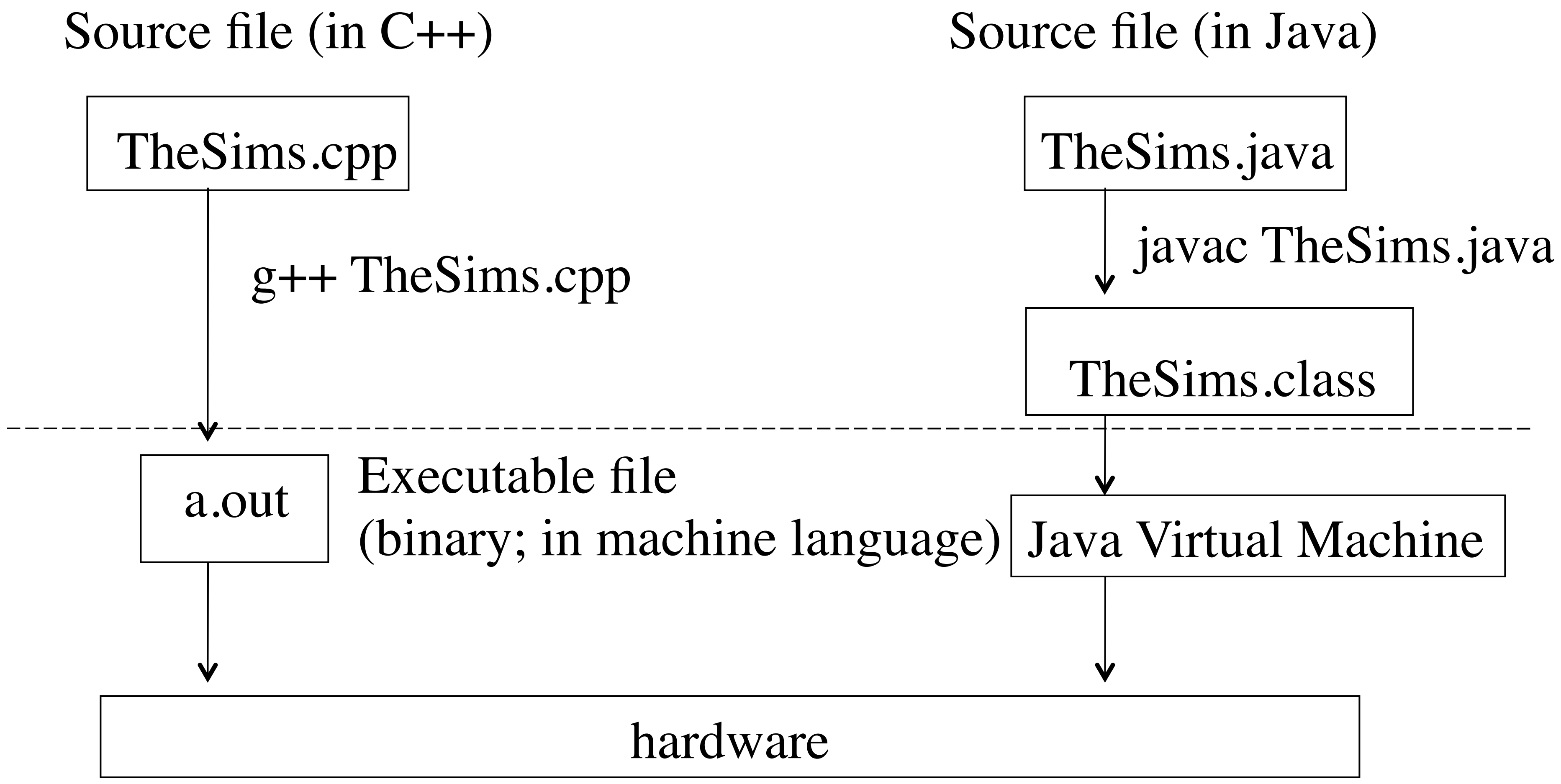
CSc 256 introduction

Understanding machine structures

Why study machine structures?

Why not x86 assembly language?

Understanding machine structures



Machine language / assembly language: simple low-level language “understood” by hardware

Differences between high-level languages (HLLs) and assembly language:

- Syntax/semantics
  - High-level language statements are relatively complex, more “English-like”
  - Assembly language instructions are simple

    add $2, $4, $10 means $2 = $4 + $10
- Portability
  - A HLL program is portable across multiple platforms
  - An assembly language program is specific to an architecture

Why study machine structures?

Most application development is in high-level languages.

But good developers must have reasonable hardware knowledge:

- Compiler development
- Low-level operating system code that communicates with hardware
- Performance-critical code on new hardware
  - Examples: games, multimedia, audio
- GPU (Graphics processor) development
- Multi-core development

Some common CPU architectures

Complex instruction set computers (CISCs):

- Intel x86 (Pentium 4, Celeron PCs, Apple MacBooks)

Reduced instruction set computers (RISCs):

- PowerPC (Apple Macs, IBM servers, Motorola smartphones, Sony PS3, Xbox 360, Nintendo Wii)
- Sun Sparcs
- Intel/HP IA64 (Itanium)
- MIPS (Sony PS 2, Nintendo 64, cable modems, cameras)
- ARM processors (smartphones, iPod, iPhone)

Why not Intel x86 assembly language?

Intel x86 assembly language is relatively complex

- Will cover at end of semester

Intel hardware translates x86 instructions into RISC-like operations before execution

Intel uses RISC-like architectures for non-PC markets:

- Itanium for servers
- Xscale for low-power devices (Kindle, media players; sold to Marvell 2006)