Exercise 4.2

The basic single-cycle MIPS implementation in Figure 4.2 can only implement some instructions. New instructions can be added to an existing ISA, but the decision whether or not to do that depends, among other things, on the cost and complexity such an addition introduces into the processor datapath and control. The first three problems in this exercise refer to this new instruction:

	Instruction	Interpretation
a.	`SEQ Rd,Rs,Rt`	`Reg[Rd]` = Boolean value (0 or 1) of (`Reg[Rs]` == `Reg[Rs]`)
b.	`LWI Rt,Rd(Rs)`	`Reg[Rt] = Mem[Reg[Rd]+Reg[Rs]]`

4.2.1 [10] <4.1> Which existing blocks (if any) can be used for this instruction?

4.2.2 [10] <4.1> Which new functional blocks (if any) do we need for this instruction?

4.2.3 [10] <4.1> What new signals do we need (if any) from the control unit to support this instruction?

Solution 4.2

4.2.1 Existing blocks that can be used for this instruction are:

a.	This instruction uses instruction memory, both existing read ports of Registers, the ALU (to compare Rs and Rt), and the write port of Registers.
b.	This instruction uses instruction memory, both register read ports, the ALU to add Rd and Rs together, data memory, and the write port in Registers.

4.2.2 New functional blocks needed for this instruction are:

a.	This instruction needs the Zero output of the ALU to be zero-extended to compute the value for Rd. Then we need to add this as another input to the Mux that selects the value to be written into Registers.
b.	None. This instruction can be implemented using existing blocks.

4.2.3 The new control signals are:

a.	We need a new control signal for the Mux that selects between values that can be written into Registers.
b.	None. This instruction can be implemented without adding new control signals. It only requires changes in the Control logic.

4.2.4 Clock cycle time is determined by the critical path, which for the given latencies happens to be to get the data value for the load instruction: I-Mem (read instruction), Regs (takes longer than Control), Mux (select ALU input), ALU, Data Memory, and Mux (select value from memory to be written into Registers). The latency of this path is 400ps + 200ps + 30ps + 120ps + 350ps + 30ps = 1130ps.

When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are starting with a datapath from Figure 4.2, where I-Mem, Add, Mux, ALU, Regs, D-Mem, and Control blocks have latencies of 400ps, 100ps, 30ps, 120ps, 200ps, 350ps, and 100ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively. The remaining three problems in this exercise refer to the following processor improvement:

	Improvement	Latency	Cost	Benefit
a.	Add Multiplier to ALU	+300ps for ALU	+600 for ALU	Lets us add MUL instruction. Allows us to execute 5% fewer instructions (MUL no longer emulated).
b.	Simpler Control	+100ps for Control	–400 for Control	Control becomes slower but cheaper logic.

4.2.4 [10] <4.1> What is the clock cycle time with and without this improvement?

4.2.5 [10] <4.1> What is the speedup achieved by adding this improvement?

4.2.6 [10] <4.1> Compare the cost/performance ratio with and without this improvement.

	New Clock Cycle Time
a.	1430ps (1130ps + 300ps, ALU is on the critical path)
b.	1130ps. Control latency is now equal to Regs latency, so we have a second critical path (same as the existing one, but going through Control to generate the control signal for the Mux that selects second ALU input). This new critical path has the same latency as the existing one, so the clock cycle is unchanged.

4.2.5 The speedup comes from changes in clock cycle time and changes to the number of clock cycles we need for the program:

	Benefit
a.	We need 5% fewer cycles for a program, but cycle time is 1430 instead of 1130, so we have a speedup of (1/0.95) × (1130/1430) = 0.83, which means we actually have a slowdown.
b.	Speedup is 1 (no change in number of cycles, no change in clock cycle time).

4.2.6 The cost is always the total cost of all components (not just those on the critical path, so the original processor has a cost of I-Mem, Regs, Control, ALU, D-Mem, 2 Add units, and 3 Mux units, for a total cost of $1000 + 200 + 500 + 100 + 2000 + 2 \times 30 + 3 \times 10 = 3890$.

We will compute cost relative to this baseline. The performance relative to this baseline is the speedup we computed in 4.2.5, and our cost/performance relative to the baseline is as follows:

	New Cost	Relative Cost	Cost/Performance
a.	3890 + 600 = 4490	4490/3890 = 1.15	1.15/0.83 = 1.39. We are paying significantly more for significantly worse performance, so the cost/performance is a lot worse than with the unmodified processor.
b.	3890 – 400 = 3490	3490/3890 = 0.9	0.9/1 = 0.9. We are reducing cost and getting the same performance, so the cost/performance improves.