CS 6241 Compiler Design

Instructor: Santosh Pande (santosh@cc.gatech.edu)
Office: Klaus 2338
Office Hrs: TBD

TA: TBD
Email: TBD
TA Office Hrs: TBD
TA Office : TBD

[Course Description | Prerequisites | Course Outline | Homework and Project Information]

Course Description (back to top)

Compilers have always played a key role in the backbone of computing systems. From high performance to tiny systems, compilers optimize code which can fit small memory, which can execute faster on high performance computers, which can reduce power on battery-driven systems, which can be predictable (in real time systems) and which can make systems ubiquitous (such as mobile code). This interesting and important area offers a beautiful blend of science and engineering with a bit of art mixed in it. This course is a first graduate course on compiler optimization techniques that are commonly used at intermediate level and in the backend. The following are the objectives of this course:

- In depth understanding of foundational concepts of optimizing compilers.
- Hands-on experience in building optimizations inside a real-world open source compiler called LLVM.
- Understanding of current research results in one or more areas of optimizing compilers

We will achieve these goals by in-depth discussion of classical and modern compiler analyses and optimizations. The course starts with a discussion of basic dataflow and control flow analysis and building of necessary information at various levels of intermediate forms through analysis. The techniques covered include first bit-vector analysis followed by discussion of SSA form. Special analysis based on SSA form (such as global value numbering) is discussed next. Finally, advanced dataflow problems such as partial redundancy elimination (PRE) are discussed. We then shift our attention to data dependence analysis used in modern high performance compilers and its use in loop transformations for optimizing for memory hierarchy. We will also talk about classical vectorization and SIMDization techniques relevant to modern multi-cores. We will also discuss a bit of code generation optimizations involving advanced register allocation, rematerialization, live range splitting, coalescing and coloring. We will attempt to cover two important emerging areas through papers in the second half of the course : (a) optimizing for embedded

processors, (b) compiler based techniques for multi-cores. Small projects based on a skeletel compiler (LLVM) are given out that develop some optimizations. After successfully completing this course, it is expected that students will be ready to explore research literature on one or more topics in compilers. Similar to other systems courses, this course is intense (not meant for faint hearted in terms of understanding large scale software and programming work) and one should stay on top of things in it.

Prerequisites (back to top)

Students should have taken UG class in compilers (CS 4240 or equivalent) and should have top notch programming skills (C++) and knowledge of data-structures. Others must take permission of the instructor.

Course Outline (back to top)

Textbooks:

- Aho Lam Sethi Ullman "Compilers : Principles, Techniques and Tools" (Dragon Book), 2nd Edition (Required)
- Optimizing Compilers for Modern Architectures: A Dependence-based Approach by Randy Allen (Author), Ken Kennedy (Author), Morgan Kauffman Publishers, ISBN 1558602860 (Supplemental)
- Advanced Compiler Design and Implementation by Steven Muchnik, Academic Press, ISBN 1-55860-320-4 (Supplemental)

- # Outline:

- Review and Code Generation : Organization, Theory and practice behind compiler phases and Code Generation, Intermediate Forms
- Basics of control flow analyses: Basic blocks, data and control flow graphs, loops, reducible and irreducible graphs, regions.
- Data Flow analysis : Local and global data flow problems, iterative solutions and efficient algorithms
- Static Single Assignment form, Sparse conditional constant propagation, global value numbering.
- Data dependence analysis and optimizations: Data dependence tests, Subscripted variables, Necessary and sufficient conditions, Array region analysis, Bounds checking and removal.
- Theory of vectorization, SIMDization vs. Vectorization, Vector and sub-ward

level registers, Data layouts, Sub-word level parallelism,
- Loop restructuring: Simple transformations, loop fission, fusion, reversal, interchange, skewing, linear transformations, tiling
- Optimizing for locality: Instruction cache optimizations, code layout issues, data cache optimizations, general tiling problem with single and multiple references, fission and fusion for data locality.
- Register allocation: register allocation and assignment, local allocators, coloring based allocators, splitting based allocators, introduction to register allocation of for modern processors (speculative loads, register files etc.)
- Code scheduling: Trace scheduling, formation of scheduled regions, software pipelining.
- Emerging topics : Embedded processors : Compact code generation, power efficient code gen, multi-criteria optimizations
- Optional topic (time permitting): Topics in Parallelizing Compilers (Multicores and traditional parallelization techniques)

Emphasis and Grading

- Understand the theory behind different compiler analyses and optimizations.
- Apply the concepts learnt in existing compilers (LLVM)
- Explore new optimizations or develop ability to read papers.
- Homeworks : 23%, Midterm : 20 %, Projects : 32 %, Final : 25 %

Homework and Project Information (back to top)

To be posted on Canvas