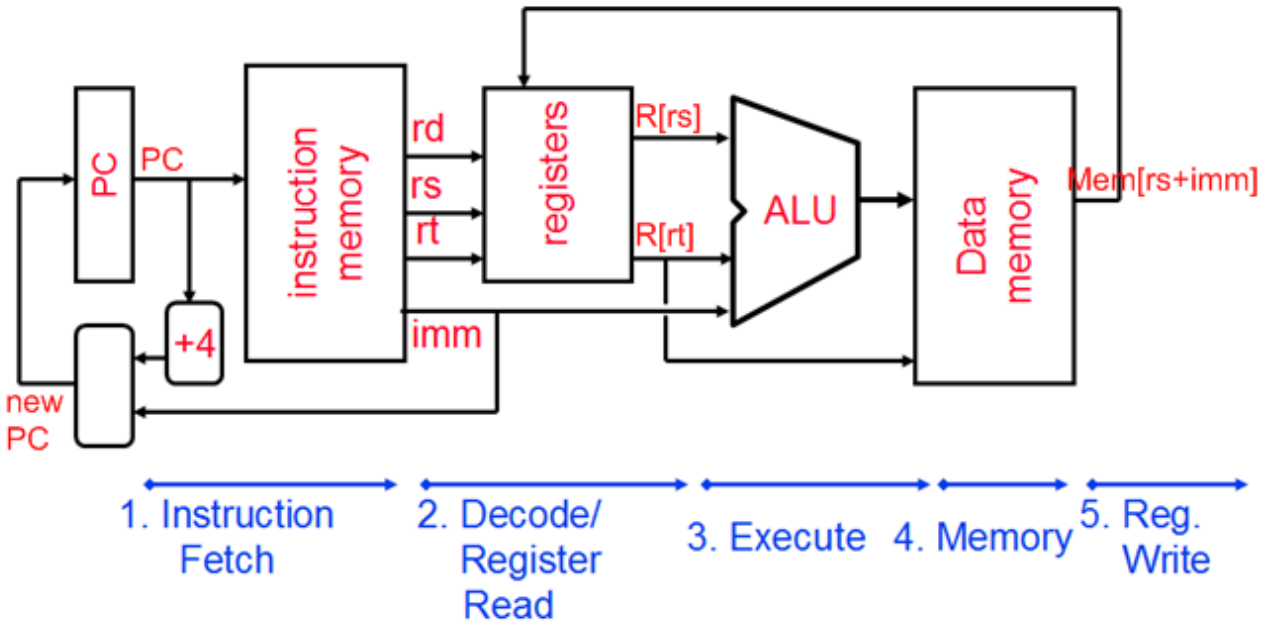
Quick Review

- Finite state machines (FSMs) are a useful tool for solving many problems in computer science.
- An FSM can be written down in truth table format, with one row in the truth table for each transition in the FSM.
- We can use a truth table to create a Boolean expression for the output based on the inputs, and use this expression draw a circuit level diagram. We can also use Boolean algebra to simplify the expression first before drawing the diagram.

CPU Design

Here is the basic datapath as discussed in lecture, shown in simplified format.



rd, rs, and rt are 5 bit wires, imm is a 16 bit wire. All other wires are 32 bits wide.

Register Transfer Language(RTL)

- Use to describe flow of data: *dest* ← *src*
- Each line happens in parallel (at the same time): a ← b, c ← b
- In MIPS, use R[x] for register x, and Mem[y] for memory at y

