State Elements

State elements provide a means of storing values and controlling data flow in a circuit. The most basic state element is a D-type Flip-Flop (figure 1a). D and Q are single bit input and output, respectively. When n D flip-flops are connected in parallel and controlled by the same clock, they form an n-bit register. To determine the input and output values of state elements, timing diagrams are often used in digital circuit analysis.

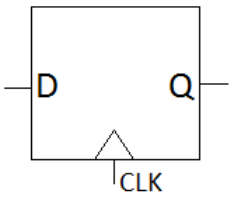
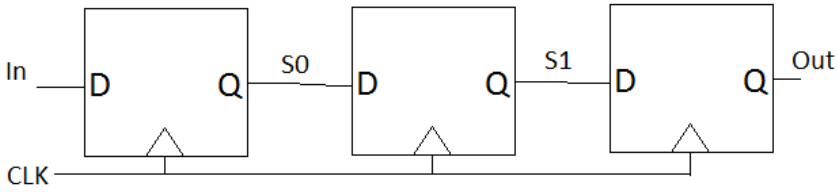


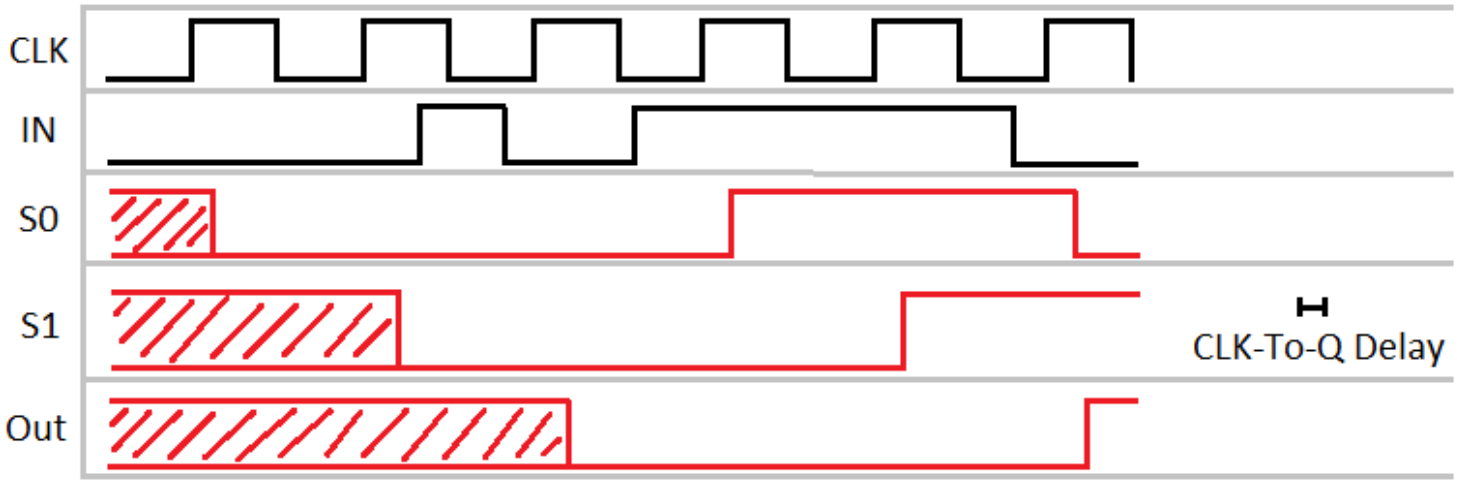
Figure 1a. D-type Flip-Flop



1b. Configuration for Q1.

Q1: Figure 1b represents three D-type flip-flips connected in series. On the **rising edge** of the clock, the value from D is copied to Q, after a small delay (known as the Clk-to-Q delay). Complete the following timing diagrams for this configuration.

