Q 4.10

10.1)

let n be the number of instructions the CPU completes

ET_0 = execution time before improvement $= n(250 + 150 + 25 + 200 + 250 + 30 + 25 + 20) = 950 \cdot n_0$

ET_1 = execution time after improvement $= n(250 + 160 + 25 + 200 + 250 + 30) = 960 \cdot n_1$

The speedup is attributed to the reduction of loads and stores by 12%, in due to the increase of registers,

that is,

$$\Delta n = (.12 \cdot .25) + (.12 \cdot 0.1)$$
$$= 0.03 + 0.012$$
$$= 0.042 \quad \Rightarrow$$

T_0 takes n instructions

T_1 takes $n_1 = n_0(1 - 0.042) = 0.958 n_0$

hence, number of instructions decreases by 0.958n which attributes a speedup of

$$\frac{ET_0}{ET_1} = \frac{950 \cdot n_0}{960 \cdot n_1}$$
$$= \frac{950\, n_0}{960(0.958 \cdot n_0)}$$
$$= 1.032$$

speedup of 1.032

10.2)

C_0 = cost before improvement $= 1000 + 200 + 10 + 100 + 30 + 2000 + 5 + 100 + 1 + 500 = 3946$

C_1 = cost after improvement $= 1000 + 400 + 10 + 100 + 30 + 2000 + 5 + 100 + 1 + 500 = 4146$

$$\Delta C = \frac{\left(\frac{3940}{950}\right)}{\left(\frac{4140}{960}\right)} = 1.04 \quad \Rightarrow$$

4% increase in cost

10.2 continued)

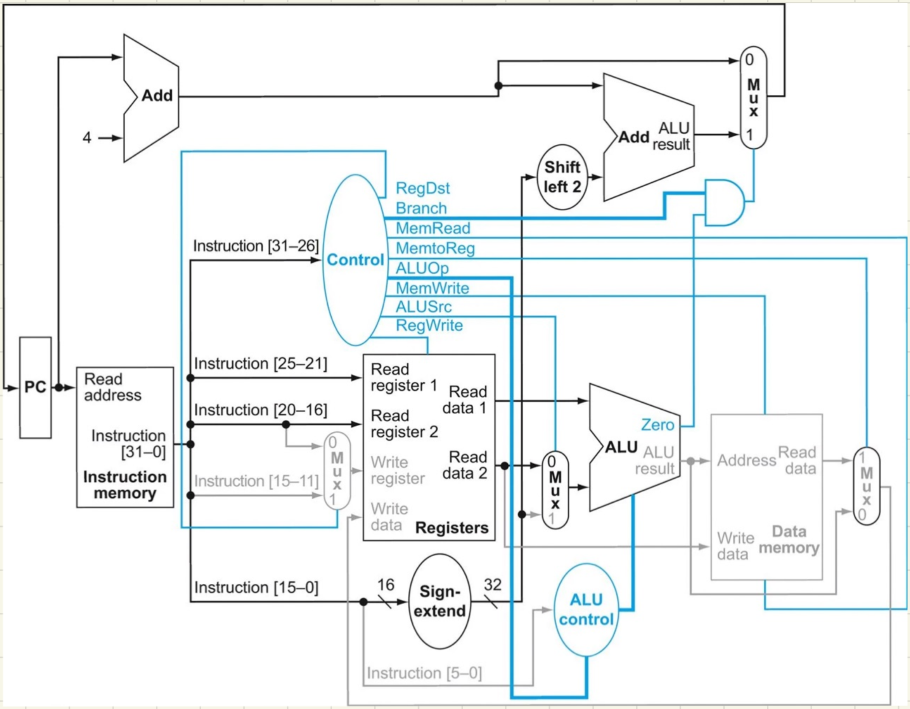
$$P_0 = \text{performance before improvement} = \frac{1}{\text{Latency}} = \frac{1}{950 \cdot n_0 \cdot 10^{-12}}$$

$$P_1 = \text{performance after improvement} = \frac{1}{\text{Latency}} \approx \frac{1}{960 \cdot n_0 \cdot 0.958 \cdot 10^{-12}}$$

$$\Delta\text{performance} = \frac{\left(\frac{1}{960 \cdot n_0 \cdot 0.958 \cdot 10^{-12}}\right)}{\left(\frac{1}{950 \cdot n_0 \cdot 10^{-12}}\right)} = 1.03 \Rightarrow \text{3\% increase in performance}$$

