



Lecture Notes

Compilers: Principles, Techniques, and Tools

These course lecture notes supplement topics from each textbook chapter:

- Lexical Analysis (Chapter 3)
 - [Intro. to lexical analysis](#) [3]
 - [Implementing a lexical analyzer](#) [3]
 - [Lexical analysis](#) [5]
- Syntax Analysis (Chapter 4)
 - [Specifying languages with regular expressions and context-free grammars](#) [2]
 - [Formal grammars](#) [5]
 - [Top-down parsing](#) [2]
 - [Top-down parsing](#) [3]
 - [Top-down parsing](#) [5]
 - [Bottom-up parsing](#) [3]
 - [Bottom-up parsing](#) [5]
 - [SLR-SR parsing](#) [5]
 - [LALR parsing](#) [5]
 - [Constructing predictive parsers](#) [3]
 - [Parse table construction](#) [2]
 - [Parsing action conflicts](#) [3]
 - [Misc. parsing](#) [5]
- Syntax-Directed Translation (Chapter 5)
 - [Intro. to syntax-directed translation](#) [3]
 - [Syntax-directed definitions](#) [3]
 - [Syntax-directed translation schemes](#) [3]
 - [Syntax directed translation](#) [5]
- Intermediate-Code Generation (Chapter 6)
 - [Intermediate representations](#) [2]

- [Intermediate representations](#) [3]
- [Intermediate representations](#) [5]
- [Semantic analysis](#) [2]
- [Semantic analysis](#) [5]
- [Types in programming languages](#) [3]
- [Arrays, boolean expressions, flow of control](#) [3]
- Run-Time Environments (Chapter 7)
 - [Runtime environments](#) [5]
 - [Run-time storage allocation](#) [3]
 - [Intro. to garbage collection](#) [1]
 - [Incremental and partial garbage collection](#) [1]
- Code Generation (Chapter 8)
 - [Unoptimized code generation](#) [2]
 - [Code generation](#) [3]
 - [Code generation algorithms](#) [3]
 - [Final code generation](#) [5]
 - [Code optimization overview](#) [5]
 - [Optimization of basic blocks](#) [3]
 - [Register allocation via graph coloring with live ranges](#) [1]
 - [Register allocation via graph coloring with webs](#) [2]
- Machine-Independent Optimizations (Chapter 9)
 - [Intro. to program analysis and optimization](#) [2]
 - [Intro. to data flow analysis](#) [2]
 - [Intro. to data flow analysis](#) [1]
 - [Data-flow analysis](#) [4]
 - [Data flow analysis framework](#) [1]
 - [Foundations of data flow analysis](#) [2]
 - [Foundations of data-flow analysis](#) [4]
 - [Data-flow analysis frameworks](#) [4]
 - [Constant propagation, loop detection](#) [1]
 - [Partial redundancy elimination](#) [1]
 - [Flow graphs](#) [4]
 - [Loops in flow graphs](#) [4]
- Instruction-Level Parallelism (Chapter 10)
 - [Intro. to instruction scheduling](#) [2]
 - [Basic block scheduling and \(basic\) global scheduling](#) [1]
 - [Software pipelining](#) [1]
- Optimizing for Parallelism and Locality (Chapter 11)
 - [Intro. to parallelism and locality optimizations](#) [1]
 - [Optimizations with affine transforms](#) [1]
- Interprocedural Analysis (Chapter 12)

- [Pointer analysis with Datalog](#) [1]
- [Interprocedural analysis](#) [4]
- [Applications of interprocedural analysis](#) [4]
- [Datalog](#) [4]
- [Binary decision diagrams for pointer analysis](#) [1]
- [Binary decision diagrams](#) [4]
- Misc.
 - [Dynamic compilation](#) [1]
 - [Processor architectures](#) [5]
 - [Compiler course summary](#) [2]

Sources

- [1] [Stanford CS243](#): Advanced Compiling Techniques, Winter 2008, Prof. Monica Lam
- [2] [MIT 6.035](#): Computer Language Engineering, Fall 2005 (OpenCourseWare), Prof. Saman Amarasinghe and Prof. Martin Rinard
- [3] [Columbia COMS W4115](#): Programming Languages and Translators, Spring 2008, Prof. Alfred Aho
- [4] [Columbia COMS W4117](#): Compilers and Interpreters: Software Verification Tools, Fall 2007, Prof. Alfred Aho
- [5] [Stanford CS143](#): Compilers

[Back to Dragon Book home page](#)

Last modified: 2008-09-15

Page created by Philip J. Guo