CS335: An Overview of Compilation

Swarnendu Biswas

Semester 2019-2020-II

CSE, IIT Kanpur

Content influenced by many excellent references, see References slide for acknowledgements.

A Bit of History

- In the early 1950s, most programming was with assembly language
  - Low programmer productivity
  - Cost of software development far exceeded cost of hardware
- In 1954, John Backus proposed a program that translated high level expressions into native machine code for IBM 704 mainframe
- Fortran I project (1954-1957): The first compiler was released

Impact of Fortran

- Fortran I compiler was the first optimizing compiler
  - Programmers were initially reluctant to use a high-level programming language for fear of lack of performance

- The Fortran compiler has had a huge impact on the field of programming languages and computer science
  - Many advances in compilers were motivated by the need to generate efficient Fortran code
  - Modern compilers preserve the basic structure of the Fortran I compiler!

Executing Programs

- Programming languages are an abstraction for describing computations
  - For e.g., control flow constructs and data abstraction
  - Advantages of high-level programming language abstractions
    - Improved productivity, fast prototyping, improved readability, maintainability, and debugging
- The abstraction needs to be transferred to machine-executable form to be executed

What is a Compiler?

- A compiler is a system software that **translates** a program in a source language to an **equivalent** program in a target language

- Typical “source” languages might be C, C++, or Java
- The “target” language is usually the instruction set of some processor

Important Features of a Compiler

- In addition to translation, compilers provide feedback to the user
  - Point out errors and potential mistakes in the program

Source-Source Translators

- Produce a target program in another programming language rather than the assembly language of some computer
- The output program require further translation before they can be executed
- Many research compilers produce C programs

More Examples of a Compiler

- A typesetting program that produces PostScript can be considered a compiler
  - Typesetting LaTeX to generate PDF is compilation

Interpreter

- An interpreter takes as input an executable specification and produces as output the result of executing the specification

- Scripting languages are often interpreted
  - For e.g., Perl, Python, and Bash

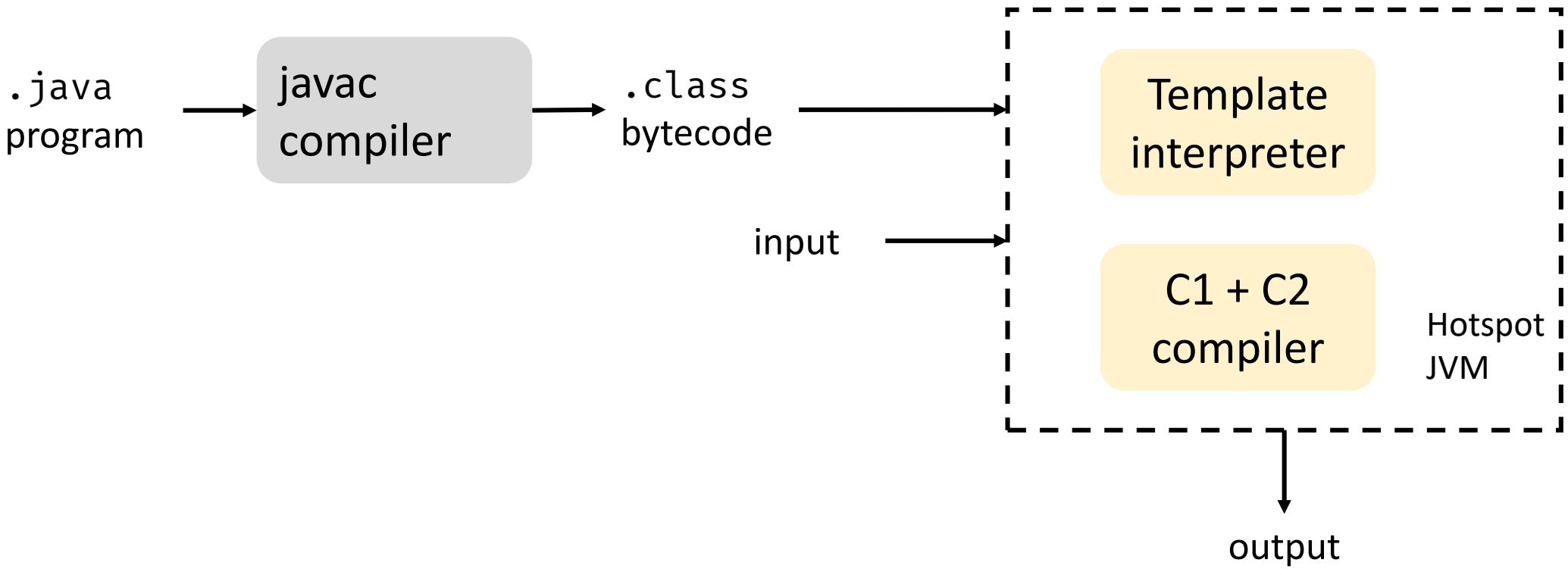
Compilers vs Interpreters

Compilers	Interpreters
• Translates the whole program at once	• Executes the program one line at a time • Compilation and execution happens at the same time
• Memory requirement during compilation is more	• Memory requirement is less, since there is less state to maintain
• Error reports are congregated	• Error reports are per line
• On an error, compilers try to fix the error and proceed past	• Stops translation on an error
• Examples: C, C++, and Java	• Examples: Python, Ruby, PHP

