**Introduction**: In this lab, I set up a Raspberry Pi 3 with a Linus OS, and configured the Pi to allow for access across a Wi-Fi network. Finally, from a MAC terminal, I wrote and debugged an assembly program to sort an array of 12 signed bytes, which was validated using various input arrays.

**Design and Testing Methodology**: To initially set up the Rasberry Pi, I installed Rasbian, a Pi-specific Linux OS, to the Pi. Installation was done using Rasberry Pi’s New Out of the Box (NOOBS) software, downloaded onto an 8GB SD card and placed into the Pi. Once Rasbian was installed, I configured the Pi to be able to access Claremont WPA by creating a hash of my network password, and editing a secure network configuration file. I then updated and upgraded the Pi in terminal, and installed the kdbg debugger and sendmail package. Using sendmail package, I configured the Pi’s rc.local file to email its IP address every boot. I then downloaded XQuartz on MAC to access the Pi over a wireless network using a secure shell (SSH) command. Accessing the Pi, I created an assembly .s file to sort an array of 12-signed bytes. For my sorting algorithm, I chose a simple bubble sort, which runs in O(n^2) time. The kdbg debugger was used to check for and resolve errors. For final validation, I use the memory section of the kdbg debugger to find the stored array. The address was stored into register 3 at the start of the program, so I view register 3’s data first and type that value into the memory window. The 12 bytes starting at this address should be the sorted array. Our test cases were an array of all positive bytes, an array of all negative bytes, and an array of positive and negative bytes. The primary obstacle in assembly implementation was understanding the use of a global variable written under the “.data” section of the code. In order to load from and store to this array, we needed to use the load signed byte (LDRSB) and store byte (STRB) commands, rather than the more common LDR and STR commands. Further, through resolving bugs, I discovered, when using STRB and LDRB, I need to index variables from the starting address by the number of elements, rather than the number of bits between elements.

**Results and Discussion**: The sorting algorithm worked as expected and properly sorted all test arrays. The only remaining shortcoming in testing and debugging was the extremely low speed of the kdbg debugger. I later found that the kdbg debugger performs much faster when using the pi directly with a monitor, keyboard, and mouse. Given more time, I would explore using the ddd or gdb debugger, which would make debugging a much faster process when SSHing in. Which that debugging flexibility, I would look to implement more efficient and sorting algorithms like merge sort, which runs in O(nlogn) time.

**Conclusions**: In this lab, I configured a Raspberry Pi with the Rasbian Linux distribution. I further configured the pi to have a network connection to Claremont-WPA, which allowed me to ssh into the Pi via wireless network from my MAC. Finally, I implemented a bubblesort algorithm in assembly language, and validated its functionality in the kdbg debugger using various test cases, all of which were successful.