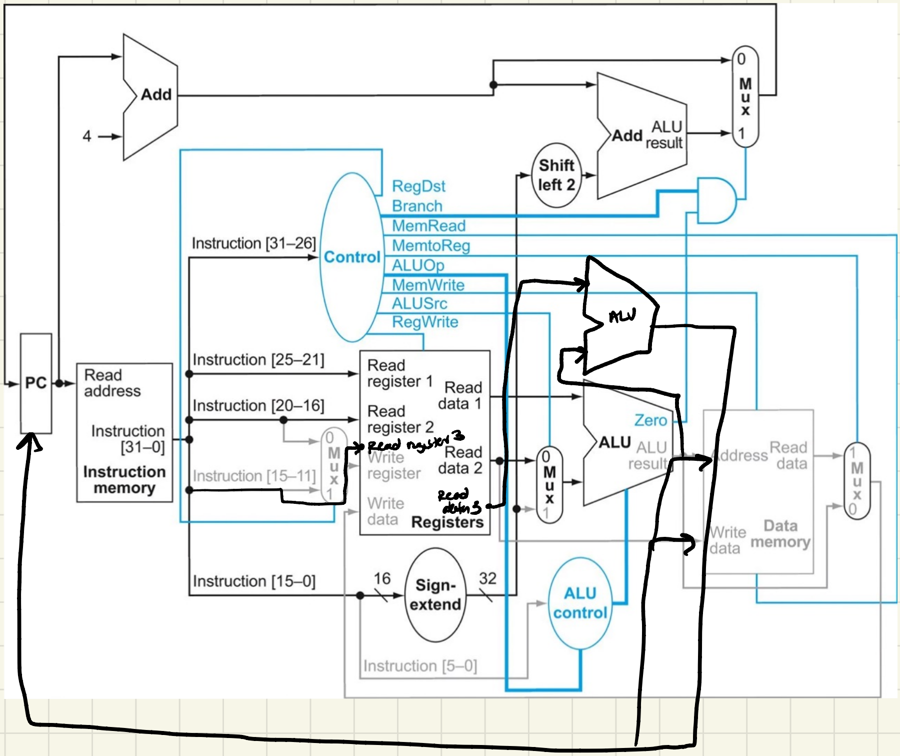
10.3) If the performance increase is needed for the optimization of some critical software, it would make sense despite the greater increase in cost. However, if the optimization does not provide to the software in a substantial way, the cost would not be desired.

Q 4.11



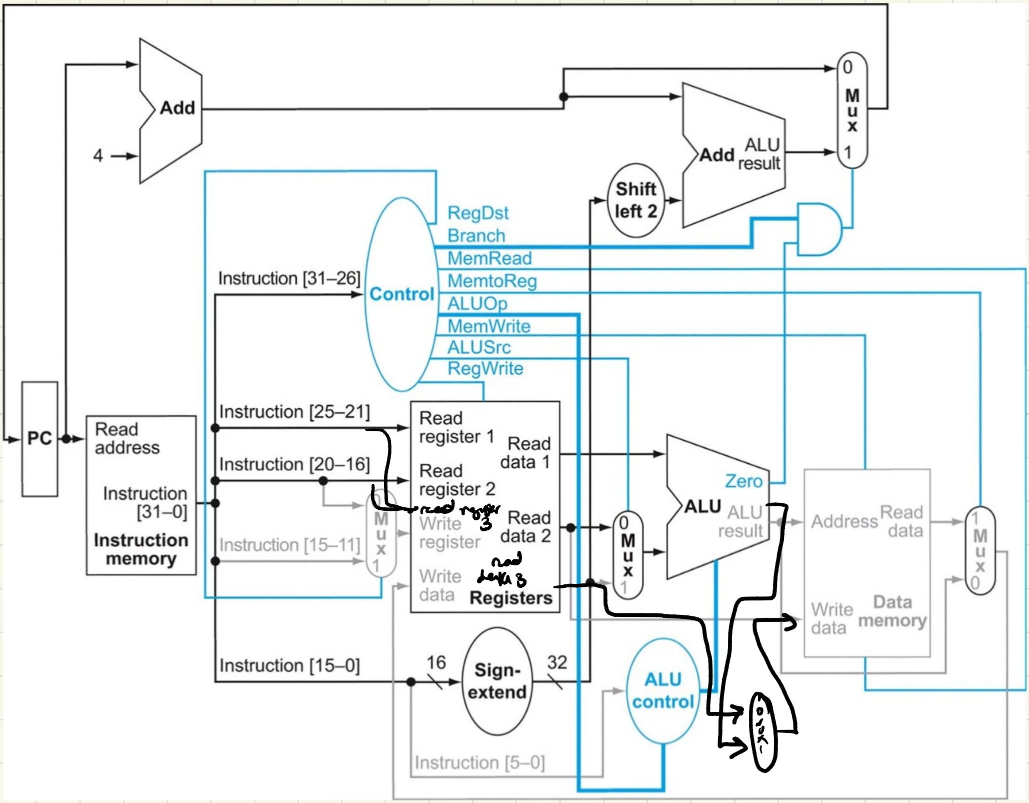
No signals are required to be added – the control manages the read registers and increments as needed. Therefore, the diagram naturally has load with increment instructions.

Q 4.12



Read port is added to read Rx, and a second ALU to compute the add.

Q 4.13



Read data from read data 1 and place into ALU and write into data. To do this we add another register block to store the add, and inputted into the MUX to modify the data path for the storage of the adds.