Slide 1

## Administrivia

- Reminder: Quiz 5 Wednesday.
- References to numbered figures in these slides are to figures in the textbook.

Slide 2

## Minute Essay From Last Lecture

- Question: Give Boolean expressions for two more rows of truth table in figure 5.27 — `MemtoReg` and `Branch`. (Inputs, recall, are the 6 bits of the opcode.)
- Answer?

Slide 3

## Datapath for Multiple-Cycle Implementation, Recap

- Assume cycle time is long enough for any of the following, but not two of them in sequence:
  - Memory access (read *or* write).
  - Register-file access (two reads *or* one write).
  - ALU operation.
- First sketch of datapath — figure 5.30. Notice that now we also need some "temporary storage" areas. (Why?)
- Add needed multiplexors to get figure 5.31. Compare this to datapath for single-cycle implementation (figure 5.17) — overall, probably will be cheaper to build.
- Add control signals to get figure 5.32.

Slide 4

## Datapath for Multiple-Cycle Implementation, Continued

- Is this okay? For R-format instructions and load/store word, yes. For `j` and `beq`, we need some more stuff — figure 5.33.

  Meanings of control signals in figure 5.34.
- Notice that most *but not all* of the "temporary storage" areas change every step. What's the exception, and why?
- Next step — define "steps" for instructions.

Slide 5

## Instruction Fetch Step

- Pseudocode:

```
IR = Memory[PC]
PC = PC + 4
```

- Okay to do these both in one cycle? (Why?)
- What control signals do we need?
- This will be the first step of all instructions. Notice that updated PC will be in two places during the next cycle.

Slide 6

## Instruction Decode and Register Fetch Step

- Pseudocode:

```
A = Reg[IR[25-21]]
B = Reg[IR[20-16]]
ALUOut = PC + (sign-extend(IR[15-0]) << 2)
```

- Okay to do all this in one cycle? Could we have combined with previous step?
- What control signals do we need?
- This will be the second step for all instructions — assume we don't know yet which instruction we have, so do things that will help or at least not hurt for all.

Slide 7

## Execution / Memory Address Computation / Branch Completion Step

- For instructions that reference memory, compute address:

  ```
  ALUOut = A + sign-extend(IR[15-0])
  ```

- For R-type instructions, operate on register data:

  ```
  ALUOut = A op B
  ```

- For branch, decide whether to take branch and update PC:

  ```
  if (A == B) PC = ALUOut
  ```

- For jump, update PC:

  ```
  PC = PC[31-28] || (IR[25-0] << 2)
  ```

Slide 8

## Memory Access / R-Type Instruction Completion

- For memory-access instructions, read/write memory:

  ```
  MDR = Memory[ALUOut]
  ```

  or

  ```
  Memory[ALUOut] = B
  ```

- For R-type instruction, store result into register:

  ```
  Reg[IR[15-11]] = ALUOut
  ```

## Memory Read Completion

- For memory read, store value in register file:

```
Reg[IR[20-16]] = MDR
```

**Slide 9**

## Now What?

- As before, figure out how to generate control signals for each step (from instruction, etc.).
- Also figure out how to produce the right sequence of steps for each instruction.

**Slide 10**

Slide 11

## Minute Essay

- Looking at figure 5.33, what control signals do we need for the “instruction fetch” step? Give values for all of the following (or as many things on the figure as you have time to decide about):

  RegDst<br>
  MemRead<br>
  IorD<br>
  PCWriteCond<br>
  RegWrite<br>
  MemWrite<br>
  IRWrite<br>
  ALUSrcB<br>
  ALUSrcA<br>
  MemtoReg<br>
  PCWrite<br>
  PCSrc

  Pseudocode, again:

  ```
  IR = Memory[PC]
  PC = PC + 4
  ```

- Reminder: Homework 5 due at 5pm today.