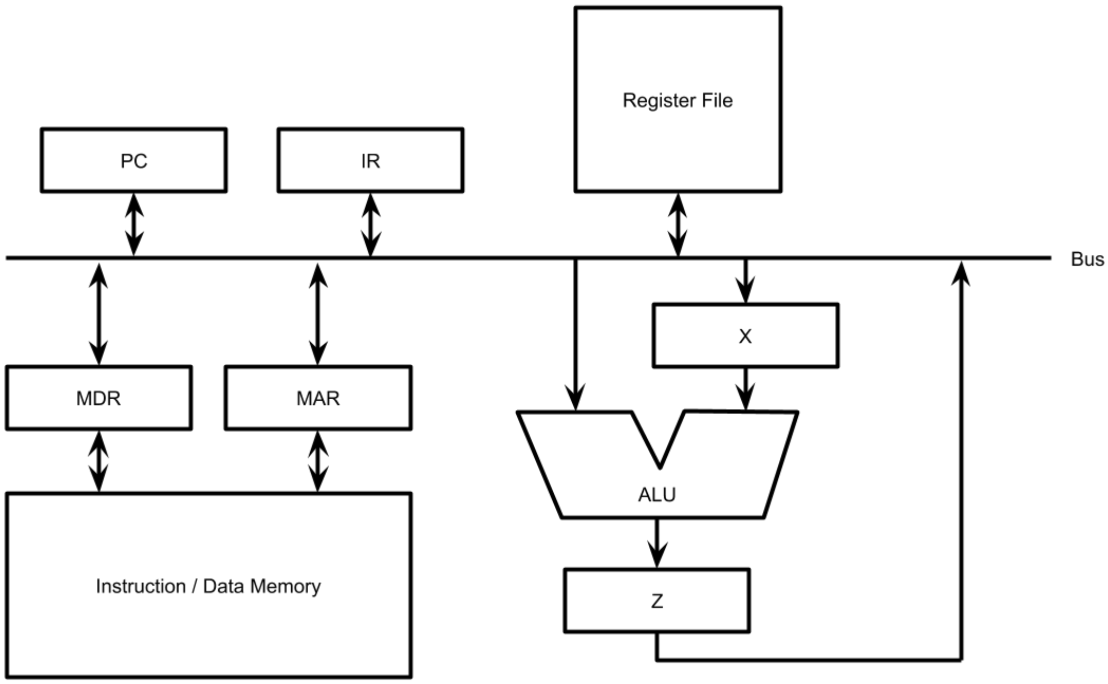
1. From last time
   a. Went over the single bus machine below



   b. Can see that one bus is a huge limiting factor
      i. Lots of contention for the bus, many things want to use it
      ii. Idea: another bus
         1. Works better, but not perfect
         2. Good for A = A + B
         3. Adds more cost to machine to support another bus
      iii. Solution: third bus
         1. Now we can handle A = B + C
         2. Costs even more
   c. Three bus CPU takes fewer cycles than one or two buses to execute an instruction
      i. Still takes more than 1 to do so, though
   d. Couple of reasons why we can't reduce to 1 cycle
      i. Must increment PC to reach next instruction
         1. Requires using the ALU to do so
         2. Solution: place an adder by the PC to do only that
      ii. Complex addressing modes require multiple trips to memory
         1. Solve by limiting the instruction set to a load/store architecture
         2. ALU instructions can only use registers
         3. With all the above, instructions can execute in a single cycle
         4. Idea behind RISC

2. RISC versus CISC machines
   a. Review of material from ECS 50
   b. Back in the 1970s, had dozens of machines and instruction sets
      i. CISC – complex instruction set computer
      ii. Lots of powerful instructions that did a lot at once

- c. Different teams at Stanford and Berkeley looked over this
    - i. Looked at making instruction sets simpler
    - ii. Keep only a few instructions that can be done very fast
    - iii. Found that if you implement a program in this method, can be done faster than CISC
- d. RISC – reduced instruction set computer
    - i. Less powerful instructions, and will need more of them to run same program versus CISC
    - ii. However, can make the clock *much* faster to compensate

3. Back to buses
    - a. Key characteristic of bus
        - i. Shared transmission medium
        - ii. Anything using the bus must obtain the “right” to use the bus first
    - b. Functional groups
        - i. Data lines
        - ii. Address lines
            1. Select where to source from, where to send to
            2. Could pick different parts of the CPU, memory, or I/O
        - iii. Control lines
            1. Control use of data and address lines
            2. Variety of signals
                - a. Memory write/read
                - b. I/O write/read
                - c. Transfer ACK (acknowledgment), data has been taken from or placed on bus
                - d. Bus request/grant, to give permission to somebody who wants the bus
                - e. Interrupt request/ACK, will discuss this later
                - f. Clock, same one we’ve been talking about
                - g. Reset
    - c. Types of buses
        - i. Dedicated – permanently assigned to a subset of components, or to one function
            1. Example: data bus versus address bus
            2. Need more buses to send all information
        - ii. Time multiplexed – using bus for multiple functions based on clock
            1. Send address for first part of clock, then data for second
            2. Don’t need as many buses, but can’t transmit at same speed a dedicated bus can