ECSE 324 Lab 1 Report

Group 34

1. Part One

Part one serves as reminder of what we have learnt in the class. The syntax, how to load and store, how to set conditional flags and how to branch. We were also refreshed on how to use register as a pointer, the difference between memory and resister operands etc.

2.1 Fast Standard Deviation Computation

First, we need to decide the structure of our code, that is, how many zones we might need. To find the maximum and minimum number in the array, we will need to loop through the entire list twice. First, we set up a MAX\_LOOP to find the maximum number. After this is done, we enter MAX\_DONE to store the maximum number from a register to desired memory location. After this, it is very important to reinitialize the pointer towards the first element in the array. And then we use the same approach to find minimum number. After this is done, we enter MIN\_DONE, instead of storing the min number into memory, we leave it in the register as we will load the maximum number into another register to have it subtract the minimum number. And then we implement ‘divide by 4’ by LSR 2 bits. However, the remainder will be dropped as the quotient must be integer.

A potential improvement to this code would be to implement an algorithm that capture the maximum and the minimum value within one for loop instead of two.

2.3 Centering an array

To compute the average, we need to add all elements in the array up and divide it by the number of elements.

Adding elements up is very easy and straight forward. The difficult part lies in how to divide the sum by the number of elements. Thankfully, the signal length is power of two. Which means we only have to LSR the sum according to the length. A signal length of 8 would be represented by 1000 (last 4 digit) in binary. We would need to create an algorithm to take 8 (1000) as input and output 3(number of bits to be shifted). We do this by LSR 8(1000) in a for loop and have the shifted result CMP to #1 , each time we shift, increment the counter by 1. By the time it hits 0001, the counter would be 3 and the CMP it to #1 and would return 0(the Condition flag to break out off the loop). Now we are able to compute the average.

Next, we subtract each element by the average number and store the result back into the same memory address we load the element from. We do this in a for loop until all the original elements have been replaced by the ‘centered’ version of themselves.

Potential improvement would be to use post index mode to have the pointer point towards the next element after each time the pointer has been used instead of manually increment the address that pointer register holds by #4.