Hybrid Translation Schemes

- Translation process for a few languages include both compilation and interpretation (e.g., Lisp)
- Java is compiled from source code into a form called bytecode (`.class` files)
- Java virtual machines (JVMs) start execution by interpreting the bytecode
- JVMs usually also include a just-in-time compiler that compiles frequently-used bytecode sequences into native code
  - JIT compilation happens at runtime

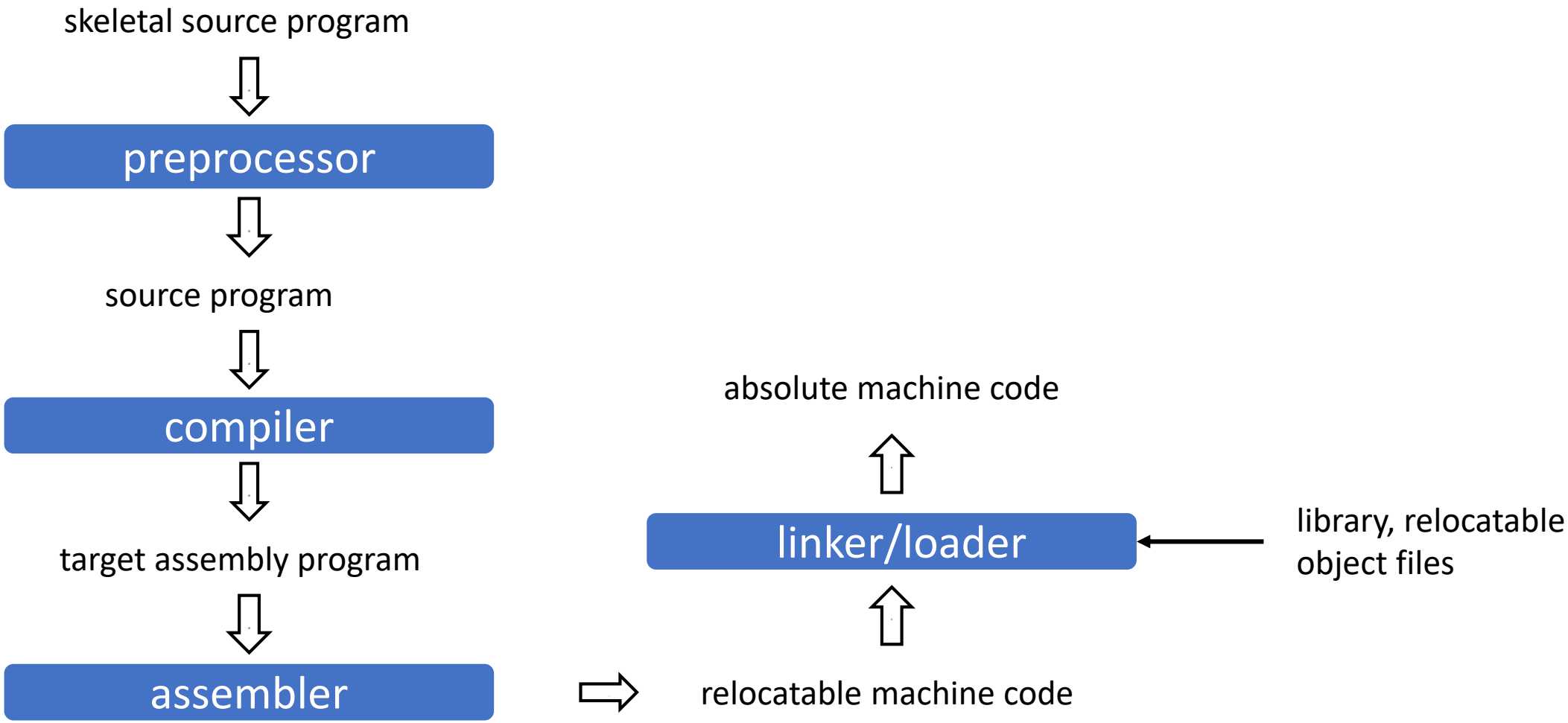
Compilation Flow in Java with Hotspot JVM



