Samuel Larsen
Research Summary

High-level programmers rely on compilers to hide the complexities of modern computer systems. In order to be effective, however, the compiler must provide high performance on the target architecture. My doctoral research focuses on an important aspect of performance optimization: automatic compilation for multimedia extensions. These extensions appear widely in general-purpose and embedded processors, and consist of a set of short-vector operations that apply the same opcode to a vector of operands. Initially, high-level access to multimedia extensions required inline assembly or specialized library calls. This process is tedious, error-prone, and non-portable. Moreover, it places an extra burden on the software developer since it requires manual identification of data parallelism and an in-depth knowledge of the target processor’s instruction set.

I have made two primary contributions to the field of multimedia compilation. The first concerns the overhead of misaligned memory references. In today’s general-purpose architectures, vector memory operations function most efficiently when accessing aligned regions in memory. Compiler techniques that detect or enforce aligned references allow for improved program performance. As a graduate student, I was one of the first to address the alignment issue. One of my accomplishments is the design of a dataflow analysis that extracts alignment information at compile-time. When available, static alignment information allows the compiler to employ more efficient, aligned vector opcodes. I have also developed a loop transformation that enforces aligned references at runtime. The transformation operates in conjunction with a practical profiling method I have developed to eliminate unnecessary code expansion.

My second contribution addresses performance from the perspective of code selection. I have formulated selective vectorization, a technique for balancing computation across a processor's scalar and vector resources. The approach leads to software pipelines with lower initiation intervals, and therefore, higher performance. An important aspect of selective vectorization is its ability to manage operand transfer between scalar and vector instructions. Even when this communication is expensive, the technique is sufficiently sophisticated to achieve large performance gains. In contrast to conventional approaches, selective vectorization operates on a low-level intermediate representation. This method allows the algorithm to accurately measure the performance trade-offs of code selection alternatives on the specific target architecture.