Slide 1

Administrivia

- Review sheet for midterm on Web. We will use Monday's class for a review.

Slide 2

Minute Essay From Last Lecture

- Question: Here's what we talked about in chapters 3 and 4:
  1. MIPS instruction set.
  2. Translating C to MIPS assembly language.
  3. Translating assembly language to 1s and 0s.
  4. Representing numbers (integers and reals) in binary.
  5. Computer arithmetic.
  6. Turning Boolean functions into circuits.

  How well do you think you understand each of these?
- Answers? Let's take a poll.

Building a Datapath, Recap/Review

- What we're doing is figuring out what "functional units" we need to implement a representative subset of MIPS instructions.
- Last time we talked about what's needed just to store the program and step through it — instruction memory, program counter.

Slide 3

Datapath for R-format Instructions

- What do we need in order to execute R-format instructions (`add`, `sub`, `slt`, etc.)? E.g., `add $t0, $t1, $t2`.

  We need to be able to read values for two registers, do arithmetic/logical operation, write a value for a register.
- Functional units we need:
  - "Register file", as discussed last time. See figure B.18.
  - ALU. What are inputs, outputs?

  Connect them up as in figure 5.7.
- Trace through example — `add $t0, $t1, $t2`.

Slide 4

Datapath for Load/Store Instructions

Slide 5

- What do we need in order to execute `lw`, `sw`? E.g., `lw $t0, 4($t1)`. What we have to do here is
  - Compute a memory address — by reading contents of a register, getting offset from instructions, sign-extending offset, and adding.
  - For `lw`, read contents of this memory location and put in register.
  - For `sw`, store contents of register in this memory location.

Datapath for Load/Store Instructions, Continued

Slide 6

- Functional units we need:
  - Register file and ALU as before.
  - “Sign-extension unit” — CL block, right? inputs and output? see figure 5.8.
  - “Data memory” — state element, right? inputs and outputs?

  Connect them up as in figure 5.9.
- Trace through examples:

```
lw $t1, 4($s1)
sw $t1, -4($s1)
```

Slide 7

Datapath for Control-Flow Instructions

- What do we need in order to execute `beq`, `j`? E.g., `beq $t0, $t1, L1` or `j L2`.
- For `j`, what we have to do is replace last 28 bits of PC with 26 bits from instruction. Mostly a matter of routing things, so we'll do this later.
- For `beq`, what we have to do is
  - Compute "branch address" by taking (PC+4) and adding offset from instruction, sign-extended and shifted.
  - Compute difference between registers; use "is it zero?" bit to decide whether to change PC.

Slide 8

Datapath for Control-Flow Instructions, Continued

- Functional units we need:
  - Something to compute PC + 4.
  - Sign extender, ALU, and register file as before.
  - Something to shift offset from instruction 2 bits left.
  - Something to calculate PC + 4 + offset. (Why can't we do this with the ALU?)

  Connect them up as in figure 5.10.
- Trace though example — `here:  beq $t1, $t2, here`.

Minute Essay

- Is there anything you particularly want me to review Monday?