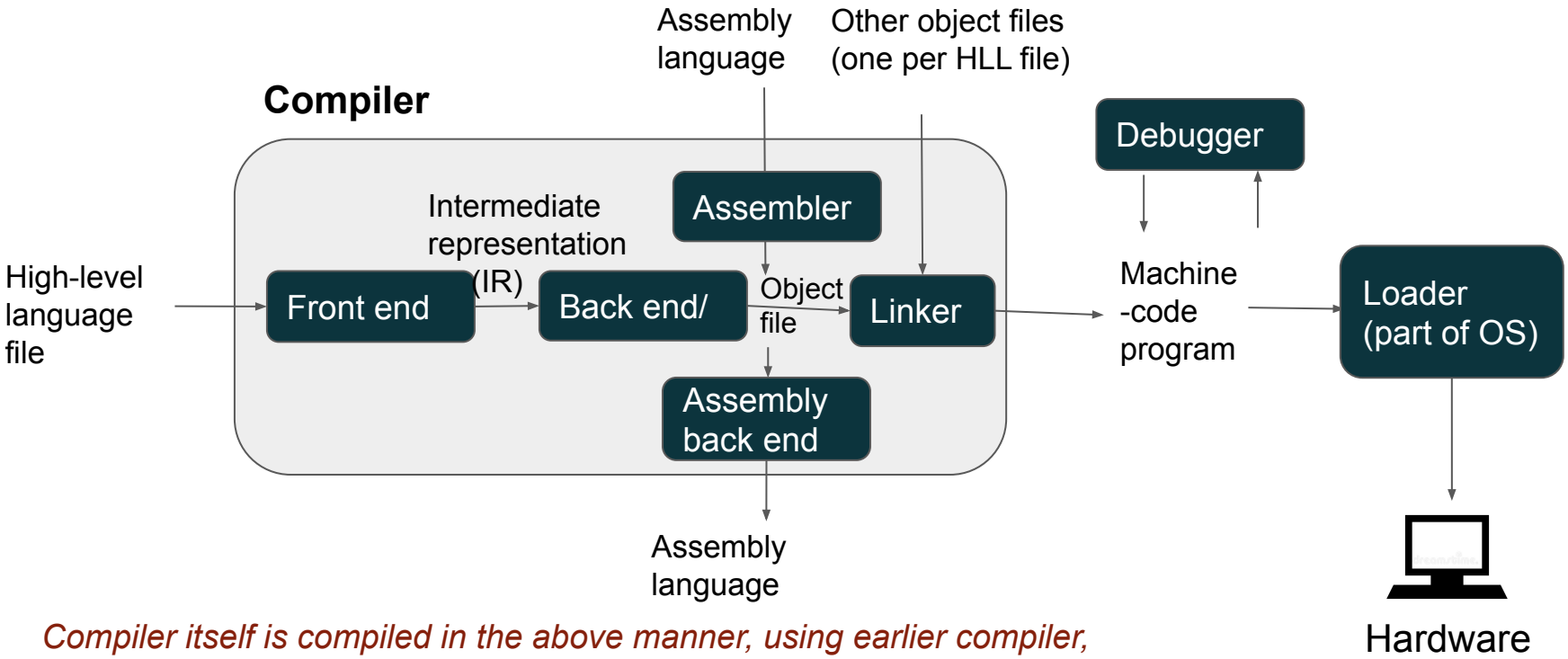
Levels in Modern Computers

Prof Rajeev Barua

E-A 001 -- Slide set 1

Software development process



Compiler itself is compiled in the above manner, using earlier compiler, or after that, using itself.

Levels in modern computers

Computers can be understood at various levels.

- **Reason:** Master the complexity of machines. There is no human alive who know everything about a Pentium chip!!

Two types of levels:

- Hardware: tangible objects.
- Software: sequences of code (at any level).

Software and HW are logically equivalent. Can do a task in either, but may be more efficient in one of them.

