CSc 453
Compilers & Systems Software

Saumya Debray
The University of Arizona
Tucson, AZ 85721

Course Objectives

- Understand the design and implementation of compilers and related systems software.
- Understand how source language programs are implemented at the machine level.
- Understand compilation as an instance of language translation.

Compilers

A *compiler* (more generally, *translator*) maps *source language strings* to "equivalent" *target language strings*. E.g.:

- gcc : C/C++ programs to assembly/machine code
- f2c : Fortran programs to C programs
- latex2html: Latex documents to HTML documents
- javac : Java programs to JVM byte code
- ps2pdf: PostScript files to PDF files

Languages

- *Syntax*:
  - "structural" aspects of program units.
  - specified by a grammar.
- *Semantics:*
  - the "meaning," i.e., behavior, of program units.
  - specified using *actions* associated with grammar rules.

Phases of a Compiler

1. Lexical analysis ("scanning")
   - Reads in program, groups characters into "tokens"
2. Syntax analysis ("parsing")
   - Structures token sequence according to grammar rules of the language.
3. Semantic analysis
   - Checks semantic constraints of the language.
4. Intermediate code generation
   - Translates to "lower level" representation.
5. Program analysis and code optimization
   - Improves code quality.
6. Final code generation.

Grouping of Phases

- *Front end* : machine independent phases
  - Lexical analysis
  - Syntax analysis
  - Semantic analysis
  - Intermediate code generation
  - Some code optimization
- *Back end* : machine dependent phases
  - Final code generation
  - Machine-dependent optimizations

Costs of different phases

- Typically, a compiler spends most of its time doing I/O and lexical analysis:
  - ~ 35-40% of time spent in I/O
  - ~ 30% in lexical analysis
  - ~ 10% in symbol table management
  - ~ 7-15% in parsing and other control