Slide 1

Administrivia

- (Grading status.)
- “Slides” last time include many not used in class. Meant as a summary. Skim . . .

Slide 2

Minute Essay From Last Lecture

- The code shown is basically a version of an example I use a lot, including in CSCI 1120. Some people noticed. I’ve put it and another “floating point is strange” example on the course “sample programs” page.

Slide 3

A Little About Circuit Design

- Goal — sketch design of a (hardware) implementation of MIPS architecture in terms of some simple building blocks (AND and OR gates, inverters).
- Things we'll need:
  - Something to implement instructions: ALU (arithmetic/logic unit).
  - Something to implement registers: register file.
  - Something to implement fetch/decode/execute cycle: control logic.

Slide 4

Implementing Logic Gates — Executive-Level Summary

- The ones and zeros of low-level software become two distinct voltages in hardware, and the logic of Boolean algebra is implemented using "switches" (things that connect an input to an output, or not, depending on the state of a control input).
- Currently these switches are (usually?) transistors. In widely-used "CMOS technology", there are two types of switches, one that's good if the input is "one" and one that's good if the input is "zero". These can be combined to implement logic. Simple example: Inverter. (See link from "useful links" page.)

Slide 5

Circuit Design — Overview Continued

- AND and OR gates implement Boolean-algebra functions of the same names; inverter implements "not".
- A word about notation: We'll use the textbook's notation for Boolean algebra, which alas is (probably?) different from what you used in CSCI 1323.

CSCI 2321	*CSCI 1323*
$a \cdot b$	$a \wedge b$
$a + b$	$a \vee b$
$\overline{a}$	a'

Slide 6

Circuit Design — Overview Continued

- "Combinational logic" blocks implement Boolean functions/operations — map input(s) to output(s) without a notion of persistent state. (Think of these as "pure" functions that don't change any variables but can have multiple output.)
- "Sequential logic" blocks also implement Boolean functions/operations but include a notion of persistent state. (Think of these as methods in object-oriented programming, which map input(s) to output(s) but also have access to member variables that can be read/written.)

Combinational Logic

- How to specify combinational logic block?
- One way — truth table with one line for each combination of inputs.
- Another way — Boolean-algebra expression(s) that define output(s) in terms of input(s).
- (Sketch example — 1-bit “adder” and how to combine them to get something that will add n-bit integers.)

Slide 7

Minute Essay

- Bitwise AND is an example of an operation that can be implemented using an array of 32 independent identical logic blocks. What’s another operation that can be implemented that way?
- Anything that seems especially puzzling about this material?

Slide 8

Minute Essay Answer

- Any of the other bitwise logical operations (e.g., OR) can be implemented with an array of independent elements. Addition and subtraction, though, require the array elements to be connected.