Slide 1

Administrivia

- None.

Slide 2

Minute Essay From Last Lecture

- Question: For the logic block drawn on the board, what results does it give for all possible inputs (combinations of a and b)?
- Answer? How to write this in terms of "and" and "or"?

Slide 3

Big Picture (Recap)

- Ultimately what we want to do is "build" an implementation of the MIPS architecture:
  - Memory.
  - Registers (32 general-purpose, plus "special-purpose" we haven't talked much about).
  - Something to do the basic fetch/execute/decode cycle for the instructions we've defined.
- Building blocks are AND and OR gates and inverters. (These in turn can be built from things that work as switches — transistors, in current practice.)
- We can connect these into "circuits" with "inputs" and "outputs". Idea is that if we change inputs (provide voltages representing ones and zeros), after some delay the answers we want appear as outputs. (Do you think the delay depends on anything about the circuit? Size? Complexity?)

Slide 4

Big Picture (Recap, Continued)

- Obviously this is going to be complicated. We start by building something that will do arithmetic and logical operations ("ALU"). We'll figure out later (next chapter) how to manage its inputs/outputs.
- In terms of the "five classical components", ALU is part of "datapath" and will be managed by "control" we'll build in chapter 5.

  You'll notice that in the figures some inputs are shown in black and some in color? The ones in color are "control inputs".
- A general comment: In chapters 4 and 5, if you understand the "important" figures, you probably understand what you need to.

Slide 5

Building an ALU (Recap/Review)

- So far we have an ALU that does `and`, `or`, add, subtract. (Notice how we got subtract almost for free because we're using two's complement notation.)
- Before moving on, notice that real hardware doesn't use ripple adders — too slow. (Why?)

  Instead, they do something more complex but faster — "carry lookahead". "Executive-level" summary:
  - For each bit, calculate whether inputs "generate" or "propagate" a carry-in. Can be done in parallel for all bits.
  - Combine these results with carry-in for least significant bit to get carry-in for all bits, faster.
  - Result is shown in figure 4.24.

Slide 6

Building an ALU, Continued

- What else do we need for MIPS instructions so far?
  - `sll`, `srl`, `sra` — defer for now. Actually usually done with "barrel shifter" outside ALU.
  - `j`, `jr`, `jal` — also defer for now, since more about managing program counter and registers than about arithmetic/logical operations.
  - `slt`.
  - `bne`, `beq`.
- Keep in mind — for now we're not trying to figure out where inputs come from (especially the "control inputs"). We'll do that in chapter 5. For now we just want to build something that does the basic arithmetic/logical operations we need.

Slide 7

Building an ALU, Continued

- What should we do for `slt`? Is there something we already have that would help? $a < b$ — when?

Slide 8

Building an ALU, Continued

- $a < b$ when $a - b < 0$ — which is true when MSB (most significant bit) is 1.
- So `slt` result can be zeros in all but LSB, and that can be MSB of result of $a - b$.
- Result? See figures 4.17 and 4.18. (At this point we also add something to test for overflow.)

Slide 9

Building an ALU, Continued

- What should we do for `bne` and `beq`? Is there something we already have that would help? $a = b$ — when?

Slide 10

Building an ALU, Continued

- $a = b$ when $a - b = 0$. $a \neq b$ when $a - b \neq 0$. How to check an output for zero? Probably easier to check for non-zero — at least one bit non-zero.
- Result? See figure 4.19.

Slide 11

More Arithmetic — Multiplication

- As with addition, first think through how we do this "by hand" in base 10. (Review terminology: In $a \times b$, call a the "multiplicand" and b the "multiplier".) Example?

  Notice also that overflow could be a lot worse here — so normally we'll compute a result twice as big as the inputs.

- We can do the same thing in base 2, but it's simpler, no? computing the partial results is easier.

  (To be continued.)

Slide 12

Minute Essay

- How are you doing with the reading? Does the textbook explain things in ways that make sense to you? (And has your opinion changed since you read, say, chapter 1?)