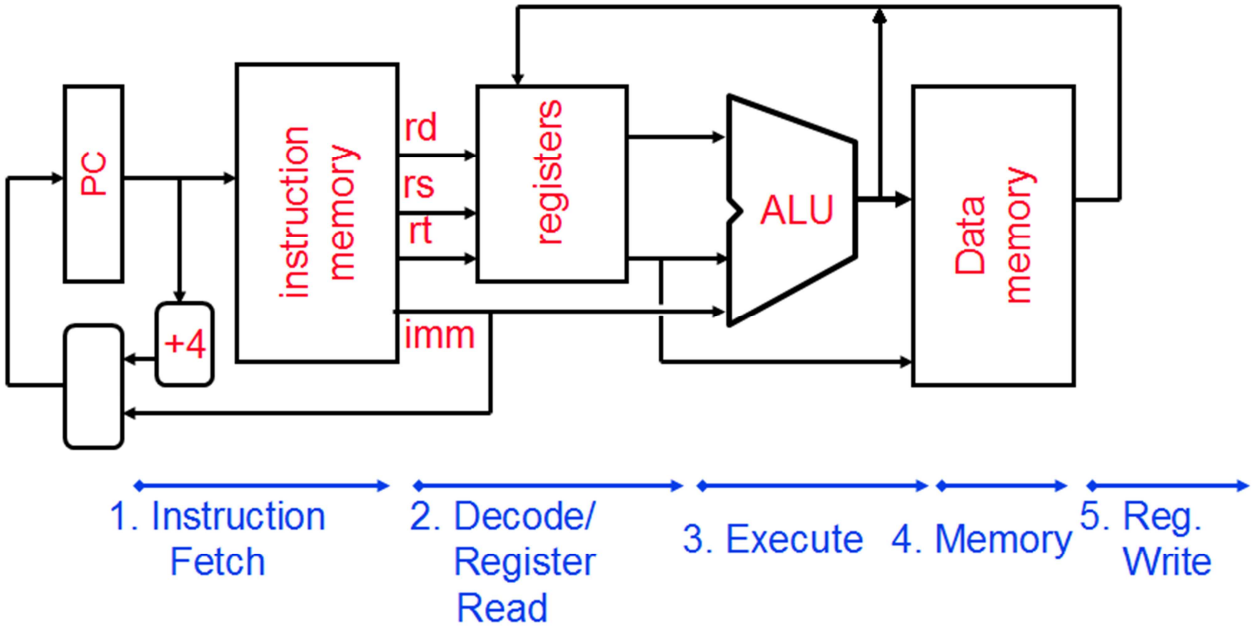
CPU Design

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)

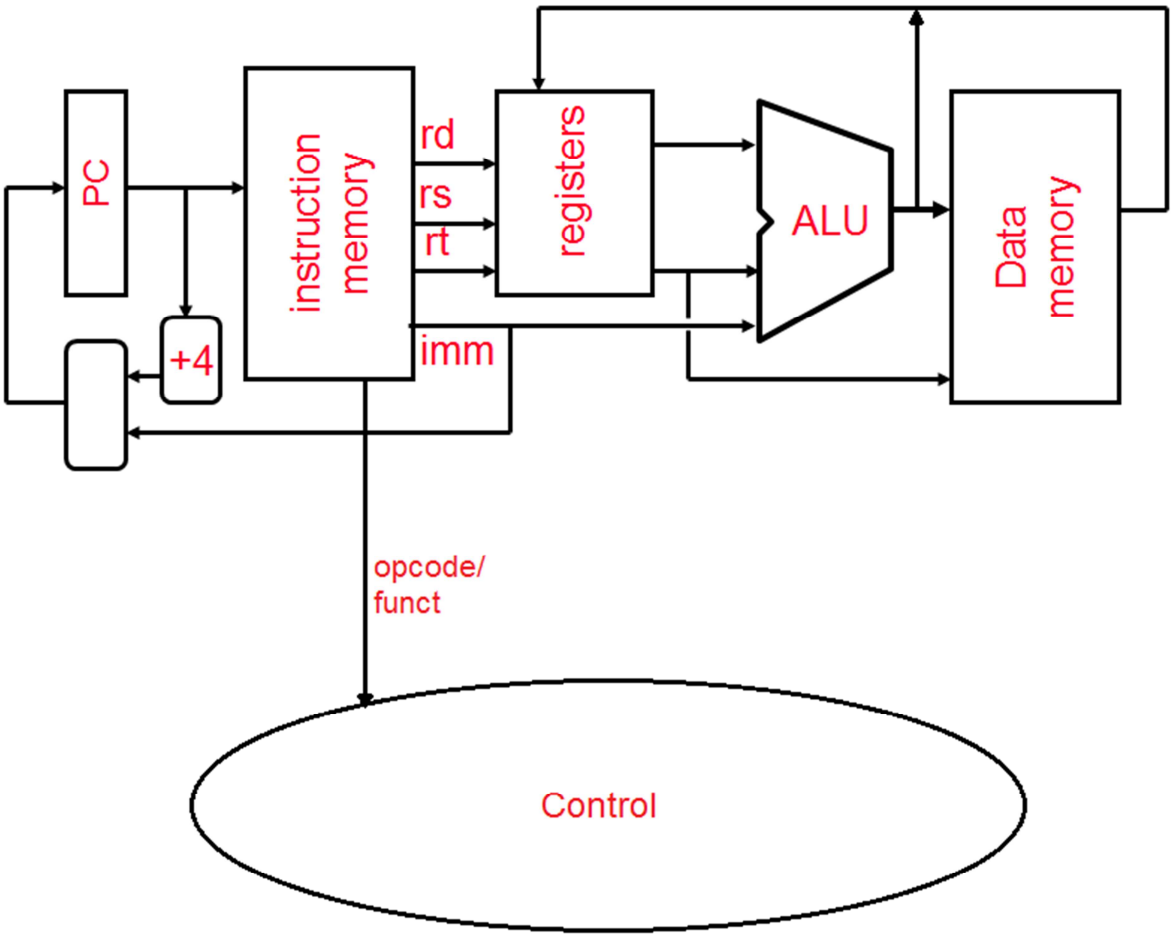
- Use to describe flow of data: `dest ← src`
- Each line happens in parallel (at the same time): `b ← c, a ← b`
- In MIPS, use `R[x]` for register `x`, and `Mem[y]` for memory at `y`. Similar to array syntax.

