Lab 2 Notes and Comments –

Part 1)

Implemented the sort re-using the code from lab 1 part 1 (to find the largest value in a list) and swapping using registers to reorder as needed. This was not likely the most efficient sorting algorithm, but was easy to work with. Notwithstanding, still can't "think" in assembly, so was writing it out in pseudocode, and translating. Took longer than it should have, but works.

Part 2)

Modified code from part one to operate as a subroutine. Looking online for info about push/pop stack operations, as the handbook is less than helpful, found out about macros (cited in the code) which made things a lot easier.

Part 3)

Lazily, I'll admit, change the code from part 2 so that what had been stored in registers to pass to the subroutine, was pushed to the stack, and then popped into the same registers that were previously used. Almost no code was changed as a result, and the application performs as expected.

Part 4)

Had trouble in this one being able to keep track of my preserved registers and return address, but was eventually able to figure it out. The algorithm is simple and easy to implement. Only had issues because I had trouble visualizing the recursive calls and the contents of the stack. Once I recalled that I was able to view the contents of the stack, was able to track down my issues and correct them (was not pushing and popping in the same (reverse) order). Tested code and manually calculate Fibonacci numbers and is correct.