## Symbol Tables

- During which compilation stages is the symbol table accessed?
- What information is stored in the symbol table?
- How can a symbol table be implemented? Compare different structures.
- How is scope information handled? If the same identifier appears in a number of scopes, how do we access the write info? What happens when we exit a scope?

## Intermediate Representations

- What are the criteria for selecting an IR?
- IR classifications
- Linear IRs: quadruples vs. triples
- Graphical IRs: trees, DAGs, CFGS
- Creating DAGs
- Control flow graphs
  - How to identify basic blocks
  - How to identify back edges
  - Building a dominator tree
  - How to identify natural loops.
- SSA form

## Runtime Storage Organization

- Static, stack and heap allocation
- The stack frame model
  - What goes into a stack frame?
  - Do we know its size at compile time?
  - How is the stack managed? Calling sequence.
  - Stack and frame pointers. What are they? Do we need both?
  - What are dynamic links?
  - What are static (access) links and what are Displays? How do they work? Compare.
- Static vs. Dynamic scoping

## Instruction selection, Instruction scheduling

- Basic criteria for instruction selection
- What is instruction scheduling? How do dependencies between instructions determine scheduling decisions? You should be able to come up with an optimal schedule, given a short sequence of instructions.
- How do instruction selection and scheduling interfere with register allocation? In what order should they performed?
- When given a simple machine architecture/instruction set, you should be able to generate some basic code for specific operations (e.g. array accesses, simple loops, if/else statements, etc)

## Optimization

- Principles of optimization.
- You should be able to perform the following optimizations:
  - Constant Folding
  - Local Value Numbering
  - Local and Global Copy Propagation
  - Local and Global Common Subexpression Elimination
  - Loop-Invariant Code Motion
- You should be able to recognize dead code.
- Local Value Numbering vs. Local Common Sybexpression Elimination vs. Local Constant Propagation.
- Constant Folding vs. Constant Propagation
- Principles of Iterative Data Flow Analysis. You should be able to perform the following analyses:
  - Reaching Definitions
  - Live Variables
  - Available Expressions
  - Very Busy Expressions
  - Truly Busy Expressions (not really :))
- You should be able to come up with iterative data flow analysis equations for any given problem.

## Local Register Allocation

- Local vs. Global. Is Local sufficient?
- You should know the tree labeling algorithm discussed on 2/24 and be able to adapt it to different architectures.

## Global Register Allocation

- Criteria in estimating the cost of storing a variable in a register.
- Basic idea behind register allocation via priority-based graph coloring
- Computing priorities
- You should be able to identify live ranges, build an interference graph, apply the coloring algorithm and suggest which live range should be split, if necessary. Read the paper for more information (sections 1-5), especially the section on live ranges.