ECSE 324 - Lab 1 Report - Group 46

Jessica Li (260665795)
Claire Liu (260654285)

Discussion on Approaches to Each Section:

part1.s

The goal of this section is to find the largest integer in a list of numbers given. We approached the task by looping through each element of the list and compare the current element (stored in R1) with the current maximum (stored in R0). If the current element is larger than current maximum, then the current element replaces the current maximum. Otherwise, the code branches back to the loop and start looping through the remaining elements again until the loop counter reaches 0 (meaning all elements have been checked). Finally, the result, which is the maximum number is stored in the memory location of RESULT.

stddev.s

The goal of this section is to compute the standard deviation of a list of numbers given using the simplified standard deviation formula: (maximum number - minimum number)/4. The maximum number of the list is found using the code from *part1.s* and the minimum number is found by modifying *part1.s*' branch condition statement BGE to BLE.

We approached this task by using two registers to store the memory locations of the current minimum and maximum, then we loop through each element in the list and compare it with the current minimum first. If the current element is less than the minimum, then the current element replaces minimum. Otherwise, the code branches to the NEXT label and check if the current number is larger than the maximum. If it is, then the current element replaces the maximum. Otherwise, the code branches back to the loop and start the iteration again. The code terminates until the loop counter reaches 0 (meaning all elements have been checked).

center.s

The goal of this section is to 'center' a list of numbers, in other words, the average of the list should be 0 after centering. The new centered list should be stored in the original memory location that the input list is passed in after the execution of this code.

We approached this task by using two loops. The first loop computes the overall sum of the numbers by looping through all the elements in the list and accumulate them. The sum is then divided by the number of element by arithmetic right shift (ASR) to obtain the average. After the average is computed, the second loop will loop through all elements in the list and subtract the average from each element. Finally, the new, centered list of elements is stored in the corresponding memory location, replacing the original given list.

sort.s

The goal of this section is to sort the elements in the list and rearrange them from smallest to largest. For this section, we implemented a bubble sort algorithm where we created one nested loop consists of two loops in total. Both loops operate N-1 times, where N is the number of elements in the given list. The loops will loop through all elements in the list multiple times, check two neighboring elements and swap them if they are in the wrong order (which is bigger number before smaller number).

Challenges Encountered:

Not many challenges were encountered during this lab, since all of the tasks are relatively basic. At the beginning of the lab, we had difficulties differentiating memory locations and registers, but these problems were solved as we went through the sample code. In addition, we experienced some issues with updating the pointers when sorting the numbers and these problems were solved by using breakpoints and the debugger. In the end, we were able to update the pointers correctly.

Improvements:

1. Use quicksort or mergesort for faster sorting. Quicksort and mergesort has O(logN), while bubble sort has O(N^2).