Processing Arrays and Strings; Using a Link Library

Required Materials:

- Your textbook, *Assembly Language for x86 Processors* (7th edition)
- Removable or network device (Flash drive, memory card, MyMocsNet account, etc.) for storage of your programs
- These instructions
- Intel-compatible, Windows-based personal computer (like the ones in EMCS 306) with text editor, Microsoft Assembler, and Microsoft Visual Studio 2015 (plus DOSBox/CodeView if you elect to program in real-address mode)

Preparation for Laboratory:

Read sections 4.3-4.5 and 5.2-5.4, pages 112-128 and 145-178 of your textbook. Also read through the Programming Exercises in section 4.10, pages 137-138.

Discussion:

Indirect and indexed addressing and program loops, which are all introduced in Chapter 4, are the programming tools we need in order to process tables or arrays of data (including strings, which are arrays of character data). This laboratory exercise will give you an opportunity to practice using these tools to accomplish typical simple programming tasks. You will also be introduced to the process (covered in Chapter 5) of making calls to pre-written routines that are stored in a link library.

Instructions:

Read Programming Exercises 1 through 8 on pages 137-138 of your textbook. You are not required to submit these programs for grading (except as mentioned below), but you should think about what is being asked in each exercise and how you would attempt to do it in a program. You may wish to write code for some of these tasks and trace through it in the debugger in order to see how it works.

Write a program (in either protected or real mode; your choice) similar to the one described in Exercise 5 on page 138, that generates the Fibonacci series of integers. Make your series start with the values 1 and 1; then let each successive number be the sum of the two previous numbers. Modify the original directions as follows: generate the first **thirty-two** (rather than seven) numbers in the series; also define an uninitialized array of 32 doublewords in memory and fill it with the 32-bit values in the series as you compute them. **Call Irvine's DumpRegs procedure after each value is generated** so the registers will be displayed to the screen; in addition, after the last number is generated, **call Irvine's DumpMem procedure to display the final contents of the thirty-two doubleword values in your array**. (See pages 159-160 for descriptions of these two procedures and how to call them.)

Use MAKE16, MAKE32, or a Visual Studio project as appropriate to assemble your program and link its object code to the Irvine library procedures. Run your program and test it to make sure it generates correct output (hint: counting 1 and 1 as the first and second values, the 32nd value in the series is 2,178,309 decimal or 00213D05h). Capture and print out a **screen shot(s) showing the results** of the program (you may have to scroll up/down to see all the output).

Next, write a program like the one described in Exercise 7 on page 138. Make the source string the phrase "I am CPEN 3710 student John Doe.",0 (note: replace the 'John Doe' portion with your own name) and reserve sufficient space for the target string, which will be the original string written backwards. (The "null byte" containing zero at the end of the string is not part of the reversal; it should still be at the end, not the beginning, of the reversed string.) After you have finished reverse-copying the string, **in addition to calling DumpMem** to display the area of memory containing the two strings, **also call Irvine's WriteString procedure (see page 169) twice, to display the original string and the reversed string** to the screen as characters. Assemble and link your program as usual, run it, and capture and print out a **screen shot showing the displayed results**.

To Hand In: *(due no later than 3:00 p.m. Tuesday, October 2)*

Turn in a copy of ***both .LST printouts*** and the ***screen shots showing the results of each of the two programs***. *Staple the program listings and results together neatly* and submit them by the date and time indicated above. ***Make sure your name, the date, and the course number and section are shown in comments at the top of each program listing,*** **and of course make sure each listing is thoroughly commented** to explain its operation as discussed in the "Program Style and Grading Guidelines" handout (available under the Lab Assignments tab in UTC Learn).