Note: The shaded region in S0, S1, and Out represent the so-called "high-Z" value. That means



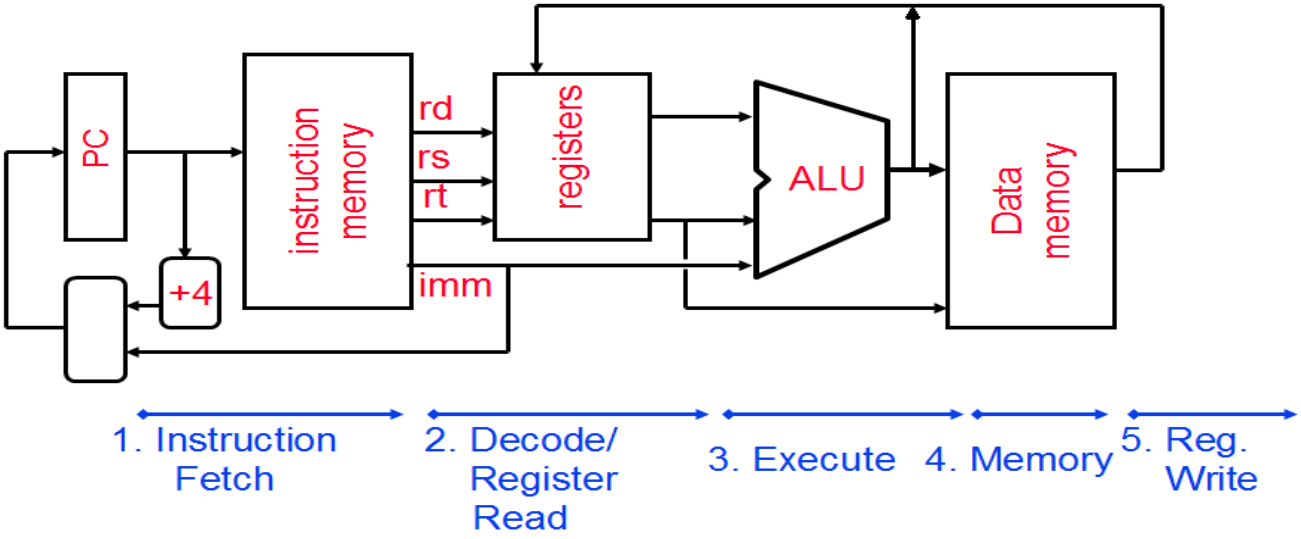
the exact values are unknown (depend on the previous states) until the defined values penetrate through the flip-flops.

Register Transfer Language

RTL is used to describe flow of data. In MIPS, we use R[x] to represent the value of register x, and Mem[y] for the value at memory location y (similar to array syntax).

Single Cycle CPU

Below is the simplified datapath for a single cycle CPU as shown in class.

