Exercise 4.7

In this exercise we examine how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. For problems in this exercise, assume the following latencies for logic blocks in the datapath:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
a.	200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps
b.	750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

4.7.1 [10] <4.3> What is the clock cycle time if the only types of instructions we need to support are ALU instructions (`ADD`, `AND`, etc.)?

4.7.2 [10] <4.3> What is the clock cycle time if we only have to support `LW` instructions?

4.7.3 [20] <4.3> What is the clock cycle time if we must support `ADD`, `BEQ`, `LW`, and `SW` instructions?

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
a.	200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps
b.	750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

4.7.1 [10] <4.3> What is the clock cycle time if the only types of instructions we need to support are ALU instructions (ADD, AND, etc.)?

4.7.1 The longest-latency path for ALU operations is through I-Mem, Regs, Mux (to select ALU operand), ALU, and Mux (to select value for register write). Note that the only other path of interest is the PC-increment path through Add (PC + 4) and Mux, which is much shorter. So for the I-Mem, Regs, Mux, ALU, Mux path we have:

a.	200ps + 90ps + 20ps + 90ps + 20ps = 420ps
b.	750ps + 300ps + 50ps + 250ps + 50ps = 1400ps

In this exercise we examine how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. For problems in this exercise, assume the following latencies for logic blocks in the datapath:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
a.	200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps
b.	750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

4.7.2 [10] <4.3> What is the clock cycle time if we only have to support `LW` instructions?

4.7.2 The longest-latency path for `LW` is through I-Mem, Regs, Mux (to select ALU input), ALU, D-Dem, and Mux (to select what is written to register). The only other interesting paths are the PC-increment path (which is much shorter) and the path through Sign-extend unit in address computation instead of through Registers. However, Regs has a longer latency than Sign-extend, so for I-Mem, Regs, Mux, ALU, D-Mem, and Mux path we have:

a.	200ps + 90ps + 20ps + 90ps + 250ps + 20ps = 670ps
b.	750ps + 300ps + 50ps + 250ps + 500ps + 50ps = 1900ps

In this exercise we examine how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. For problems in this exercise, assume the following latencies for logic blocks in the datapath:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
a.	200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps
b.	750ps	200ps	50ps	250ps	300ps	500ps	100ps	5ps

4.7.3 [20] <4.3> What is the clock cycle time if we must support ADD, BEQ, LW, and SW instructions?

4.7.3 The answer is the same as in 4.7.2 because the LW instruction has the longest critical path. The longest path for SW is shorter by one Mux latency (no write to register), and the longest path for ADD or BNE is shorter by one D-Mem latency.

For the remaining problems in this exercise, assume that there are no pipeline stalls and that the breakdown of executed instructions is as follows:

	ADD	ADDI	NOT	BEQ	LW	SW
a.	20%	20%	0%	25%	25%	10%
b.	30%	10%	0%	10%	30%	20%

4.7.4 [10] <4.3> In what fraction of all cycles is the data memory used?

4.7.4 The data memory is used by LW and SW instructions, so the answer is:

a.	25% + 10% = 35%
b.	30% + 20% = 50%

For the remaining problems in this exercise, assume that there are no pipeline stalls and that the breakdown of executed instructions is as follows:

	ADD	ADDI	NOT	BEQ	LW	SW
a.	20%	20%	0%	25%	25%	10%
b.	30%	10%	0%	10%	30%	20%

4.7.5 [10] <4.3> In what fraction of all cycles is the input of the sign-extend circuit needed? What is this circuit doing in cycles in which its input is not needed?

4.7.5 The sign-extend circuit is actually computing a result in every cycle, but its output is ignored for `ADD` and `NOT` instructions. The input of the sign-extend circuit is needed for `ADDI` (to provide the immediate ALU operand), `BEQ` (to provide the PC-relative offset), and `LW` and `SW` (to provide the offset used in addressing memory) so the answer is:

a.	20% + 25% + 25% + 10% = 80%
b.	10% + 10% + 30% + 20% = 70%

For the remaining problems in this exercise, assume that there are no pipeline stalls and that the breakdown of executed instructions is as follows:

	ADD	ADDI	NOT	BEQ	LW	SW
a.	20%	20%	0%	25%	25%	10%
b.	30%	10%	0%	10%	30%	20%

4.7.6 [10] <4.3> If we can improve the latency of one of the given datapath components by 10%, which component should it be? What is the speedup from this improvement?

4.7.6 The clock cycle time is determined by the critical path for the instruction that has the longest critical path. This is the `LW` instruction, and its critical path goes through I-Mem, Regs, Mux, ALU, D-Mem, and Mux so we have:

a.	D-Mem has the longest latency, so we reduce its latency from 250ps to 225ps, making the clock cycle 25ps shorter. The speedup achieved by reducing the clock cycle time is then 670ps/645ps = 1.039.
b.	I-Mem has the longest latency, so we reduce its latency from 750ps to 675ps, making the clock cycle 75ps shorter. The speedup achieved by reducing the clock cycle time is then 1900ps/1825ps = 1.041.