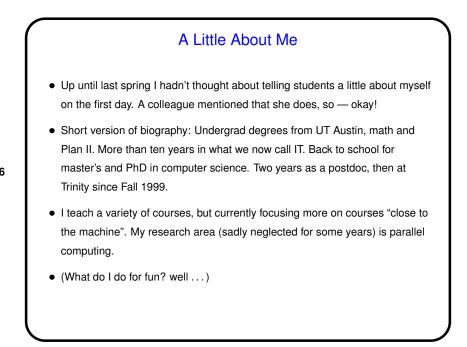


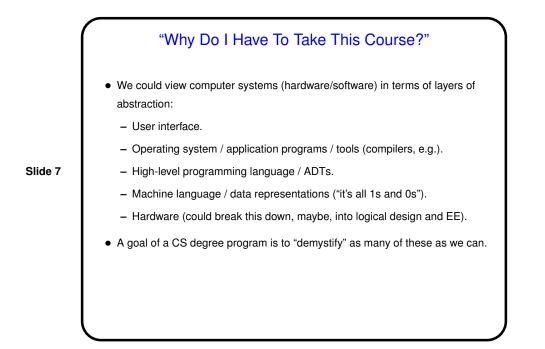
Course FAQ, Continued

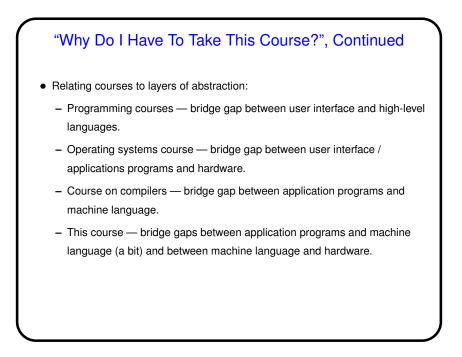
• "What computer(s) can I use to do programming homework?"

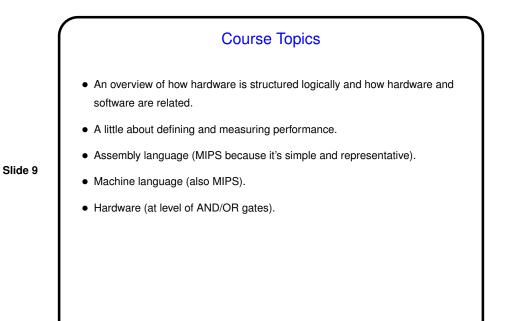
Easiest option may be department's Linux classroom/lab machines. You should have physical access (via your TigerCard) to all the classrooms and labs. You should also be able to log in remotely to any that are booted into Linux, or to a cluster of Linux-only machines in ITS's server room (names diasnn, where nn ranges from 01 to 05).

Slide 5









Why Study Assembly / Machine Language?
Understand the general principles of how things work at this level, which helps you:

Write more efficient programs.
Understand operating systems (which also helps you write more efficient

Slide 10

- Generally understand better what's really happening in the machine.
- It might be fun?

programs).

Introduction

 "Computers are everywhere" — you know about servers and desktops and smaller computing devices, all of which are more and more central to our lives, but also consider "embedded processors", largely invisible but even more prevalent.

Slide 11

- It seems to be a truism that however fast computers can process information, they can't keep up with humans' ability to imagine things for them to do. So performance matters.
- We'll start with an overview of hardware and software and how they interact (cf. textbook subtitle) and also talk a little about measuring performance.

"Below Your Program" — Review?

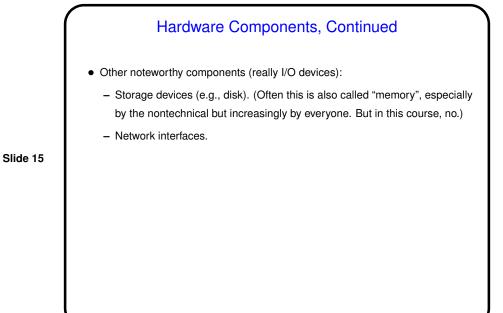
- Most programming these days is done in a high-level language (HLL), often using a lot of library code. Processors, however, can't execute it directly. How do we get from HLL to something the processor can do? Traditional view:
- First step is to *compile* conceptually, to *assembly language* (symbolic representation of instructions the processor can execute).
- Next step is to translate assembly language into machine language (actual instructions for processors, in 1s and 0s), a.k.a. object code. Might be combined with compiling.

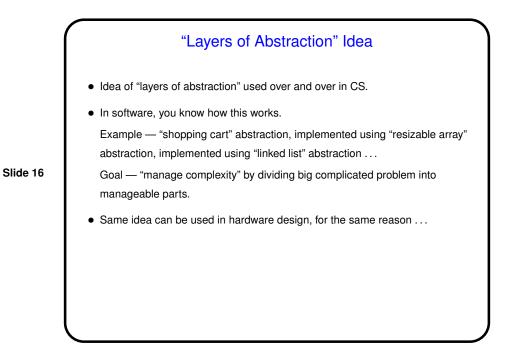


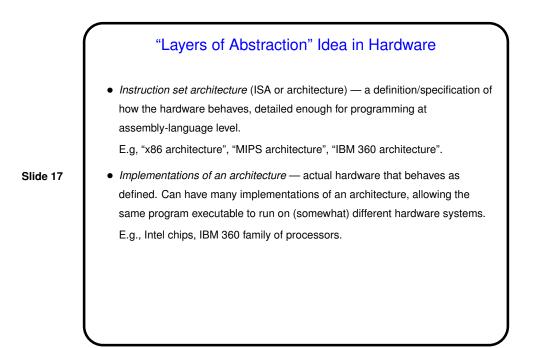
- Final step is to combine object code for your program with library object code. Can be done as part of compiling process to create an *executable file* or at runtime, or some combination of the two.
- Actual execution of program typically involves operating system (something manages physical resources / provides abstraction for applications).
 Contents/format of executable files depends on operating system as well as hardware.
- Worth noting that some languages/implementations don't exactly follow this scheme — some languages (e.g., shell scripts) are translated/interpreted at runtime, and others (e.g., Scala and Java) are compiled to machine language for a virtual processor (the JVM), which may then be translated into "native code" at runtime.



- Input devices way to get info into computer from outside. Examples include keyboard and ...?
- Output devices way to get information back to outside world. Examples include display and ...?
- Slide 14
- Processor "brain" that does actual calculations, etc. Can divide into
 - Datapath stores values, performs operations (e.g., addition).
 - Control "puppet master" for datapath.
- Memory stores values, "scratch pad" for calculations. Now typically includes "main memory" and "cache memory" (possibly multiple levels).







"Layers of Abstraction" Idea in Hardware, Continued
• For programs that will run on a computer with an operating system, also define *application binary interface* (ABI) that describes application's interface with both hardware and operating system.

Minute Essay

 (Most lectures will end with a "minute essay" — as a quick check on your understanding, a way for me to get some information, etc., and also to track attendance. Just put your answer in the body of the message; no Word documents please, and put "minute essay" and the course in the Subject line.)

- Tell me about your background: What programming classes have you taken (at Trinity or elsewhere)? What programming languages are you reasonably comfortable with?
- What are your goals for this course? Anything else you want to tell me? (Maybe something interesting you did over the break?)