CSCI 2525 *Assignment 3*

Develop two assembly programs implementing two tasks described below.

1. reorderThree.asm

- Sort the following array of three numbers below in ascending order. At the end of running the program, the array will be in ascending order in memory, not just the registers.

  Original Array: MyArray WORD 5767h, 2132h, 4798h

- You may use only MOV and XCHG instructions to accomplish the desired result.
- **DO NOT use any immediate values except in the .data section**.
- Use only direct offset addressing to accomplish the goal. Be as efficient as you can.
- **Note:** You will have to look at memory to see if you have achieved your goal. You may not create other data elements to assist with this problem.

- You will be using a new Irvine Library function, ***DumpMem*** to display the following information. (See page 173 of Irvine's book for more information)

  ESI = Offset of **myArray**
  ECX = # of elements in **myArray**. Use the $ directive in .data to get this information.
  EBX = unit size (good place to use **TYPE** instruction).

  After you have stored the required information in ESI, ECX, and EBX, the program will execute call DumpMem on the next line to display the information.

Sample code for DumpMem:

```
mov esi, xxxxxxx ;Load the offset of the array to ESI
mov ecx, xxxxxxx ;Load # of elements in the array to ECX
mov ebx, xxxxxxx ;Load size of each element into EBX
call DumpMem
call waitMsg
```

2. Fibonacci.asm

Compute and display the output of Fibonacci sequence as follows:

Fibonacci sequence - Wikipedia

a. **Compute** *fib*(*n*) for *n* = 2, 3, …, 9 using an array of the manageable size and data type. You may declare a constant value for fib(0) and fib(1). However, all computation of the remaining elements of the array must be computed by your program.
b. U**sing immediate values is NOT allowed**. You must use the formula shown below (figure 1) to determine the values of the remainder of the required elements. **Do not define a pre-filled array with the pre-computed elements.**
c. After your array is filled with computed values, store *fib*(6) through *fib*(9) in consecutive bytes of the ebx register starting from the lowest byte. i.e. *fib*(6) is stored in the low byte (bl) of ebx, *fib*(7) is stored in the next byte (bh), *fib*(8) is stored in the next byte of ebx and *fib*(9) is stored in the highest byte.

Notes for Fibonacci.asm

1. Assume fib(0)=0, fib(1)=1. These are the only two values you may directly initialize in the .data section. You can declare them as part of the array.
2. You may use any instruction/directive/operator through **chapter 4 pg 132, including any of the arithmetic operators +, *, /, -.**
3. Your program must use indirect operands in some way as discussed in chapter 4. 4. You program MUST calculate all values of the Fibonacci sequence except Fib(0) and Fib (1).
4. You must use a loop. (LOOP instruction)

3. Instructions for Entire Assignment

1. Your program must call DumpRegs as necessary to prove the correctness of your program
2. If you use immediate values for any portion of this assignment (except where specifically allowed), you will receive a zero for that portion of the assignment. Example: mov ebx, 08050302 ;// This is NOT ALLOWED
3. Part of the program will be graded based on program style. We reserve the right to judge style as I deem fit for the assignment. This includes commenting, whitespace, use of the required header, etc.