAX =`

iii. `MOV EAX, SIZEOF DARRAY ; EAX =`

iv. `MOV EAX, OFFSET WARRAY ; EAX =`

v. `MOV EAX, DWORD PTR BARRAY ; EAX =`

vi. `MOV AL, BYTE PTR DARRAY ; AL =`

vii. Would the following instruction set the zero flag? Explain.

```
MOV AX, 0000h              ;clear the AX register
```

viii. Is it possible for a NEG instruction to set the Overflow flag?

Consider a program that has the following data segment:

```
I EQU 2Eh, 2h
J BYTE '6789'
K EQU 140
L WORD 3412h, 8765h
M DWORD 4, 3, 5, 6, 7
```

Indicate whether the following instructions are valid or not. If valid, give the result of the operation in hexadecimal. If invalid, give the reason.

ix. `MOV AL, I+1`

x. `MOV AL, J+2`

xi. `MOVSX EAX, L[1]`

xii. `MOV EBX, M[2]`

xiii. `INC [ESI] ;ESI = OFFSET J`

xiv. `MOV I, L`

xv. `MOV EAX, DWORD PTR J`

xvi. `MOV L, WORD PTR M`

xvii. `MOV ESI, L`

xviii. Consider the following code:

```
mov ax, 0h
mov cx, 0Ah
doLoop:
dec ax
loop doLoop
```

What is the value of the *ax* register after the completion of the doLoop?

`Ax =`

xix. When an interrupt occurs, arrange the following operations in their order of occurrence?

a) interrupt service routine executed
b) the registers are restored by popping their values off of the stack
c) the processor identifies the source of the interrupt
d) the program counter and other registers' values are pushed onto the stack
e) the address of the interrupt service routine is placed in the program counter

1. 2. 3. 4. 5. [02]

xx. In the following code sequence, show the value of AL after each shift or rotate instruction has executed:

```
mov al,0D4h

shr al,1 ;                a. AL =
mov al,0D4h
sar al,1 ;                b. AL =
```

Suppose that you have the following initial register content: AX=F2E9H, BX=0002H CX=08A0H and DX=F1E0H

xxi. Show the contents of AX and the flags (CF,OF,SF and ZF) after executing:

```
ADD AX, BX     ;  a.  CF =     b. OF=     c. SF=    d.   ZF=
```

xxii. Show the contents of CX and the flags (CF,OF,SF and ZF) after executing:

```
SUB CX, DX     ;  a.  CF =     b. OF=     c. SF=    d.   ZF=
```

xxiii. Show the contents of BX and the flags (CF,OF,SF and ZF) after executing:

```
NEG BX         ;  a.  CF =      b. OF=    c. SF=    d.   ZF=
```

xx. After the execution of the following sequence of instructions, what is the value of EAX?

```
MOV AH, 9Fh
MOV AL, FFh
XOR AH,AH
OR AH,AL
```

EAX =

xxi. Write a single instruction to mask out 1st and 3rd nibble of EAX.

xxii. Compares the integers 7FFFh and 8000h and show how the JB (unsigned) and JL (signed) instructions would generate different results.

Question No.3 : Assembly Language Programming [7x5=35 Points]

(i) Implement the following pseudo-code in assembly language (Intel IA-32) . Also, give the corresponding data definition directives:

(a)

```
; All values are
; 32-bit signed integers


while (OP1 < OP2)
{
       OP1++;
if (OP3 == OP2)
X = Y + 2;
else
X = Y + 10;
}
```

(b)

```
; All values are
; 32-bit unsigned integers


if(VAL1>VAL2) AND (VAL2>VAL3) then

X=10

else

X=20
```

(ii) Write an assembly language procedure MINIMUM that is called from the MAIN procedure to find the minimum MIN among X, Y and Z. The arguments are passed by value to the procedure MINIMUM using registers. The result is also returned in a register. Also, write the corresponding data definition directives. The Intel IA 32 version of this program is required.

(iii) Suppose that there are two tables defined in the data segment, DS=2FF0H, namely Table1 and Table2. Table1 is at offset 1000H and Table2 is at offset 2000H. Both tables have a size of 100 bytes.

Solve here

(a) Write a code segment to copy the content of Table1 to Table2.

(b) Write a subroutine to search for a constant number that can be represented in a byte, in a table, and returns the index of the table where the number is found in the DI register. Assume that the constant number to be searched is pushed first in the stack, followed by the table address, and finally the size of the table. Then, write a code segment to search for the number 5 in Table1 and the number 10 in Table2, using the subroutine, and store the corresponding indices in registers AX and BX respectively.

