Course Review

Tom St. John
Important Dates

May 13 (Tuesday): Quiz
May 15 (Thursday): Project 2, Phase 1 Due
May 22 (Thursday): Project 2, Phase 2 Due

Course work will carry the following weights towards your final grade:

Homework, Lab and Class Participation: 25%
Quiz: 45%
Projects: 30%
Major Topics

- Topic 1: An Overview on Compiler Design
- Topic 2: Compiler Front-End and IR
- Topic 3: Run-time Environment
- Topic 4: Flow Analysis
- Topic 5: SSA Form and Its Applications
- Topic 6: Back-End Optimization
- Topic 7: Loop Scheduling and Software Pipelining
- Topic 8: Parallelism and Locality
Topic 1: Outline of Compiler Design

• Ability to apply knowledge of compiler design flow in understanding overall structure of compiler design and optimization (e.g. phases of a modern compiler, role of each phase)

• An understanding of professional and ethical responsibilities of a compiler engineer
Topic 2: Compiler Front-End and IR

- Ability to apply knowledge of compiler design flow to solve lexical, syntax and semantic analysis problems.
- An ability to use Regular Expressions (RE) and Context-Free Grammar (CFG) design simple lexical analyzer and parser.
- Ability to use high-level intermediate representation to transform source program to a specific intermediate representations, such as syntax trees, AST (Abstract Syntax Trees), DAG, three address code or SSA (Static Single Assignment) form.
- A Knowledge of contemporary issues on this topic.
Topic 3: Run-time Environment

• Understand the software conventions necessary to support various source languages, including data representation, storage allocation for the various storage classes of variables, visibility rules, call sequence, entry, exit, and return.
• Ability to apply the knowledge of run time support systems to trace the program execution.
• Ability to use a modern compiler development platform and tools for the practice of above.
• A Knowledge of contemporary issues on this topic.
Topic 4: Flow Analysis

- Ability to apply knowledge of basic flow analysis to solve dataflow problems (definition of CFG, structure of CFG, domination relation, loops, reducibility, dominance frontier, etc.).
- An ability to identify, formulate and solve dataflow analysis problems in compiler optimization (e.g. DU/UD chains, live variable analysis, etc. and iterative dataflow analysis method).
- Ability to analyze the basic algorithms on the above techniques and conduct experiments to show their effectiveness.
- Ability to use a modern compiler development platform and tools for the practice of above.
- A Knowledge of contemporary issues on this topic.
Topic 5: SSA Form and Its Applications

• Ability to apply knowledge of SSA techniques in compiler optimization (definition of SSA, concept of Φ-nodes, advantages of SSA, where SSA is used, etc.)
• An ability to formulate and solve the basic SSA construction problem based on the techniques introduced in class (e.g. iterative dominance frontiers).
• Ability to analyze the basic algorithms using SSA form to express and formulate dataflow analysis problems (e.g. dead code elimination, common subexp elimination, etc.)
• Ability to formulate the SSA-PRE problem
• A Knowledge of contemporary issues on this topic.

- Ability to apply knowledge of basic code generation techniques, e.g. instruction selection, instruction scheduling, register allocation, to solve code generation problems.
- An ability to identify, formulate and solve instruction scheduling problems (e.g. list scheduling for basic blocks, etc.) and register allocation problems (e.g. graph coloring method for both interference and interval graphs, live range coalescing and splitting, spilling, etc.)
- Ability to analyze the basic algorithms on the above techniques and conduct experiments to show their effectiveness.
- Ability to use a modern compiler development platform and tools for the practice of above.
- A Knowledge of contemporary issues on this topic.
Topic 7: Loop Scheduling and Software Pipelining

• Ability to apply knowledge of basic code generation techniques for loop scheduling and software pipelining techniques to solve code generation problems.
• An ability to identify, formulate and solve loop scheduling problems using software pipelining techniques (e.g. rate-optimal resource constraint software pipelining: RecMII, ResMII, resource constraints, repeated pattern identification, etc.)
• Ability to analyze the basic algorithms on the above techniques and conduct experiments to show their effectiveness.
• Ability to use a modern compiler development platform and tools for the practice of above.
• A Knowledge of contemporary issues on this topic.
Parallelism and Locality

• A basic notion of parallelism and locality
• An understanding of loop nest transformation and optimization