# CSE P 501 – Compilers

Exam Review

Hal Perkins

Autumn 2025

# When/Where

- Wednesday, Dec. 3, 6:30-8:00, here
- Will plan to end @8:00 but if people are still working we can let it run longer, maybe up to 8:20 or so (i.e., want to avoid time pressure)
- Would like to start on time
- Review session: Tue. Dec. 2, 6-7 pm, location TBA. Bring questions!
- Compiler project codegen due the following Mon. Dec. 8, with report due Tue. Dec. 9.

# What

- Info, topic list, and old exams on course web
  - Includes all topics covered this quarter, including hw, project, lectures
    - General details about x86-64 and codeshape, plus basics of backend algorithms included, based on lecture contents, even though the project is not completed yet. See old exams from years when these are included for examples; old 401/m501 exams also have lots of related questions.
- Exam is (almost) closed book
  - Brief reference info and definitions will be supplied on the exam as needed (see old exams for examples)
  - OK to bring two 5x8 index cards with any *hand-written* notes you wish
    - Blank cards available after class today

# Exam topics 1 (from online list)

- Interpreters and compilers - key differences
- Gross structure of compilers - tasks of front/middle/back ends
- Basic notions of grammars - productions, terminals, non-terminals
- Regular expressions and DFAs
  - RE operators (as done in class, not Bash/PERL/Python/…)
  - Constructing REs and DFAs (but you do not need to know the full RE -> NFA -> DFA construction algorithm details)
- Scanners - transforming character streams to token streams

# Exam topics 2 (from online list)

- Context-free grammars
  - Derivations, leftmost, rightmost, etc.
  - Constructing grammars for sets of strings
  - Ambiguity
  - First, follow, and nullable
- LR parsing
  - Shift-reduce parsing
  - Construction of LR(0) and SLR(1) parse tables
    - Items, item sets, and parser states
    - Shift-reduce and reduce-reduce conflicts
    - LR(0) vs. SLR grammars and how lookahead is used in SLR
- LL and recursive-descent parsers
  - Constructing hand-written recursive-descent parsers
  - Grammar problems and solutions - left recursion removal, left-factoring common prefixes, etc.

# Exam topics 3 (from online list)

- Static semantics & symbol tables
  - Kinds of things checked in this phase of a compiler
  - Basic symbol table structures for languages like MiniJava
  - Representation of type information in a compiler

# Exam topics 4 (from online list)

- Basic x86-64 architecture
  - Core instruction set.  Don't memorize details – handouts or ref. info will be provided
- Code shape
  - Representation of common high-level language constructs in x86-64 assembly language
  - Representation of objects and implementation of `new`
  - Implementation of method calls and dynamic dispatch
    - Method tables and overriding
    - Be sure you understand basic Java rules for method overriding and field hiding in extended classes
  - Be able to translate simple C or Java code to x86-64, including calling conventions

# Exam topics 5 (from online list)

- Intermediate representations, particularly:
  - Abstract syntax trees (ASTs)
  - Control-flow graphs, basic blocks
- Analysis and optimization
  - General form of dataflow equations (def, use, in, and out sets) and how these are used to analyze typical problems like liveness; be able to solve simple problems like the ones we saw in class & on HW4
  - Dominators and immediate dominators
  - Basic idea of SSA - what it means; dominance frontiers; be able to hand translate a simple CFG into SSA with appropriate phi functions (do not need to precisely simulate the detailed algorithms for this)
  - Interaction between analysis and optimizations - what can we do with the information that is discovered by the analysis; when is a transformation safe

# Exam topics 6 (from online list)

- Back-end algorithms. You should have general familiarity with
  - Instruction selection
  - Instruction scheduling
  - Register allocation by graph coloring

  at the level presented in lecture, and be able to do simple problems involving these. But test coverage will be general, not as detailed as hw problems like dataflow, SSA, grammars, etc.

# Questions?