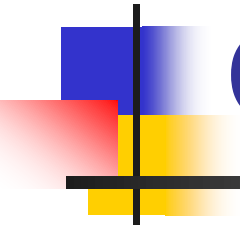
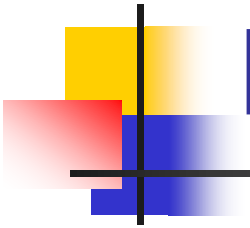


# CSE P 501 – Compilers

Exam Topics

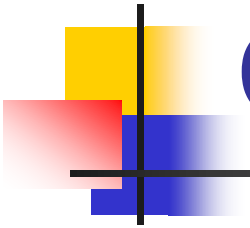
Hal Perkins

Autumn 2009



# Logistics

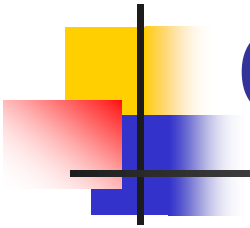
- Exam targeted for 90 min.
- Open textbook(s) – whichever one or two books you use regularly
- Open slides with annotations
- But no additional sources, laptops, pdas, etc.
  - Except that you can use a laptop to browse the slides *only* from the course web



# Content

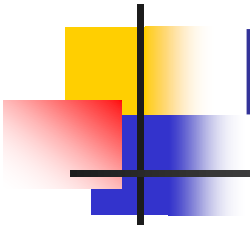
- Most anything covered this quarter, with a bias towards things you've done on assignments or in the project
- General familiarity with other topics we've talked about, but not implementation details

- Note: This is the ppt version of the summary already on the course web – hopefully nothing new or surprising



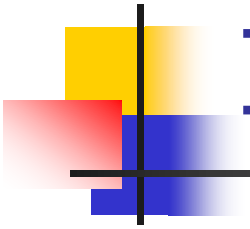
# Grammars

- Regular expressions and DFAs
- Scanners – principle of longest match, describing scanners with a DFA, etc.
- Context-free grammars
  - Derivations (leftmost, rightmost)
  - Ambiguities and how to resolve them
    - Particularly precedence and associativity
  - First, follow, and nullable



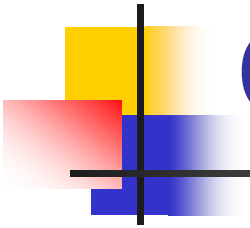
# Parsing

- Basics of LL and LR parsing
- LL parsing and recursive descent
- LR (shift-reduce) parsing
  - Items, constructing LR states and tables
  - Conflicts – significance of shift/reduce and reduce/reduce; how to fix
  - LR(0) vs SLR; lookahead



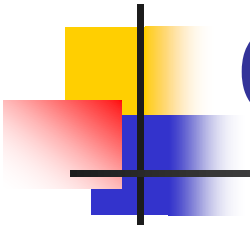
# IRs and Static Semantics

- AST representation of programs
- Symbol tables & type checking
  - What sort of information is collected?
  - What sort of questions can this analysis answer?



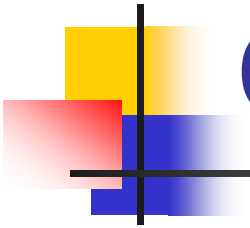
# Code Shape – Particularly x86

- Implementing common language constructs in x86 (assignment, loops, conditionals)
- Procedural programming
  - Function call conventions
  - Stack layout and storage allocation
- Object-oriented programming
  - Object creation and layout
  - Dynamic dispatch and method override
- Be able to read/write simple x86 code



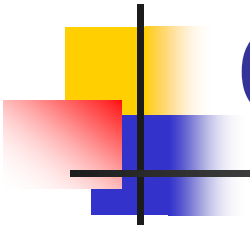
# Optimization Basics (1)

- Control flow graph and basic blocks
- Analysis overview
  - Value numbering
  - Dataflow equations (in, out, use, def)
  - Dominator relationships
  - Core idea behind SSA – be able to show the SSA version of a simple flowgraph with $\Phi$-functions in necessary places
- Be able to set up analysis framework for a problem, or solve simple instances of a problem (e.g., live variables)



# Optimization Basics (2)

- Know some common optimizations and what is needed to allow them
  - Common subexpression elimination, dead code elimination, loop hoisting, induction variables, etc.
  - Conservative analysis – when is a proposed transformation safe?
- Basic ideas about interaction between code and memory hierarchy – loop transformations for better cache utilization



# Overview of Back-End Issues

- Instruction selection
- Instruction scheduling – particularly list scheduling; operation latencies
- Register allocation – interference graphs, graph coloring

- Know the general ideas here; you are not expected to, e.g., be able to implement a graph-coloring allocator in detail