CS 152: Syllabus

References are to the text *Compiler Construction Principles and Practice* by Louden:

Overview — *Chapter 1*: Introduction

Lexical Analysis (Scanning) — *Chapter 2*:

1. Sec. 2.1: The scanning process
2. Sec. 2.2: Regular expressions
3. Sec. 2.3: Finite automata (skip NFAs)
4. Sec. 2.4: From regular expressions to DFAs (skip NFAs)
5. Sec. 2.5: Implementation of a scanner
6. Sec. 2.6: Using Lex (will be covered in a lab session)

Context-Free Grammars and Parsing — *Chapter 3*:

1. Sec. 3.1: The parsing process
2. Sec. 3.2: Context-free grammars
3. Sec. 3.3: Parse trees and abstract syntax trees
4. Sec. 3.4: Ambiguity
5. Sec. 3.5: Extended notations

Top-down Parsing — *Chapter 4*:

1. Sec. 4.1: Recursive Descent parsing
2. Sec. 4.2: LL(1) parsing
3. Sec. 4.3: FIRST and FOLLOW sets

Bottom-up Parsing — *Chapter 5*:

1. Sec. 5.1: Overview
2. Sec. 5.2: Finite automata of LR(0) items
3. Sec. 5.3: SLR(1) parsing
4. Sec. 5.4: LR(1) and LALR(1) parsing
5. Sec. 5.5: Using yacc (will be covered n a lab session)

Semantic Analysis — *Chapter 6*:

1. Sec. 6.1: Attributes and Attribute grammars
2. Sec. 6.2: Algorithms for attribute computation
3. Sec. 6.3: The symbol table
4. Sec. 6.4: Type checking

Run-Time Environments — *Chapter 7:*

1. Sec. 7.1: Memory organization during execution
2. Sec. 7.2: Fully static environments
3. Sec. 7.3: Stack-based environments

Code Generation *Chapter 8*:

1. Sec. 8.1: Intermediate code and data structures for code generation
2. Sec. 8.2: Basic code generation techniques
3. Sec. 8.3: Code generation for data structure references
4. Sec. 8.4: Code generation for control statements and logical expressions
5. Sec. 8.5: Code generation for procedure/function calls
6. Sec. 8.9: A survey of code optimization techniques

Compiler Backend *Notes*:

1. Code optimization techniques
2. Machine code generation
3. Register allocation
4. Data flow analysis
