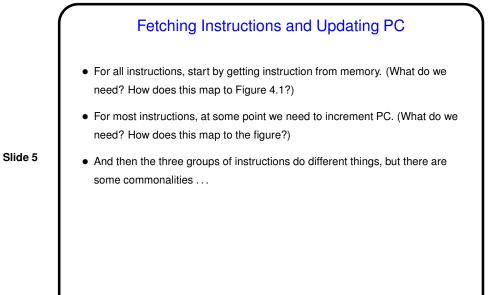
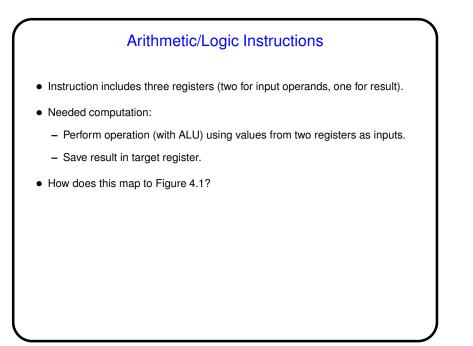


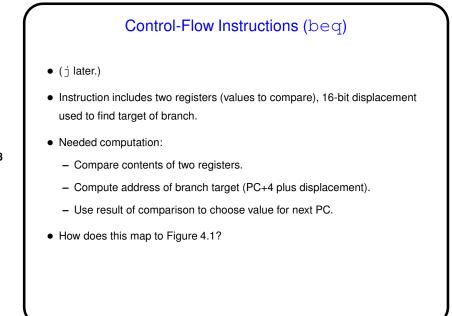
Clocking — Recap/Review
Hardware will include something that implements a "clock cycle".
State elements' inputs are "sampled" during one phase of this cycle, and outputs change as inputs change. (Some details in Appendix B. Textbook reviews some detail in Chapter 4. Okay to skim! The key point is that clocking is needed to avoid hardware race conditions.)
Length of cycle determines how complicated the various logic blocks can be (or vice versa).



Memory-Access Instructions • Instruction includes two registers (one for base address, one for where to load into / store from), 16-bit displacement. • Needed computation: - Add displacement to register containing address. - Use result to access memory, loading/storing to/from register containing data. • How does this map to Figure 4.1?



Slide 7



Slide 8

4



- Figure 4.1 seems to have ways to do everything we need to do paths for data to flow from one place to another, including into ALU(s) for computation.
- For every instruction we're in some sense doing the same things (have each ALU compute something), but some results are essentially discarded.
 (Example beq computes two "next instruction" addresses, but only stores one back into the PC.) This is very typical of how things work at this level!

Slide 9

Control Logic So we have a "datapath" that can do things, but there are some inputs that aren't connected to anything. An analogy — the datapath is a puppet, and these inputs are its strings? the "control logic" — combinational logic whose input is the current instruction plus any other needed information and whose output is those disconnected inputs to datapath. Figure 4.2 shows that addition.