Exercises

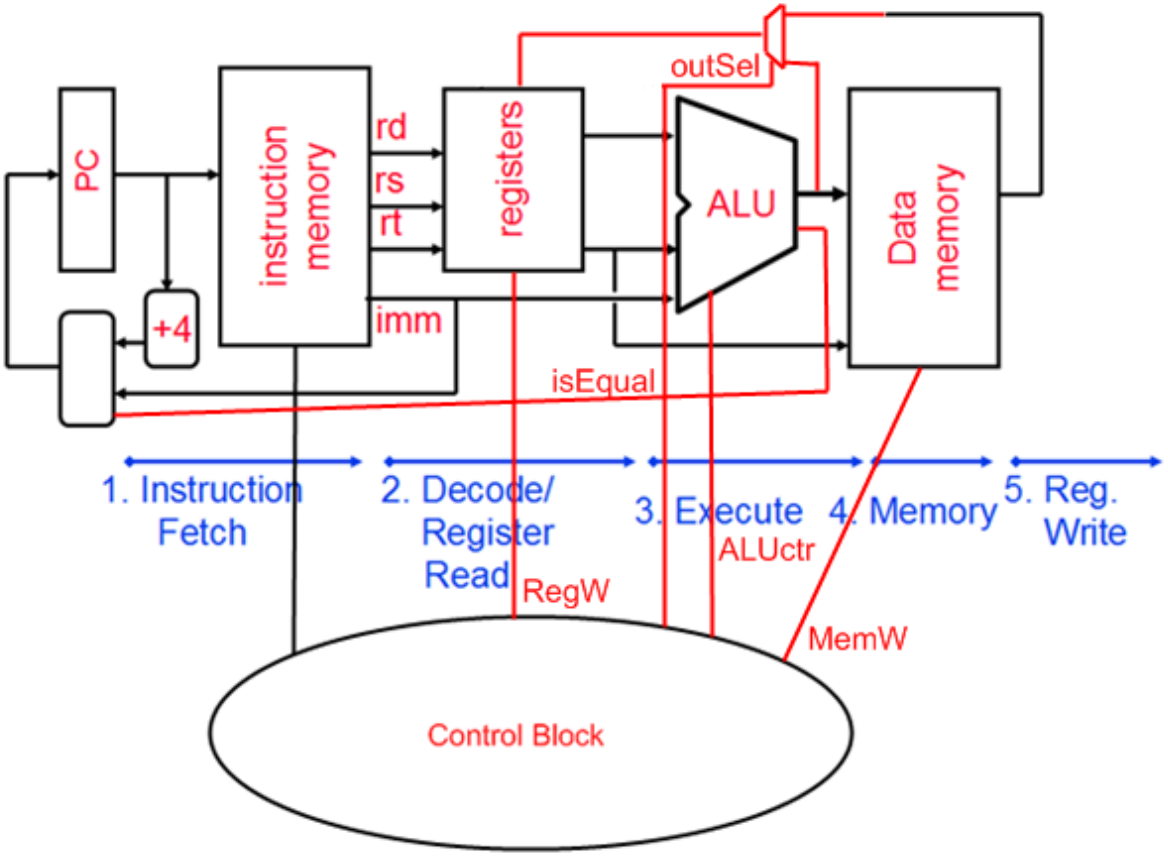
For the following exercises, assume that the ALU can output an equals signal, which is on when its two inputs are equal.

1. Add labels to unlabelled wires in the diagram above, describing what data is on the line. For example, one of the outputs of the registers block could be R[rs].

2. Add control signals to the diagram below so the datapath above could execute the following instructions: add, addi, lw, beq, and j. All control signals come out of the Control Block cloud, which takes as input the current instruction. For example, to execute add, the ALU would need an input that tells it to add its two inputs, among other things.

3. Describe what happens during each stage for each of the instructions listed above.

All instructions fetched during Inst. Fetch, decoded during Decode.
add: reads R[rs] and R[rt] during Register Read, adds during Execute, writes during Reg. Write
addi: reads R[rs] during Register Read, adds during Execute, writes during Reg. Write
lw: reads R[rs] during Register Read, adds during Execute, loads from memory during Memory, writes during Reg. Write
beq: reads R[rs] and and R[rt] during Register Read, compares during Execute,



4. On the reprinted diagram above, show the modifications you would have to make to support jr. Show any new control signals you may have to create.

Add line from R[rs] to the box that outputs into PC, and add a control signal from the Control block to control when to use R[rs] instead of PC+4 or PC+imm.

5. Suppose you wanted to add a new instruction, beqr, which will be used like this:
beqr $x, $y, $z will branch to the address in $z if $x and $y are equal, otherwise continue to

the next instruction. Show any changes that would need to be made to the datapath above to make this instruction work.

The register file would need to output a R[rd]. and this would need to be an input into the block that chooses the new PC.

Bonus Question

You have 100 ants on a rope 100 units long. Each ant either moves left or right at 1 unit/sec. Ants take up no space (assume they are points), but whenever two ants collide they both instantly change directions. You may place the ants on the rope in any initial configuration - what is the maximum amount of time you can keep at least 1 ant on the rope?

Since ants colliding happens to cause the same behavior as ants just passing through each other, the answer is 100 units of time (place at least one ant at either end of the rope travelling towards the other side).