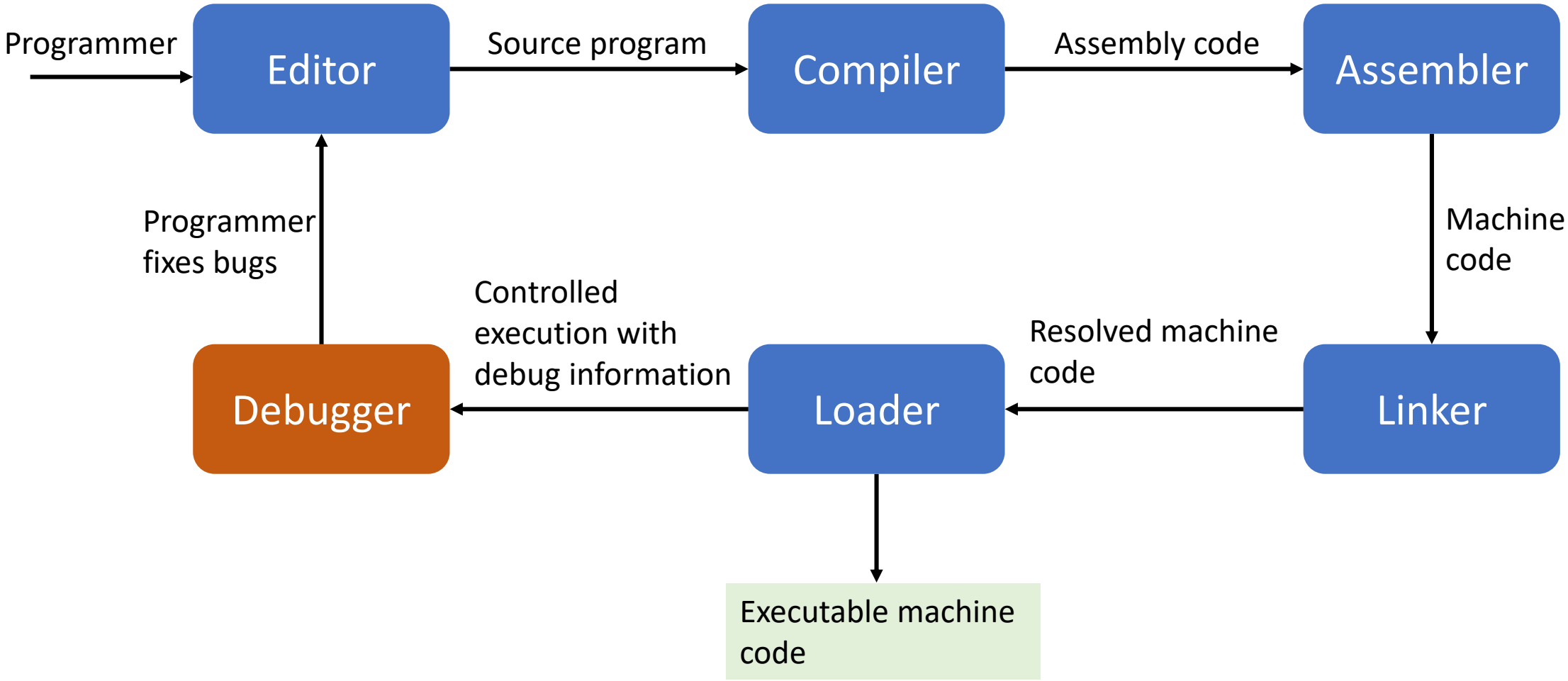
Language Processing

- Language processing is an important component of programming
- A large number of systems software and application programs require structured input
  - Command line interface in Operating Systems
  - Query language processing in Databases
  - Type setting systems like Latex

A Language-Processing System



Development Toolchain



Goals of a Compiler

- A compiler must preserve the meaning of the program being compiled
  - Proving a compiler correct is a challenging problem and an active area of research
- A compiler must improve the input program in some discernible way
- Compilation time and space required must be reasonable
- The engineering effort in building a compiler should be manageable

Applications of a Compiler

```
DO I = 1, N
  DO J = 1, M
    A(I,J+1) = A(I,J) + B
  ENDDO
ENDDO
```

Applications of a Compiler

- Perform loop transformations to help with parallelization

```
DO I = 1, N
  DO J = 1, M
    A(I,J+1) = A(I,J) + B
  ENDDO
ENDDO
```

```
DO J = 1, M
  DO I = 1, N
    A(I,J+1) = A(I,J) + B
  ENDDO
ENDDO
```