Writing the programs

Professor Hossein Saiedian

EECS 348: Software Engineering

Spring 2023

Coding conventions are essential

- Standards or coding conventions
  - For you, for testers, for maintainers
  - Documentation
  - Matching design with implementation
    * Low coupling, high cohesion, well-defined interfaces

Write self-describing programs (modules)

- Descriptive identifiers (variables, functions, constants, …)
- Prelude comments
  - Precisely but concisely describe what a unit does
  - Include pre- and post-conditions as comments or assertions
  - Clearly describe the expected parameters
- Appropriate and consistent indentation, line alignment, and use of blank lines to show the relationship between blocks of code
  - Control structures (if-then-else, loops) are especially important

Algorithms

- Balance: efficiency vs maintainability
- Efficiency may have hidden costs
  - Cost to write the code faster
  - Cost to test the code
  - Cost to understand the code
  - Cost to modify the code

Data structures

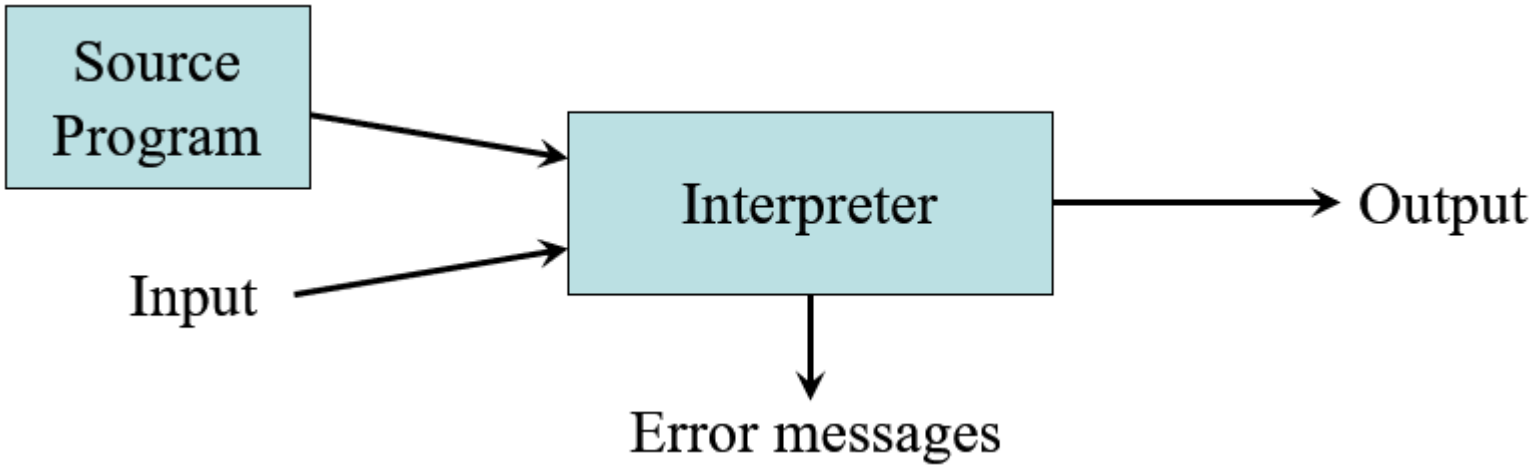
- Carefully design data structures
- All programs manipulate data
  - Read, process, store, display
  - Data can be numbers, characters, images, audio
- Data structures choices influence a program at every level
  - Improve ability to solve problems abstractly
    - Data structures are the building blocks
  - Execution speed
  - Memory requirements
  - Maintenance (debugging, extending, etc.)
  - Improve your ability to analyze your algorithms
- Goal: simple, elegant code
  - Gauge (and improve) time complexity

Documentation

- Internal
  - Programmers put comments in their program's code to help themselves and others understand the code later
  - Document whys
- External
  - Describe the program (module), data
  - Catalog keywords for future reuse
- Documentation should continue after the code has been completed