(iv) Write an Assembly Languageprogram to compute (a) the binomial coefficients C(n, k) and Power (X, N) using the recursive definition:

(a) binomial coefficients C(n, k)

(b) Power (X, N)

```
int Power(int X, int N) {
        if( N == 0 ) return 1;
        else return Power( X, N-1) * X;
}

void main(void) {
        cout <<Power(5,2);
}
```

(v) Write an Assembly Language program to find the nth term Fibonacci Sequence:

```
01 int fibonacci(int n)
02 {
03   if(n==0) return0;
04   else
05   if(n==1) return1;
06   elsereturnfibonacci(n - 1) + fibonacci(n - 2);
07 }
08
09 int main()
10 {
11   int input;
12   cin >> input;
13   cout << fibonacci(input) << endl;
14 }
```

(vi) **EXCHANGE SORT**

The exchange sort is similar to its cousin, the bubble sort, in that it compares elements of the array and swaps those that are not in their proper positions. (Some people refer to the "exchange sort" as a "bubble sort".) The difference between these two sorts is the manner in which they compare the elements. The exchange sort compares the first element with each following element of the array, making any necessary swaps.

```
for (i = 0; i < n-1; i++)
   for (j = 0; j < n-i-1; j++)
      if (a[j] > a[j+1])
         {
          t = a[j];
          a[j] = a[j+1];
          a[j+1] = t;
         }
```

Write an assembly Language program to sort the elements using exchange sort.

(vii) SELECTION SORT

Selection sort carries out a sequence of passes over the table. At the first pass an entry is selected on some criteria and placed in the correct position in the table. The possible criteria for selecting an element are to pick the smallest or pick the largest. If the smallest is chosen then, for sorting in ascending order, the correct position to put it is at the beginning of the table. Now that the correct entry is in the first place in the table the process is repeated on the remaining entries. Once this has been repeated n-1 times the n-1 smallest entries are in the first n-1 places which leaves the largest element in the last place. Thus only n-1 passes are required. The algorithm can be described as follows:

```
for (i = 0; i < n-1; i++)
  {
     // find smallest entry in ith to n-1 th place
     // p is subscript of smallest entry yet found
   p = i;
   for (j = i+1; j < n; j++)
     if (a[j]<a[p])
       p = j;
     // exchange pth element with ith element
   t = a[p];
   a[p] = a[i];
   a[i] = t;
  }
```

For intimation, you can visit the below link:

Write an assembly Language program to sort all the elements using Selection sort.

Part III

Q. No. 4 Assembly Language [9x5= 45 Points]

(i) 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 in Intel IA 32 with the corresponding data definition directives that would extract the data items and store them at memory locations Sequence_Number, Revision_Count, Status, and Sensor_Data.

a) Bits 0 to 11 reflect an integer Sequence_Number of the packet being sent.
b) Bits 12 – 14 show an integer Revision_Count of the packet.
c) Bit 15 is the Status of the sensor flag (0 – Forwarded Data and 1 – Sensed Data)
d) Bits 16 – 31 contain the Sensor_Data.

16 bits	1 bit	3 bits	12 bits
Sensor_Data	Status	Revision_ Count	Sequence_Number

Figure: 1

(ii) Using shift and add instructions multiply a decimal number X_{10} by 23_{10}. Assume that the result does not exceed the range of a16-bit register. The Intel IA 32 version of this program is required.

(iii) Give the contents of the following registers, along with the run-time stack, when the following instructions are executed. Initially, consider ESP = 00001FF8h.

Note: SOLVE THIS PART HERE. No Marks will be awarded without proper working using the stack diagrams.

```
X1 DWORD 25H
X2 DWORD 27H
MAIN PROC
                PUSH 6H
                PUSH 5H
                CALL P1
    11500000H   MOV RESULT, EAX              ; ESP:      _____________
MAIN ENDP
    P1 PROC
    115000A4H   PUSH EBP
                MOV EBP, ESP                 ; EBP:      _____________
                MOV EAX, [EBP+8]
                ADD EAX, [EBP+12]            ; EAX:      _____________
                PUSH OFFSET X1
                PUSH OFFSET X2               ; ESP:      _____________
                POP ESI
                POP EBX
                ADD [ESI], EAX               ; X2:       _____________
                ADD [EBX], EAX               ; X1:       _____________
                MOV ESP, EBP
                POP EBP
                RET 8                        ; EIP:      _____________
    P1 ENDP
```

(iv) Write an assembly language program to copy the characters of a string to a target string. The characters are stored in such a way that only a single instance of any character in the string is stored. Initialize a source string to: "This is the source string".

(v) Write a recursive procedure to find a value in a large integer array. Ask the user to enter an integer value in the main program. You should pass user supplied value as parameter to the recursive function using the INVOKE directive. Also, draw labeled diagrams to show stack values at each iteration of this recursive function.

(vi) Write an assembly language code to implement the following high-level language code showing the use of LEA instruction and OFFSET assembler directive.

```
char moon [20];
void star_array () {
      char cell[20];
      for (int i=19; i>=0; i--) {
         cell[i] = '*';
         moon[i] = 'x';
         }
}
```

(vii) Write a recursive procedure in x86 assembly language that divides a number by another number and stops when dividend is less than or equal to 5h. Consider dividend = D4A4h and divisor = Ah. The Intel IA 32 version of this program is required.

(viii) Using string primitives, write an assembly language program that searches 20 elements of array ArraySearchValues in 1000 un sorted elements of another array ArrayValues.

(ix) Using string primitives, write a program that converts the string “FAST NATIONAL UNIVERSITY” to its respective ASCII values into a new array. Also, write a procedure to search a particular string SITYA defined in the data directives.