Slide 1

Administrivia

- Notice that these "slides" are available on the course Web site (preliminary version before class, final version sometime after class).

Slide 2

Introduction

- "Computers are everywhere" — you know about desktops and servers, which are more and more central to our lives, but also consider "embedded processors", largely invisible but even more prevalent.
- 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.
- Factors that affect performance include both the ones you learn about in programming courses (order of magnitude of algorithms, e.g.) and lower-level ones (how well the compiler can map HLL onto hardware in various respects, how fast the hardware can execute instructions).

Slide 3

"Below Your Program" — Review?

- Most programming these days is 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?
- 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.

Slide 4

"Below Your Program", Continued

- 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.

Slide 5

Hardware Components — An Abstract View

- Input devices — way to get info into computer from outside. Examples include mouse and . . . ?
- Output devices — way to get information back to outside world . Examples include display and . . . ?
- 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).

Slide 6

Hardware Components, Continued

- Other noteworthy components (really I/O devices):
  - Storage devices (e.g., disk).
  - Network interfaces.

Slide 7

"Layers of Abstraction" Idea

- Idea of "layers of abstraction" used over and over in CS.
- In software, you know how this works.

  Example — "shopping cart" abstraction, implemented using "resizable array" abstraction, implemented using "linked list" abstraction . . .

  Goal — "manage complexity" by dividing big complicated problem into manageable parts.

Slide 8

"Layers of Abstraction" Idea, Continued

- Same idea can be used in hardware design, for the same reason. So we talk about
  - *Instruction set architecture* (ISA or architecture) — a definition/specification of how the hardware behaves, detailed enough for programming at assembly-language level.

    E.g, "x86 architecture", "MIPS architecture", "IBM 360 architecture".
  - *Implementations of an architecture* — actual hardware that behaves as defined. Can have many implementations of an architecture, allowing the same program executable to run on (somewhat) different hardware systems.

    E.g., Intel chips, IBM 360 family of processors.
- For programs that will run on a computer with an operating system, also define *application binary interface* (ABI) that describes application's interface

Slide 9

with both hardware and operating system.

Slide 10

Minute Essay

- The textbook uses cell phones as an example of something that uses an embedded processor. (I think they do not mean to include “smart phones”, which seem more similar to general-purpose computers.) What are some other devices or products you think probably use embedded processors?