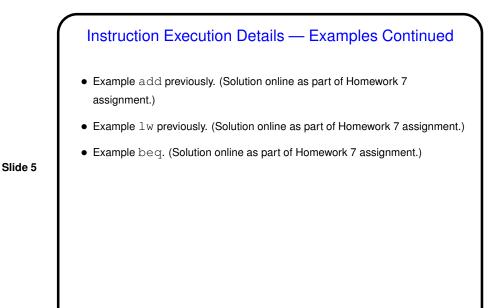


Designing a Processor — Recap/Review

- We're working through the design of a processor that implements a subset of the MIPS architecture.
- Design starts with Figure 4.1, and previously we got as far as Figure 4.17, which (with the supporting tables) shows a complete design for lw/sw, selected arithmetic and logic instructions, and beq.



Homework 7 Help — Tracing Operation of the Processor Circuit

• (I'll go through these slides quickly in class; they may be a useful summary when you start doing the assignment. Time permitting I'll also put this information in the assignment too.)

- In this homework, you'll trace through what the circuit in Figure 4.17 is actually doing. Examples in lectures for 4/13 and 4/15. Idea is for you to trace through what the circuit actually does rather than what you think it should do. But the two should match!
- So, you start with what you know current saved value of the PC and what's at that address (in instruction memory) and contents of selected registers and data memory locations - and work from there. Taking the first few steps ...



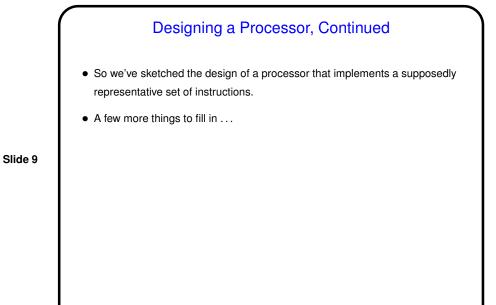
- Right away you can write down output of PC and input/output of instruction memory. The problems give you the machine language for the instruction; it may be helpful to split it into fields before going on.
- Now you can write down all the control signals, the inputs and output of the top left adder, and the register-number inputs to the register file. You can get the control signals from the table in Figure 4.18.
- Once you have those, you can write down outputs of the register file and start figuring out what the main ALU is doing. You can also determine whether the top right adder and the data memory will be used (based on control signals).

Homework 7 Help, Continued

• Figuring out what the ALU does ... You need to determine what operation it's doing (based on the ALUOP control signal and the instruction function field, as shown in Figure 4.13). You also need to determine what the second operand is (contents of a register? sign-extended value from instruction?), again using control signals.

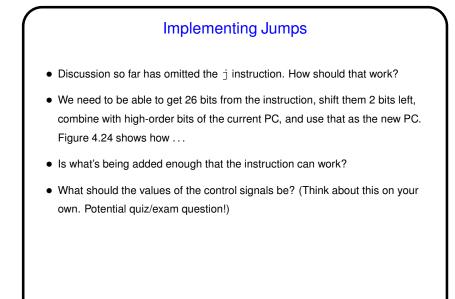
Slide 8

• "And so forth" ...

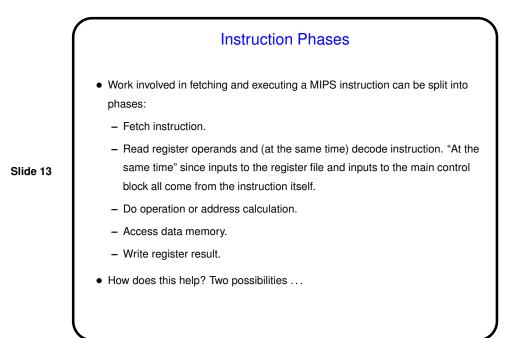


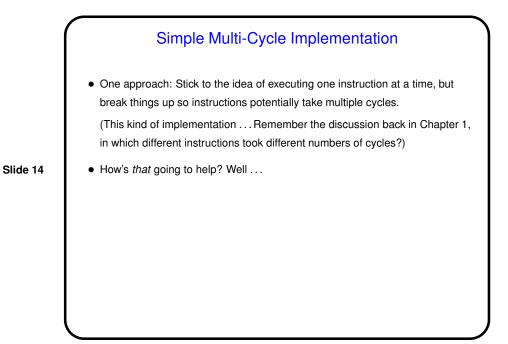


- Design shows instruction and data memory separate.
- Why? isn't it all just ones and zeros? Yes, but ... (Minute-essay question.)



Multi-Cycle Implementations
So, we have a sketch for an implementation that executes one instruction per cycle. But clearly this isn't how all real systems work (if nothing else, most don't separate instruction memory from data memory).
Why not? means cycle time is limited by length of longest path through the whole circuit, while many instructions can be done faster.
What to do? break up work into multiple pieces ...



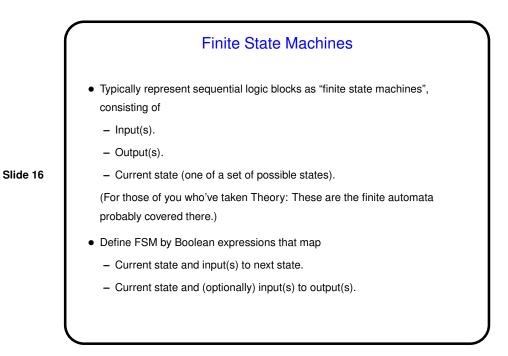




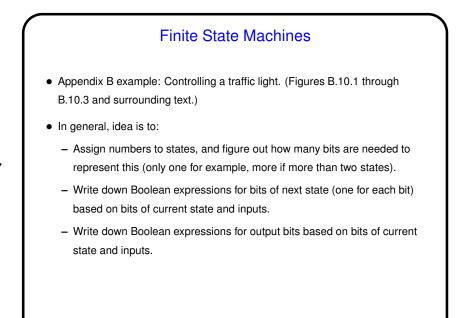
- One potential payoff is skipping unused phases: E.g., R-format (arithmetic/logic) instructions don't need to access data memory,
- Also, we don't need separate instruction/data memories.

 However, control logic becomes more complex: Must do everything we were doing before, plus keep track of which phase we're in. We can do that with a finite state machine (discussed in Appendix B, and it looks like we have time to say more about it now).

• Some previous editions of the textbook lay out a design for this.



8



Pipelined Implementation

- Another approach is to use "pipelining": Modeled after assembly line; many real-world analogies possible. Textbook describes a laundry "assembly line", with stages corresponding to washing, drying, folding, and putting away.
- Could base a pipelined implementation of MIPS on the same phases used for a multi-cycle implementation, with one pipeline stage per phase.
- How does this help? well, doesn't make individual instructions faster, but means you can get more of them done in a given time.
- Like the simple multi-cycle implementation, it means added hardware complexity ...

(To be continued!)

Slide 17

Minute Essay

The design sketched so far has two separate memory blocks, one for instructions and one for data. This turns out to be needed for the simplest implementation, one in which each instruction executes in a single cycle. Why? is there something different about the types of values to be stored, or is there some other reason? (*Hint:* Think about what has to happen for lw.)

Slide 19

• How did the way I asked you to do Quiz 4 work for you? Anything that would make it work better? I feel like there's just no good way to deal with quickly drawing pictures but hope you found an option that works okay for you?

Minute Essay Answer

• For lw, you need to be to both load the instruction and also load something from the specified address. (This is an open-ended version of one of the textbook's "check yourself" questions for section 4.3.)