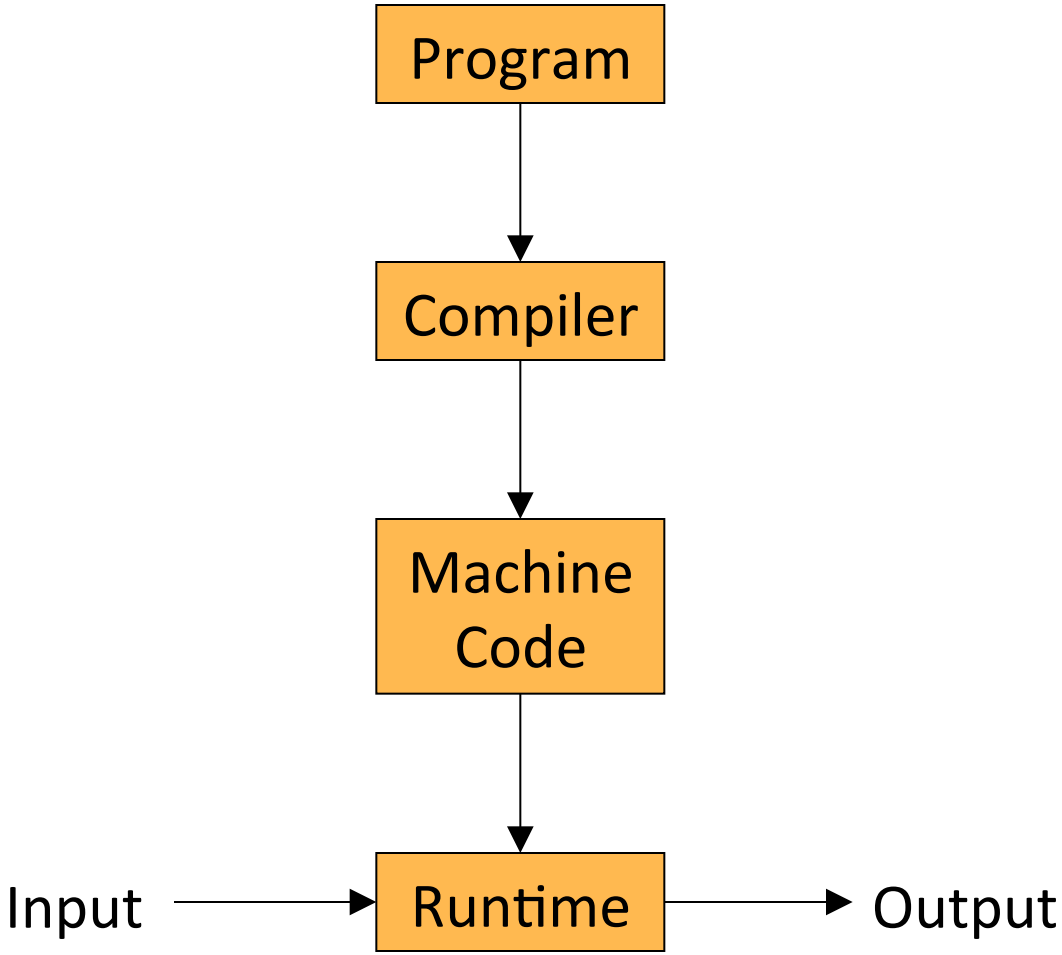
IN3: What is a Compiler

Introduction to Compilers

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

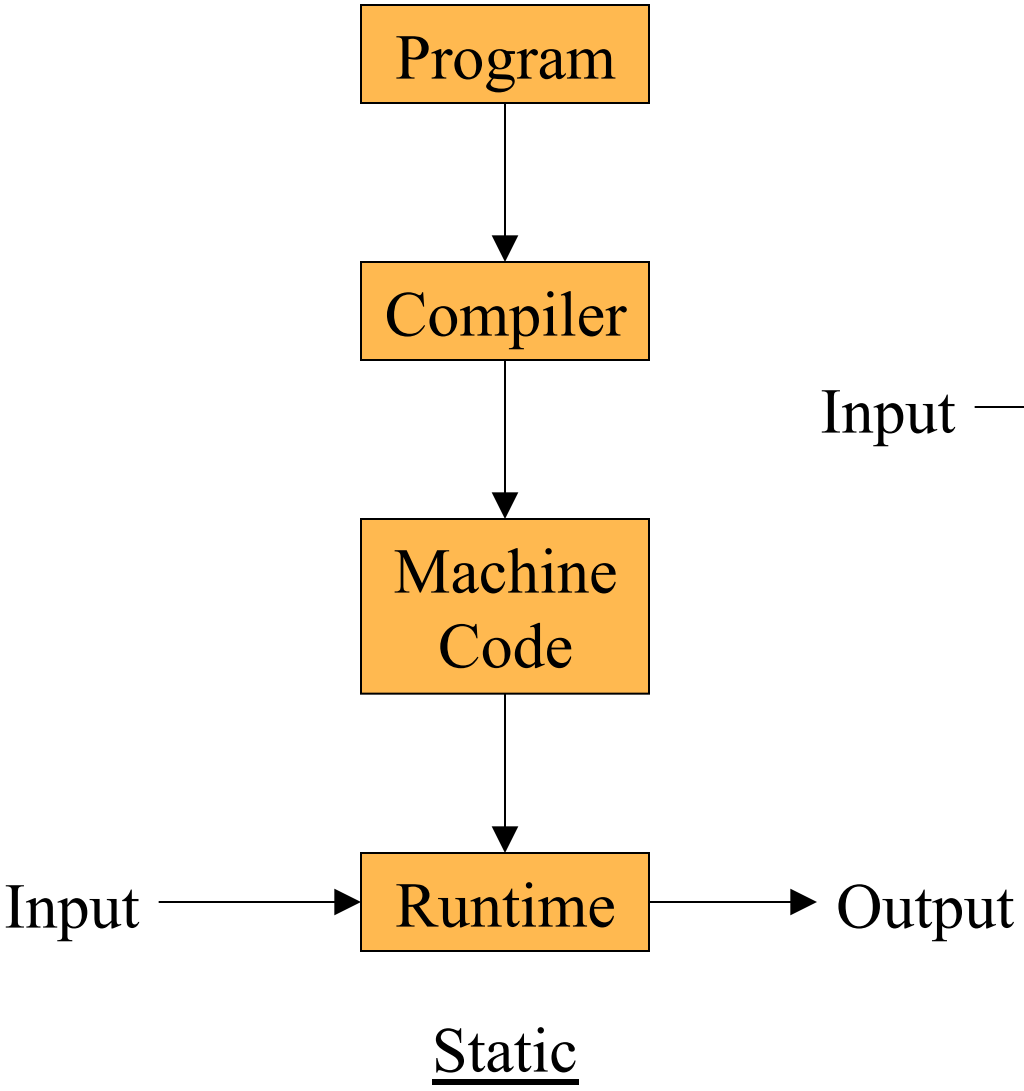

Program
Compiler
Machine Code
Input
Runtime
Output

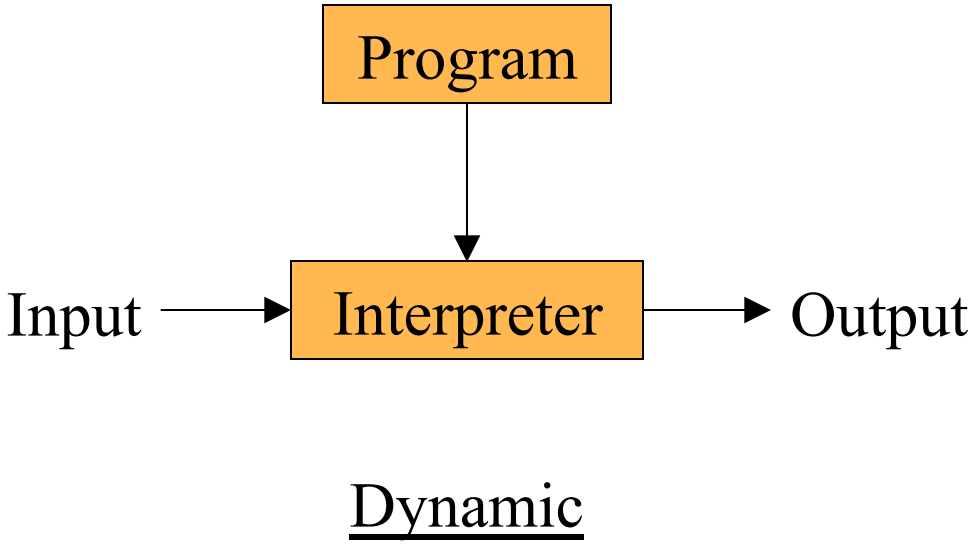
Compilers

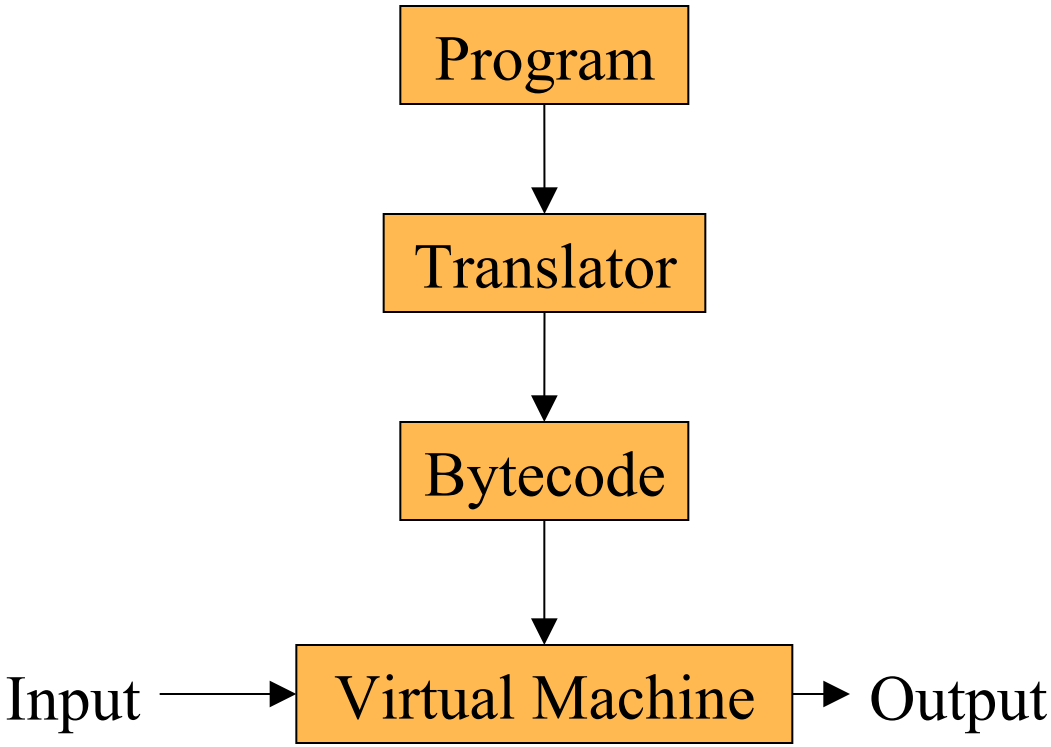
- Analysis of the source (front-end)