Hardware levels

Example levels in modern computers:

- Material Science level
  - Fabricate Silicon and doped Silicon to produce electrical materials
- Solid-state physics
  - Design transistors from materials with desired electrical properties
- Device level
  - Design gates from transistors thus implementing basic boolean functions
- Digital logic design
  - Build hardware modules from gates. **Lang:** Boolean algebra & Register-transfer language
- Micro-architecture level **Lang:** Micro-program or direct execution
  - How to implement instructions using hardware modules
    - How do instructions manipulate the machine's state
- Instruction-set architecture (ISA) level **Lang:** Machine code
  - Specification of the bit formats of machine code instructions

Software levels

Example levels in modern computers:

- Instruction-set architecture (ISA) level **Lang:** Machine code
  - Duplicated as the lowest software level
  - Specification of machine-code program (composing of 0's and 1's)
- Operating System level **Lang:** Machine code with OS features
  - Provide computer management facilities using ISA to running programs.
- Assembly language level **Lang:** Assembly language code
  - A text-based human-readable version of machine code.
  - Programming languages are compiled into assembly language.
- Higher-level language (HLL) levels **Lang:** HLL code
  - Eg: C, C++, Java, Python, ...
  - These are translated by compilers into assembly language or machine code.
- Algorithm level **Lang:** Pseudo-code
  - How to design methods to implement tasks efficiently using high-level languages.

Languages and machines per level

Related concept: Languages and machines per level

Each level there is a view of the machine, and a language to explain concepts at that level.

L0 is machine language of lowest level, Machine is M0

L1 is a higher level language more convenient for people to use.

Especially relevant for:

L0: machine code, M0 is CPU

L1: high-level language (HLL) program, M1 is interpreter software for HLL program

Translation and interpretation

To execute it, we have two choices:

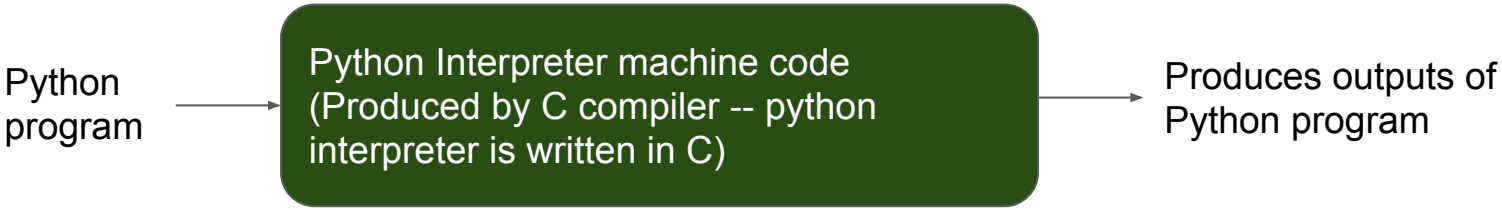
- Translation: L1->L0 offline
- Interpretation: On the fly translation by replacing L1 instr by L0 sequence.

Eg:

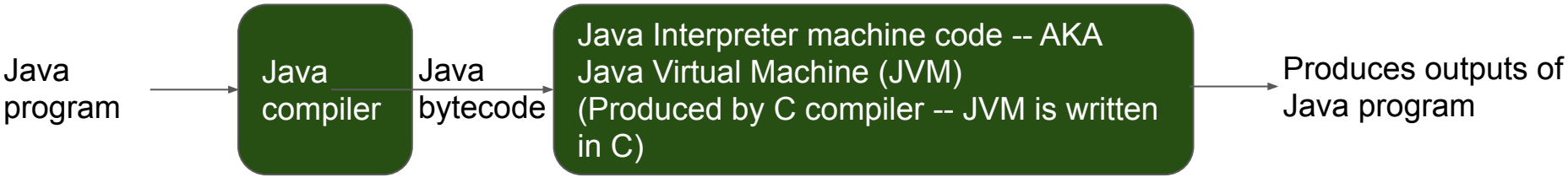
- C is compiled to machine code
- But Python is interpreted using its interpreter
- Java is a hybrid: it is compiled to Java bytecode, which is interpreted.

Example of interpreted languages

Pure interpreted language:



Hybrid:



No machine code is produced for input program in either case!

But interpreters and compilers are in machine code.

Interpreted vs. compiled languages

PLAKSHA UNIVERSITY

Advantages of Interpreted languages:

- Instrumentable
- Portable across platforms

(Used for internal corporate code or externally visible SaaS software services)

Advantages of Compiled languages:

- Protect intellectual property (IP) by not revealing source code.
- Much faster (especially vs. Python)

(Used for externally distributed software, or when speed is important.)

Translation vs interpretation generalized.

Advantages of translation: More efficient. No hardware needed. Faster
Advantages of interpretation: L1 programs are portable, L0 are not.

Corresponding machine M1 (may be a hypothetical construct only, not built!)

Why not built in some cases?

- Not efficient. Simple instructions => simple hardware.

Why have multiple layers?

So long as a converter between levels is available, person can control computer operation at a higher level, without knowing how lower levels (actual hardware) work.