Braxton Elrod
Spencer Parton

Architecture Assignment 1

Observations for the optimization of our assembly code will be discussed in this document. Any reference to optimized code will be located in Architecture/Optimization/Level#/REFERENCED_FILE where '#' in 'Level#' is replaced by the appropriate optimization level (0-2).

The immediate observation was the difference of the length of code between the optimization levels. The 0th level of optimization code was, in all cases, the longest out of all the optimization levels. For our main program we had around 600 lines of assembly code vs about 300 for 1st and 2nd level optimization. Curiously, a common pattern between the 1st and 2nd level optimization was that the 2nd level, ie: the most optimized, was most of the time a few lines longer than the 1st level of optimization. It is our belief that this is due to the fact that the definition of more optimized code does not always mean less lines of code, but rather the code is algorithmically faster at execution.

After observing the difference in the length of code it was then noticed the different way of loading and storing arguments between 0th and 1st level of optimization. At 0th level of optimization the majority of needed data is stored in memory and then called later on when needed. Contrast that with how the 1st and 2nd level of optimization avoid pulling values from memory when possible. What 1st and 2nd level of optimization seems to prefer doing to save time is to keep a value in a register and arithmetically update the value as needed. I believe this is because pulling values from memory takes much longer than moving a value from register to register. Differences between the 1st and 2nd level optimization are more difficult to pick out however. One basic example of the difference between level 1 and level 2 optimization is iPlusEquOne. In level 1 optimization the mov instruction is used to load immediate value of 0 into a register while in the 2nd level of optimization the xor instruction was used. The inference here being that xor is a faster method of "loading" 0 into a register than

loading an immediate value of 0 is. Optimization such as these are the main difference between level 1 and level 2 and are therefore much harder to identify than differences between level 0 and level 1 optimizations.

Additionally some of the optimization process of level 1 and 2 remains a bit of a mystery when observing some of the smaller constructs. For example at the 1st level of optimization for iPlusPlus the objdump will only show that 0 was moved into a register (i = 0) and then it immediately returns with instruction retq. It remains unclear how the assembler is able to convert the 0 into 10 at the end of the program and so some of the optimization processes are obscured.