- Synthesis of the target (back-end)
- The *translation* from user **intention** into intended **meaning**
- The requirements from a Compiler and a Programming Language are:
  - Ease of use (high-level programming)
  - Speed

Cousins of the compiler

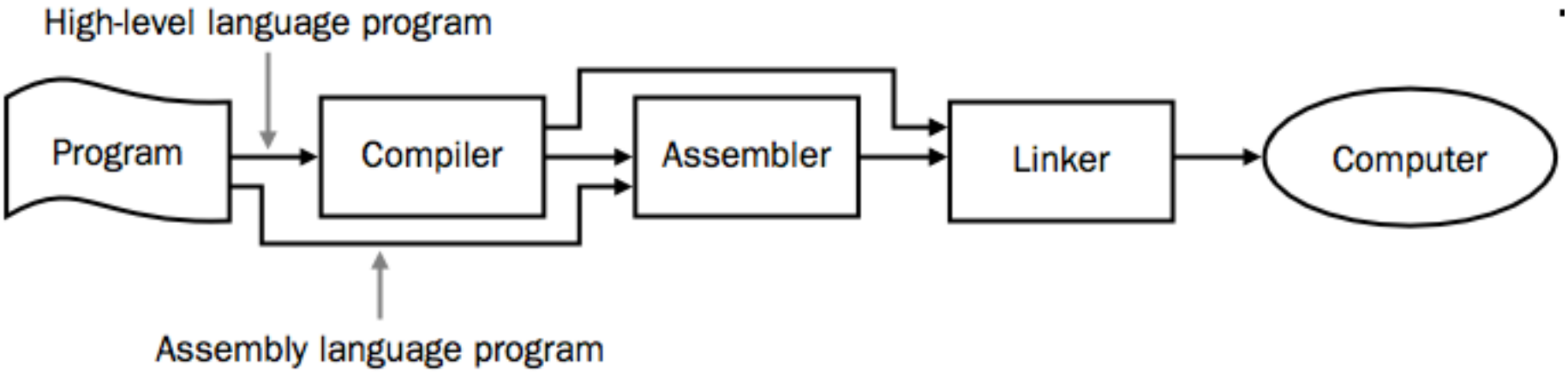
- "Smart" editors for structured languages
  - static checkers; pretty printers
- Structured or semi-structured data
  - Trees as data: s-expressions; XML
  - query languages for databases: SQL
- Interpreters (for PLs like lisp or scheme)
  - Scripting languages: perl, python, tcl/tk
  - Special scripting languages for applications
  - "Little" languages: awk, eqn, troff, TeX
- Compiling to Bytecode (virtual machines)