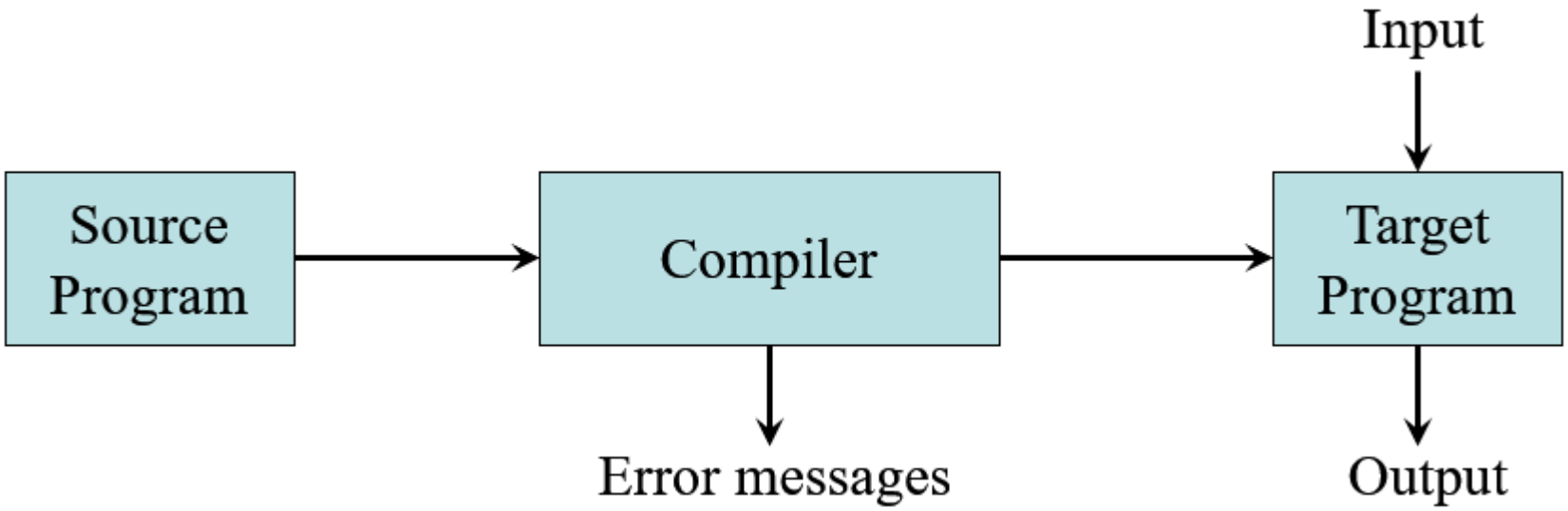
Compiled programs vs interpreters

- Interpretation
  - Performing the operations described by the source program
  - An extremely simplistic view

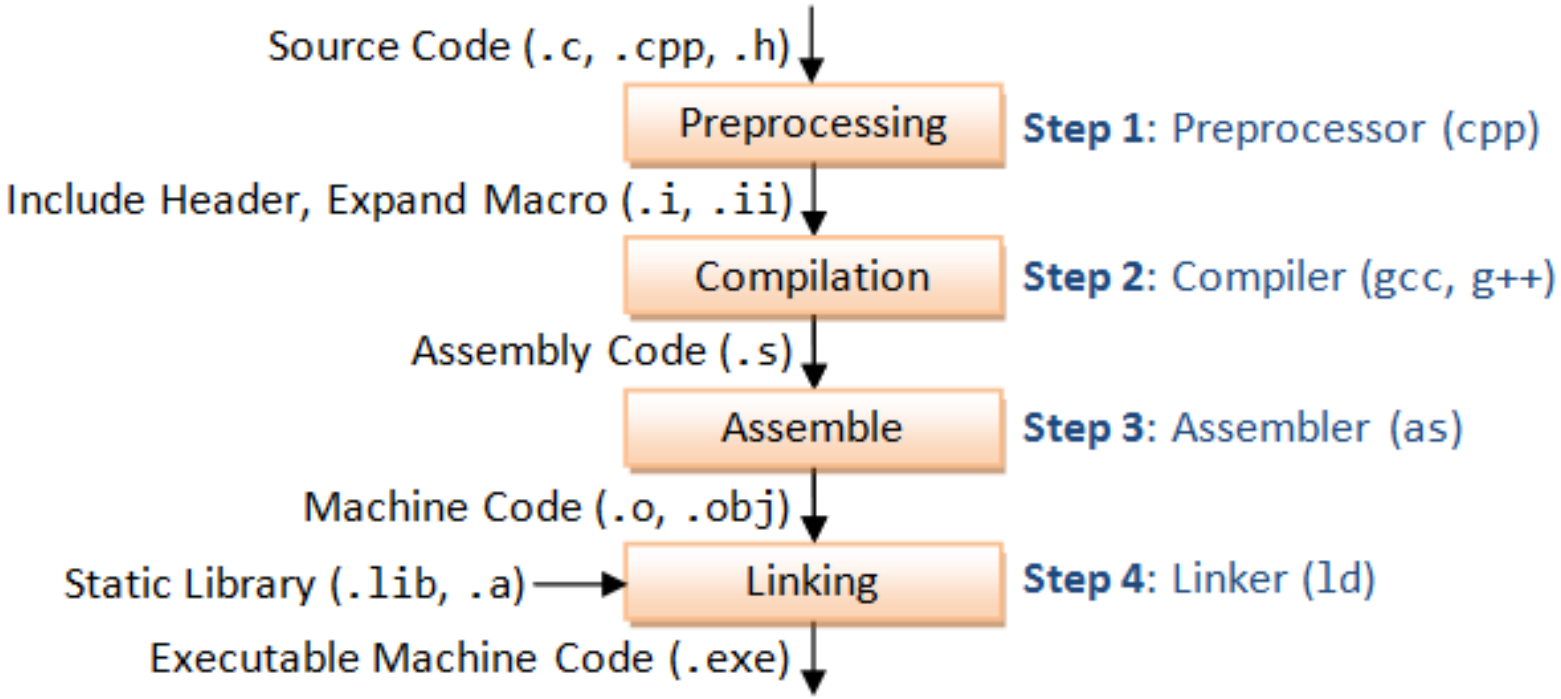


Compiled programs vs interpreters

- Compilation
  - Translation of a program written in a source language into a semantically equivalent program written in a target language
  - An extremely simplistic view



Compilation: A simplified view



The compilation process has many parts

- Programmers write the code
- Preprocessor (code may include directives like #include)
- Scanner (identifies tokens like =, +=, identifiers, …)
- Parser (performs syntax analysis based on the grammar)
- Semantic analyzer (type checking, etc)
- Intermediate code generator (assembler)
- Optimizer
- Code generator (object code)
- Linking
- Executable