EE2016 Midterm Report

Name: Gagan Deep Goru

Roll Number: EE19B023

Note: Compilation instructions are given in the file main.c

Question 1:

In the first part, we are supposed to minimise the number of loads/stores which include push and pop. So, in order to not load an element multiple times, we require 6 registers for storing a_i to a_{i+5}. The registers r0, r1 contain the arguments of the function i.e., r0 contains address of v[0] and r1 contains n (100).

Initially, v[0]-v[5] have been loaded into r2-r7 respectively. S_0 is calculated and stored in r9 and also in r10 for comparison with S_i. The register r8 contains the index of the loop, initialised to 1 (As S_0 is already calculated) and r12 contains the index of minimum S_i initialised to 0. The Loop is started by loading v[6] to r2 since v[0] is not required anymore. Then S_1 is calculated and stored in r10. If $S_1 < S_0$ then the corresponding index 1 and sum S_1 will be stored to r12 and r9 respectively.The index is incremented by 1 and checked if i == n-5. If yes then the loop breaks. Otherwise the next element v[7] is loaded to r3 since v[1] is not required anymore and the process continues.

For this we require 7 additional registers r4-r10, which will be saved to stack and restored later. The loads have been minimised to 1 per iteration as we load an element only once, and use the minimum number of registers so that push/pop will be reduced. Since each new element is loaded to the different register, the loop in itself contains 6 iterations before starting. At each iteration a branch command is used, so that the loop breaks if index value reaches n-5.

In total, we have 7 push and 7 pop instructions, 1 load for each element in the array i.e., 100 load instructions. Overall 100+7+7 = 114 instructions with n clock cycles each (since push, pop have the same cycles as ldr), where n is 1, 5, 10 or 20.

All other instructions are 1 clock cycle each. Before the loop starts, there are 9 such instructions. Once the loop starts, there are 11 of them for calculating S_i and once in every 6 such calculations we have bne Loop instruction. For a 100 element array, we calculate till S_{94} which is 15*6 + 4. So 15 times the entire loop is executed i.e., bne Loop is counted 15 times and the others are counted 94 times. Finally, there is mov r0, r12 to return the index value. Total becomes 9 + 94*11 + 15 + 1 = 1059.

Hence the total number of clock cycles = 114*n + 1059

Table giving the total clock cycles for a 100 element array for different n (number of clock cycles of LDR)

Cycles for LDR	1	5	10	20
Total Clock Cycles	1173	1629	2199	3339

Question 2:

As we are required to minimise the number of instructions, I made use of one register (r2) to load the values v[i-1] and v[i+5] to calculate S_i from S_{i-i} in the loop. The index i is stored in r11 initialised to 1 and index with minimum sum is stored in r12 initialised to 0.

Initially, r2 and r3 are used to calculate S0 and stored in r4. Then S_0 is copied into r3 for comparison later. In the loop, v[i-1] is loaded to r2, then S_{i-1} + v[i-1] is calculated and stored in r4. Then v[i+5] is loaded to r2 and v[i+5] – (S_{i-1} + v[i-1]) is calculated which is S_i, and stored in r4. Then this value is compared with the value in r3. If r4 < r3 (i.e., $S_i < S_{min}$), then the value in r3 is replaced, and also the value in r12 (index with min S_i) is replaced. Then the index value (in r11) is incremented and compared with n-5 (in r1). This process continues till the index becomes equal to n-5.

For this, we only require two registers r4,r11 to be saved to stack and restored. The number of instructions per iteration is reduced to 10 from the previous value of 12, but there are 2 loads per iteration in this case.

Before the loop begins, there are 2 push instructions, 6 load instructions and 9 regular instructions. In the loop, there are 2 load instructions and 8 regular instructions per iteration. For a 100 element array, we are supposed to calculate till S_{94} which means 94 iterations. In the end, there is a mov instruction to return the index value and 2 pop instructions.

In total we have (8*n + 9) + 94*(2*n + 8) + (2*n + 1) = 198*n + 762 clock cycles.

Table giving the total number of clock cycles for a 100 element array for different n (number of clock cycles of LDR)

Cycles of LDR	1	5	10	20
Total Clock Cycles	960	1752	2742	4722

If we compare the total number of clock cycles in Question 1 and Question 2, we can see that the number of clock cycles is significantly less in Question 2 when the load instruction is as fast as the other instructions.

Since this method uses more load instructions per iteration, this can’t be used for Question 1. If we compare the cycles when load instruction is relatively slow, we can see that number of cycles is less in Question 1.