Program
Compiler
Machine Code
Input
Runtime
Output
Static


Program
Input
Interpreter
Output
Dynamic

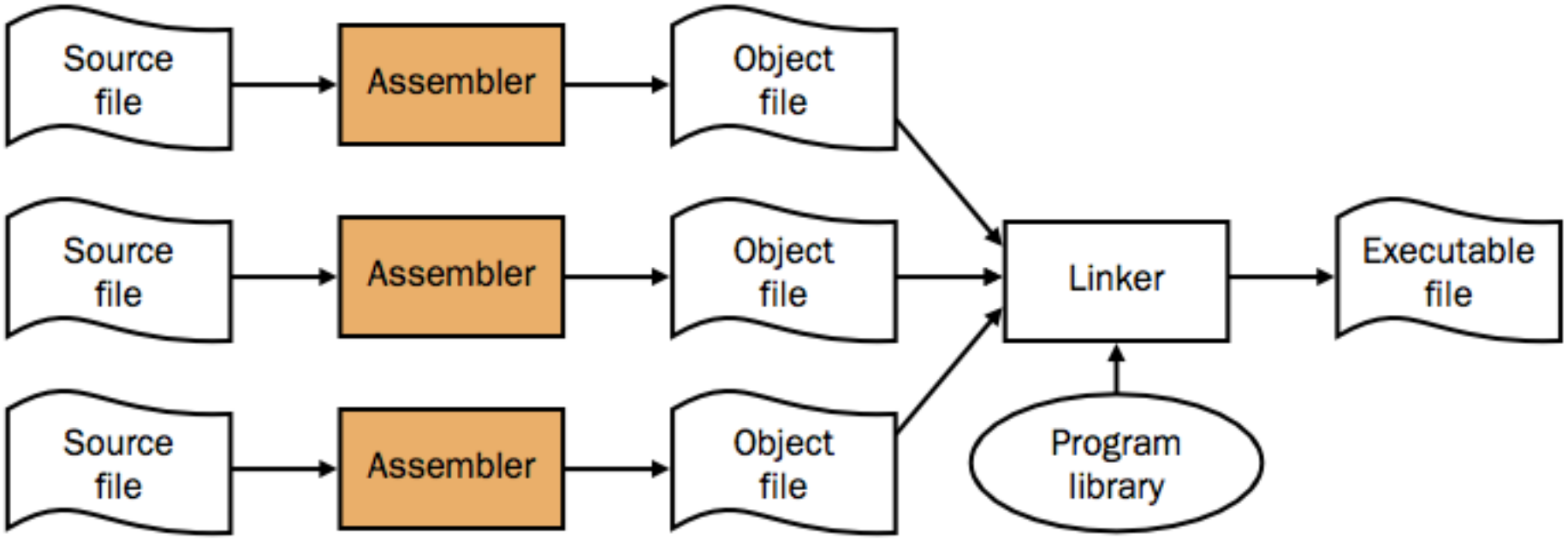


Static/Dynamic


High-level language program
Program
Compiler
Assembler
Linker
Computer
Assembly language program

The UNIX toolchain

`as, ar, ranlib, ld, …`



Bootstrapping a Compiler

- Machine code at the beginning
- Make a simple subset of the language, write a compiler for it
- Use that subset for the rest of the language definition
- Bootstrap from a simpler language
  - Interpreters
- Cross compilation

Modern challenges

- Instruction Parallelism
  - Out of order execution; branch prediction
- Parallel algorithms:
  - Grid computing, multi-core computers
- Memory hierarchy: register, cache, memory
- Binary translation, e.g. x86 to VLIW
- New computer architectures: GPUs, quantum
- Hardware synthesis / Compiled simulations