ECSE 324

LAB REPORT 1

INTESARUL KHAN – 260579991
SHIRAAZ DILAWAR - 260577486

Finding the Max:

The first part of Lab 1 is a simple program that finds the maximum value in an array. This is implemented by first storing the array size and the first element in the array in 2 separate registers. Using a loop, we then sweep through the entire list and update the max. The array length is used as a counter and is decremented for every iteration performed. This allows us to exit the loop we are done with the list. The max is found by comparing the first element (initial max) to the next element. If the next element is greater, then that is stored as the new max in the same register where value1 was stored. The new max is now compared with the next element in the list. This loop continues until the loop counter hits zero. We now exit the loop with the max value stored in the register. The final step is to store this value from the register to the assigned memory location. There were no challenges faced during this part of the lab as the machine code for this was already provided to us.

Standard Déviation(s.d):

This program finds the standard deviation of an array by finding out the maximum and the minimum number, subtracting the two numbers and then dividing the result by 4. We reused the algorithm from part 1 of the Lab to find the maximum number and made a small modification which uses BLE instead of BGE to find the minimum number and stored the value in two registers. Afterwards we computed the result of max-min and stored that value in another register. Given that there are no divide functions available to us, we had to compute the final result by right shifting the result of the subtraction by two.

Centering an array:

An array is centered if its average is zero. This can be implemented by subtracting the average value of the array by each element in the array. This is implemented by first iterating over the entire array and holding the max value in a register. The average can be found by dividing the sum by the number of elements in the array. Since array signal lengths are assumed to be a power of 2, this can be easily done by shifting the sum value right by x, where 2^x = # of elements (ASR to handle -ve lengths). This number is found by loading the size of the array in a register and using a counter to count the number of times division by 2 is required for the quotient to hit 1. We now have our x. Simply shift right (ASR) the Sum calculated by the number x found before to get the average value. Store this value in a register and subtract it from each array element to center the signal, using a loop. Finally store the values in their assigned memory location.

Sorting an array:

The purpose of this program was sorting an array of size N in ascending order. We used a simple bubble sort algorithm of time $O(n^2)$. For this algorithm we have two loops, the inner and outer loop. In the inner loop, the i^{th} element is compared to the (i+1) element. If the i^{th} element greater we swap the two elements in the memory location and then shift the pointers to update i

and i+1. If the i^{th} element is less, we simply update the pointer locations to point to the next elements in the list. This must be done N times to ensure the entire list is sorted, and we keep track of that using the outer loop.

Improvements:

1. Use less registers, would have been more optimized memory usage than.
2. Use more efficient looping, which ties in with our improvement of using less registers.
3. For S.d, Potential improvements would be to check if the length of the array is 0 as the program would not work with an array of size less than 1.
4. For the array program, The program can be improved by taking any signal lengths.
5. Another improvement could be to detect an array size of length 0.
6. For the sorting Algorithm We could have used a more efficient version of bubble sort where the inner loop would not go through all the elements in the array but instead just compare till the (n-j) element with j being the number of times the array has been iterated. We could also have a check in the inner loop to see if the array is already sorted at any point instead of iterating through the list N times.
7. Another potential improvement would be to use one of the divide and conquer algorithms to write a program with a time complexity of O(nlog(n)).

Problems Faced:

As this was the first time we programed in assembly learning the syntax from the start was the biggest challenge we faced. As we are new to assembly we weren't sure initially how to check our answers, so we would just store the values in registers which sometimes gave us incorrect values, so we spent a lot of time trying to fix the algorithm that was already correct.