Intro to X86 Assembly – Part II

This assignment continues our gentle introduction to the x86 assembly language programming on a 64-bit Linux platform.

On the Sakai assignment page for this assignment, you will find a file named assign2.asm. This code implements the bubble sort discussed in class. For this assignment, you will copy this file onto a 64-bit Linux system, spend as much time as you feel necessary familiarizing yourself with the code, then modify the code as detailed below.

More information about bubble sort may be found here:
https://en.wikipedia.org/wiki/Bubble_sort

Once you are comfortable with the operation of the sorting program, modify the code in the following manner.

1. Change the algorithm to sort in *descending* order.
2. Add logic to minimize the number of outer loop passes required to sort the data. During the sort tally the number of passes into a 4-byte global (.data or .bss) variable named exactly '**passes**'. Each iteration of the outer loop constitutes one pass.
3. Add logic to count the total number of swaps performed, tallying the value into 4-byte global (.data or .bss) variable named exactly '**swaps'**. Make sure that you minimize the number of swaps performed by avoiding swaps when two values are equal.
4. Add a 256 byte global array named exactly '**output**' in a .bss section. Once your array is sorted, you must copy the sorted 'array' into 'output'
5. Your program should be built as a 64 bit x86 executable for Linux (utilize a virtual machine if necessary).
6. Your program **must** return the contents of the 'passes' variable as its exit status code.

Deliverables

1. Upload your source code to the assignments area on Sakai. Your code must be contained in a single file named exactly **assign2.asm**
2. Your source file must contain, at a minimum, the following information in the form of comments at the beginning of the file:
   a) Your name
   b) The date that you are submitting your program.
   c) Class and assignment number
   d) The command line used to assemble your program into a 64 bit Linux object file
   e) The command line used to link your program into a 64 bit Linux executable file
   f) Any details, such as command line arguments that a user needs to know in order to run your program.

Notes

1. This program does not perform any input using stdin or stdout. In order to observe its behavior you will need to utilize GDB.