Compiling for Reduced Register Pressure

Jarrod Richardson
M.A.I
Supervisor: Prof. David Gregg
2017

One of the best-known compiler optimizations is instruction scheduling, which re-orders instructions to better exploit pipelining and instruction-level parallelism. When the compiler re-orders instructions in this way, the result is often an increase in the number of local variables and values that are simultaneously live. This also causes the number of live local values to exceed the number of available registers and the result of this is that some live values have to be spilled to memory.

Despite all the existing compiler work on instruction scheduling, it is not clear that it is actually solving the right problem for popular out-of-order processors that have few architected registers and do instruction scheduling in hardware. Instead, there may be benefits in the compiler trying to re-order instructions to reduce the number of simultaneously live values. This is otherwise known as reducing the register pressure.

For this project, an additional compiler optimization pass was developed to reduce register pressure for a basic block of code. The pass was written for LLVM, a compiler framework library, to test and determine whether reducing register pressure is a viable solution for processors with few architected registers. The paper explores a number of examples that have reduced register pressure solutions and tries to discern where reducing register pressure can improve the overall optimization and efficiency.

In addition to this, the use of randomly generated data dependency graphs were used to comprehensively test the instruction scheduler designed for the optimization pass. These were tested thousands of times to find the average reduced register pressure gained when using the instruction scheduler. It also highlights a number of variables that have an effect on these figures, and how well randomly generated data dependency graphs reflect the real world.