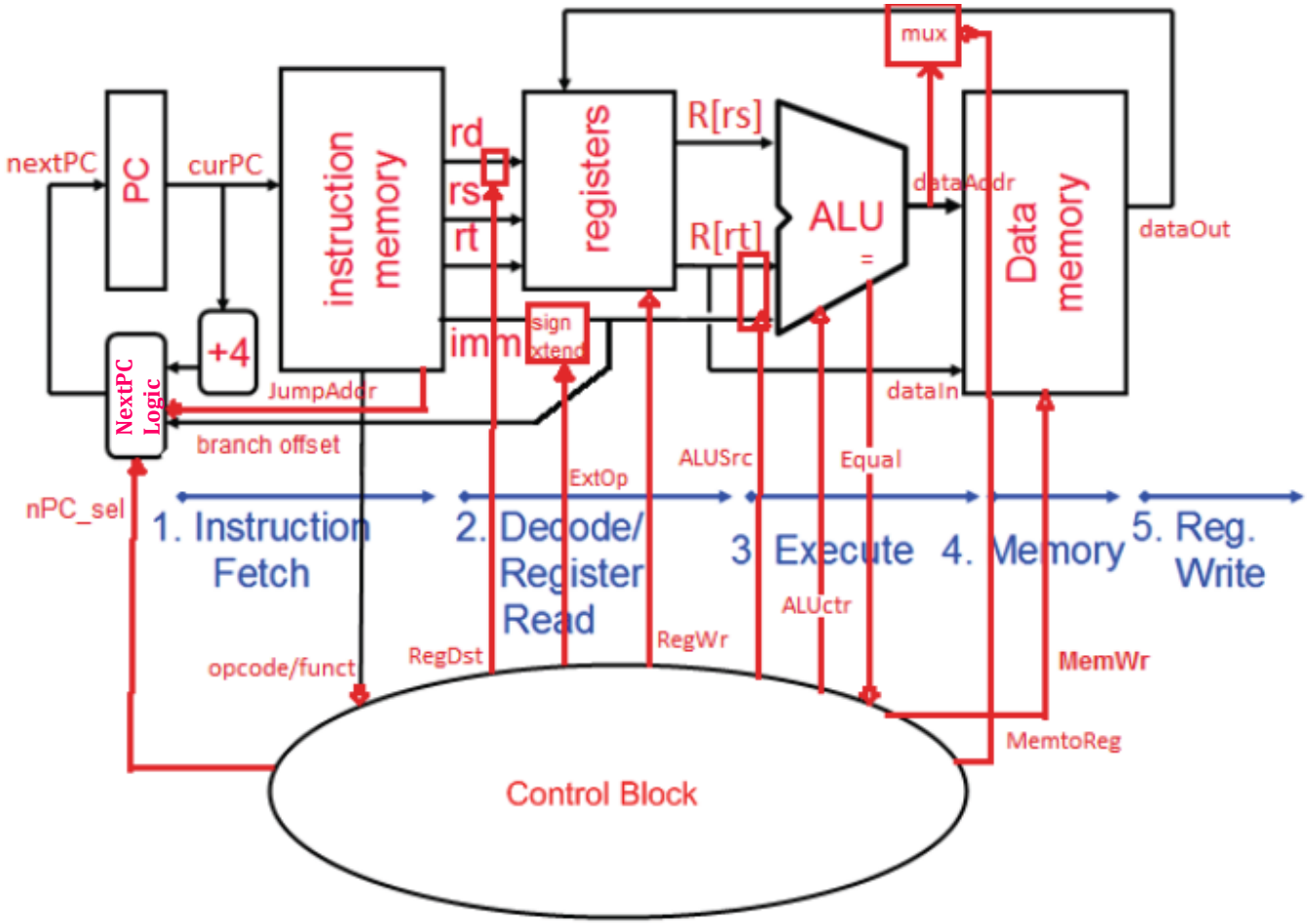


Q2: Label any unlabeled wires in the above diagram (use RTL if necessary). Labeled in Q3

Q3: We would like to modify our datapath to execute: add, ori, sw, bne, j.

Draw the new datapath, with necessary muxes and control signals. Then fill in the control table.

Assume ALU has an "=" output that is high when the two inputs are equal and low otherwise.



Instrs.	Control Signals							
	nPC_sel	RegDst	RegWr	ALUSrc	ALUctr	MemtoReg	MemWr	ExtOp
add	PC+4	rd	1	rt	add	0	0	X
ori	PC+4	rt	1	imm	or	0	0	Zero
sw	PC+4	X	0	imm	add	X	1	Sign
bne	branch	X	0	rt	X	X	0	Sign*
j	jump	X	0	X	X	X	0	X

* we assume that the calculation of the full branch address is done in the nextPC logic block; this includes multiplying the branch offset by 4, and adding the result of that to PC+4.

Note: This question covers one instruction from (almost) each of the different MIPS instruction types studied in this class. The controls for other instructions will be similar to one of the five above. Hopefully this table (and, of course, the P&H book and lecture slides) will come in handy when you dive in to project 4.

Note2: Questions like this one where you modify the given existing datapath to accommodate a new instruction is a favorite for test makers (and interviewers!). Enough said, project 4 should give you practice on this. If you feel like getting even more practice, feel free to pick instructions out of the real/commercial MIPS ISA and try to implement them.