CSE P 501 exam checklist Au09

You may bring: Your compiler textbook(s), plus the course slides and notes. No additional books or reference materials, including homework assignments, sample solutions, or old exams.

You may access the course slides using a laptop provided that you only look at material that is linked from http://www.cs.washington.edu/education/courses/csep501/CurrentQtr/lectures/slides.html . No other material on the course web site (including videos) or elsewhere on the web may be used.

- Interpreters and compilers – key differences
- Gross structure of compilers – task of front/middle/back ends
- Basic notions of grammars – productions, terminals, non-terminals
- Regular expressions and DFAs
    - RE operators
    - Constructing REs and DFAs (but you do not need to know the full RE -> NFA -> DFA construction algorithms)
- Scanners – transforming character stream to token stream
- Context-free grammars
    - Derivations, leftmost, rightmost, etc.
    - Constructing grammars for sets of strings
    - Ambiguity
- LR parsing
    - Shift-reduce parsing
    - Construction of LR(0) and SLR(1) parse tables
        - Items
        - Shift-reduce and reduce-reduce conflicts; table differences between LR(0) and SLR(1) due to lookahead
    - First, follow, and nullable
- LL and recursive-descent parsers
    - How to construct a hand-written recursive-descent parser
    - Fixing grammar rules – left recursion, ambiguities like dangling else’s
- Intermediate representations – particularly abstract syntax trees (ASTs)
- Static semantics & symbol tables
    - Typical kinds of conditions that are tested here
    - Basic symbol table structures for languages like MiniJava
- Basic x86 architecture
    - Core 32-bit instruction set – don’t memorize details, but be generally familiar with the basics
    - Be able to write simple C-level functions in x86 assembly language, including, in particular, calling conventions and stack frame layouts.
- Code shape
    - Representation of common high-level language constructs in x86 assembly language

  - Implementation of dynamic dispatch
    - Method tables and overriding
    - Be sure you understand basic Java rules for method overriding and field hiding in extended classes.
  - Representation of objects and implementation of new
- Back-end issues. You should have a general familiarity with the basic ideas discussed in lecture, but are not expected provide detailed implementations.
  - Instruction selection – what are the basic ideas behind tree pattern matching
  - Instruction scheduling – what is list scheduling; what are some of the issues that determine a good instruction schedule (resource contention including registers; operation latencies)
  - Register allocation – what is the role of the interference graph and the ideas behind using graph coloring to allocate registers
- Dataflow analysis and optimization
  - What is the control flow graph, what are basic blocks
  - Dominators and immediate dominators; how to find a loop in a cfg
  - General form of dataflow equations (def, use, in, and out sets) and how these can be used to solve typical problems like liveness analysis; be able to set up or solve simple problems like the liveness one we did in lecture
  - Basic idea of SSA – what it means; be able to hand translate a simple cfg into SSA with appropriate phi functions (you do not need to be able to trace the full algorithms that do this)
  - Interaction between analysis and optimizations – what can we do with the information that is discovered by the analysis; how to the transformations interact with the analysis