Heather Warman and David Merrick
April 18, 2014
CS 472 HW 2

Comparing the assembly with Optimization level 0 to Optimization level 1: From an initial glance at the two files, the file built with optimization level 1 (105 lines) is significantly smaller than the one with optimization level 0 (78 lines). There are six subroutines in the optimization level 0 file, while there are 9 subroutines defined in the optimization level 1 file. One similarity is that they both make use of the isPalindrome(*char) subroutine. The code with optimization level 1 makes use of more registers at once in several places. In the first line of the isPalindrome() subroutine, it executes push r3, r4, r5, lr (compared to the corresponding push r7, lr line in the optimization level 0 code).

Comparing our assembly code to optimization 0: There are some major differences between the two files. One of which is the use of variable passing in the functions in the optimization level 1 code. We didnt make use of that at all in ours, though it seemed to make the code very functional for optimization. There were also many differences in the available instruction sets since they were written for different ARM versions. Aside from that, the optimized code loaded the string variable in 2 operations, moving the lower 16 bits and then the higher 16 bits. The generated code had a lot of tedious load instructions to move through the data of the string, while our code was much shorter in length. The generated code used 105 lines, and our code only used 76 (and we made use of some extra lines to break up the code to make it more readable). So our code is arguable better memory wise, as it takes up less space. One thing that we found interesting was the amount of registers used for both versions, which were the registers up through r7. When we were writing the code it seemed a bit excessive to use all of them, though it was hard to use any less than that. Another interesting comparison is the number of functions used to achieve the end result. The generated code used 6, and we used 7. Although this is compared to the level 0 optimization, it still seems like a computer would be able to optimize it down quite a bit further than we could, but it was pretty close.

Question 3) For this question we added in a line for part2.s, which is the file used for part 2 to cause the program to have a syntax error. It's currently commented out so that it will compile and run, but it is on line 72: "AREA Data, DATA, READWRITE." When this line is not commented out, it results in the following error at compile time: lab2_part2.axf: Error: L6221E: Execution region ER_RO with Execution range [0x00000000,0x000000a4) overlaps with Execution region ER_RW with Execution range [0x00000000,0x00000004). This is a problem with the memory from both AREAS overlapping. As it turns out, this data area is not necessary for the this function to work, and we can store the string within the code area to avoid this error.