Sorting Arrays of Numbers

Required Materials:

- Your textbook, *Assembly Language for x86 Processors* (7th edition)
- Removable or network device (Flash drive, memory card, MyMocsNet account mapped to a drive letter, etc.) for storage of your programs
- These instructions
- Intel-compatible, Windows-based personal computer (like the ones in EMCS 306) with text editor, MASM, and Microsoft Visual Studio or CodeView (if needed for debugging)

Preparation for Laboratory:

Review the material on conditional processing in Chapter 6 of your textbook. Also review Irvine's I/O procedures, the concepts and techniques of calling and returning from procedures, and flowcharting techniques (covered in Chapter 5). You may also need to review the various techniques for sorting arrays of data (which should have been covered in CPSC 1110). Develop a complete flowchart for the program assigned below, at least one week before the lab exercise is due; after your flowchart has been approved, develop and test your code.

Instructions:

Write an Intel x86 assembly language program which will **sort** arrays of **unsigned doubleword** (32-bit) **integers** into **ascending order** and output the **largest** (most positive) element of each array to the standard output device.

Your program must make use of a sorting **procedure** that does the following:

> Sort (in its place in memory, without making a copy elsewhere in memory) an array of ***n*** **unsigned doubleword integers** starting at address ***m*** into **ascending numerical order**. The smallest (closest to 0) number must be put into the first location in the array, the next smallest in the second location, and so on ... until the largest (most positive) number is placed in the last location. The **number of array elements *n*** and **array starting address *m*** are the two (and *only* two) arguments that must be **passed** to the procedure **in registers**. (No other information may be passed into or used by the procedure, nor may the procedure refer by name to the array being sorted.) The **value** of the **largest** (last) element in the sorted array must be **returned** to the calling program **in a register**. Your procedure may use any sorting algorithm (insertion sort, bubble sort, quick sort, etc.) to perform its task, but may not use any information other than the two values specified above to be passed from the calling program.

Your program must use assembler directives to **define two arrays** (one of 8 elements and one of 15 elements) and **initialize them to contain the following data** (given here in order from first to last, in hexadecimal): Array 1 initial contents: 0C0D12AF, 00030256, FFAABBCC, 0F700F70, 00000000, E222111F, ABCDEF01, 01234567. Array 2 initial contents: 61A80000, 024F4A37, EC010203, FAEEDDCC, 2C030175, 84728371, 63AA5678, CD454443, 22222222, 61B1C2D3, 7A4E96C2, 81002346, FDB2726E, 65432100, FFFFFFFF.

The sorting procedure must be **called** from your main program **twice**. The first time, it should be used to sort only the first array of integers; the second time, it should sort only the second array. After each time the procedure is called, the main program must output to the screen the message "The largest unsigned value in the array is: " followed by the value just returned to main by the procedure, displayed in **hexadecimal** format.

As an example, if the program had sorted only the first three numbers in the first array, it would display the message:

```
The largest unsigned value in the array is: FFAABBCC
```

since FFAABBCC is greater than 00030256 and 0C0D12AF.

In addition to the "largest value" message just mentioned, your program must also display the entire region of memory containing **both** arrays, as hexadecimal doubleword values, **three times**: once before either array has been sorted, again after the first array (but not the second) has been sorted, and finally after both arrays have been sorted. (This is so I can verify that each call to the sorting procedure sorts all elements of the intended array, but nothing else.) This requirement can most readily be fulfilled by calling Irvine's DumpMem procedure at the appropriate places in your program.

When your program is working, have me verify its operation and sign in the space below.

To Hand In: *(final submission due no later than 3:00 p.m. Tuesday, November 6)*

1. **Submit a complete, detailed, neat *flowchart*** of your program (using standard flowchart symbols) illustrating the algorithm implemented in the program. If you need more information on flowcharting, you can look up all the standard flowchart symbols in the library or on the Internet. Microsoft's web site has a fairly good introductory flowcharting tutorial (based on Visio) at: https://support.office.com/en-US/article/Create-a-basic-flowchart-f8e57ca2-0c24-4760-bc2e-8812d7310c6a

   I recommend using a computer program such as Visio, EDraw Flowchart, or SmartDraw to create your flowchart. (A 30-day trial version of SmartDraw can be downloaded from their web site at http://www.smartdraw.com/downloads/.) If you do not use a computer drawing program, you will need to use a ruler and a flowcharting template; drawing the chart freehand is not acceptable. **Please create the flowchart on two separate pages, one for the main procedure and one for the sorting procedure.** *The flowchart must be* ***completed and approved by me no later than 3:00 p.m. on Tuesday, October 30 in order for you to receive full credit for this lab exercise.*** **I encourage you to bring your flowchart to me early, as I may not approve it the first time you show it to me.** (I must be convinced that it is of sufficient detail and correctness to facilitate coding before I will give an approval signature.) **If you make changes to your algorithm after I approve your flowchart, you will need to submit *both* the original and modified versions of the flowchart with your lab report.**

2. Turn in a **printed copy** of your *thoroughly commented* program listing (.LST file). **Be sure to follow the documentation instructions given in the "programming style and grading guidelines" handout.**

3. Submit the ***results* of your program** as follows: Run it from the command prompt and capture a "screen shot(s)" showing all the program's output; paste this into a word processor document, print it, and include it with your submitted materials.

4. **Submit *this signed sheet*** verifying that I have approved your flowchart and seen you demonstrate your working program.

Student's Name: _____

Instructor's signature for flowchart approval: _____
(due before 3:00 p.m. 10/30/2018)

Instructor's signature for demonstration of working program: _____
(due before 3:00 p.m. 11/6/2018)

Staple a cover page, the flowchart, your program listing, results, and this signed sheet together and turn them in as your lab report by the date and time specified above. *Late submissions will be penalized severely if they are accepted at all!* Since this is a two-week lab exercise, it will be weighted twice as much as any of the previous exercises (in other words, your score will be entered for both lab 7 and lab 8).