- Intermediate Code Generation
  - Intermediate Representations
  - Symbol Table Design
- Runtime Storage Layout
  - Representation of scalars, arrays, objects
  - Memory areas: static, stack, heap
- Calling Conventions
  - Caller vs. Callee
  - What things must be taken care of on call and return
  - Layout of stack frame
- Parameter Passing Modes
  - Call-by-value
  - Call-by-reference
  - Call-by-name
- Object Representation
  - Data Layout
  - Inheritance
  - Method invocation with inheritance (vtables)
- Code Generation for other Language Constructs
  - if-then-else
  - Loops
  - Arrays - new, length, assignment, access
- Target Code Generation
  - Register Allocation
  - Stack Frame Layout
  - Instruction Selection
- Interpreters
  - Compilers vs. Interpreters
  - Pros and Cons
  - Implementation Needs
- Optimization
  - Scopes of Optimization: peephole, local, global, interprocedural
  - Examples of each
- Runtime Systems
  - What functions can be handled by runtime systems
  - Why needed?
- Miscellaneous (light understanding only)
  - Decompiling, obfuscation
  - Syntax-directed editing
  - Compiling for multicore
  - Attribute grammars