Exercises

Assume that the ALU can output an Equals signal, which is on when its two inputs are equal.

1. Label the unlabeled wires in the diagram *above* to describe what data is on each line. For example, one of the outputs of the registers block could be `R[rs]`.

2. Add control signals and missing elements (such as multiplexers) to the diagram *below* so that the datapath can execute the following instructions: `add`, `lui`, `sw`, `bne`, `j`.



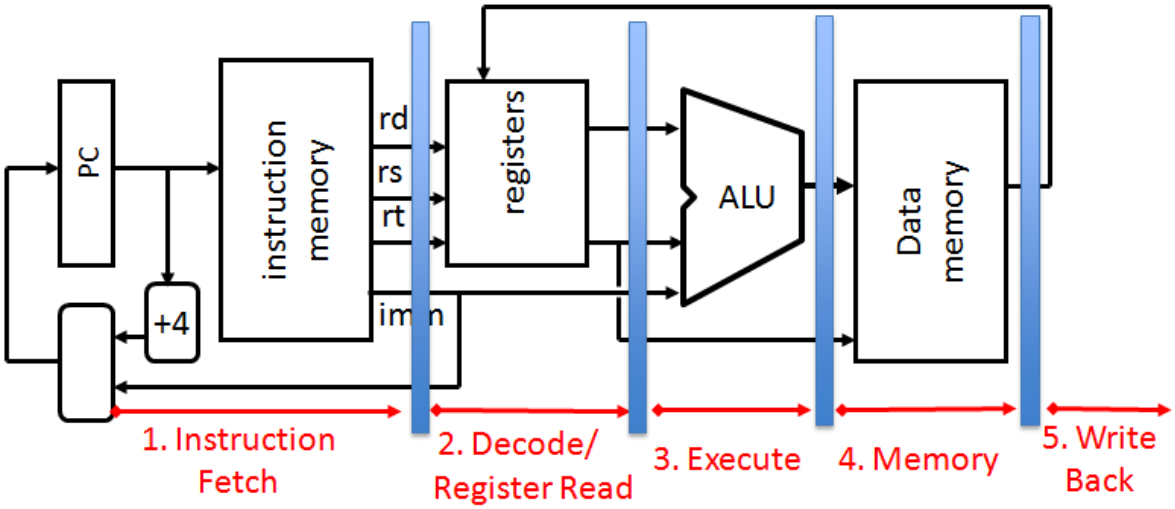
3. Fill out the values for the control signals from part 2 (write the names of your control signals in the second row):

Instr	Control Signals							
add								
lui								
sw								
bne								
j								

4. 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 Pipelined Datapath

Here is the same datapath that you've seen before but pipelined. Note that this rough diagram is not entirely correct: signals rd, rs, rt, and imm are not produced until the Decode stage. A more detailed diagram is in the textbook and in lecture slides.



What about the CPU Control: What would happen if we didn't change it and still wired each control signal directly to its destination? How do we fix this problem?

Pipelining Hazards:

Structural – Hazards due to competition for the same resource (register file read vs. write back, instruction fetch vs. data read).

How are these hazards dealt with in hardware?

Control – Hazards due to non-sequential instructions (jumps and branches).

How are these hazards dealt with in hardware?

Data – Hazards due to data dependencies (instruction requires result from earlier instruction).

How are these hazards dealt with in hardware?

Pipelining Exercises

1. Suppose you've designed a MIPS processor implementation in which the stages take the following lengths of time: IF=20ns, ID=10ns, EX=20ns, MEM=35ns, WB=10ns. What is the minimum clock period for which your processor functions properly? Where should the bulk of your R&D budget go for the next generation of processors?

2. When will there never be 5 instructions executing at the same time in your pipelined datapath? How does this affect your performance factor increase over a non-pipelined datapath?

3. Your friend tells you that his processor design is 10x better than yours, since it has 50 pipeline stages to your 5. Is he right? Why or why not? (This is intentionally vague)