Question 2 (18 points): For a benchmark program executing in the Nindle e-reader 20% of the instructions are load/store, 50% of the instructions are ALU operations and 30% of the instructions are branches. On average load/store instructions take 10 cycles to execute, ALU instructions execute in 1 cycle and branch instructions take 3 cycles to execute. The clock frequency for this processor is 4 GHz (1 GHz = 10^9 Hz). This benchmark takes 20 seconds to execute.

a. (**4 points**) What is the average number of clocks per instruction (CPI) for this benchmark?

$$\text{CPI} = 0.2 \times 10 + 0.5 \times 1 + 0.3 \times 3 = 2.0 + 0.5 + 0.9 = 3.4 \frac{\text{cycles}}{\text{instruction}}$$

b. (**5 points**) How many instructions are executed by this benchmark?

$$\begin{aligned}
\text{Execution Time} &= \#\text{ instructions} \times CPI \times \frac{1}{\text{frequency}} \\
\#\text{ instructions} &= \frac{\text{Execution Time} \times \text{frequency}}{CPI} \\
\#\text{ instructions} &= \frac{20\ seconds \times 4 \times 10^9 \frac{\text{cycles}}{seconds}}{3.4 \frac{\text{cycles}}{\text{instruction}}} = 23.53 \times 10^9 \text{ instructions}
\end{aligned}$$

c. (**6 points**) A revision of the architecture for the Nindle processor adds a new level to the memory hierarchy and thus reduces the average execution time of each load/store instruction to 5 cycles. Also an improvement to the compiler reduces the number of load/store instructions required to execute this benchmark by half. How much time does it take to execute the same benchmark in this revised Nindle processor?

First lets compute the time spent in each instruction type in the original Nindle processor for this benchmark

$$\begin{aligned}
\text{Time}_{\text{load/store,original}} &= \frac{0.2 \times 10}{3.4} \times 20 = 11.76\ seconds \\
\text{Time}_{\text{ALU,original}} &= \frac{0.3 \times 1}{3.4} \times 20 = 2.94\ seconds \\
\text{Time}_{\text{branches,original}} &= \frac{0.5 \times 3}{3.4} \times 20 = 5.29\ seconds
\end{aligned} \tag{1}$$

The time spent on load/store instructions and on branch instructions has not changed because the improvement has been only to load/store instructions.

There are half as many load/store instructions and, on average, each requires half as many circles. Thus, the time spent executing load/store instructions is reduced by a factor of 4. The new execution time is:

$$\text{Time}_{\text{revised}} = \text{Time}_{\text{load/store,revised}} + \text{Time}_{\text{ALU,revised}} + \text{Time}_{\text{branches,revised}}$$
$$\text{Time}_{\text{revised}} = \frac{\text{Time}_{\text{load/store,original}}}{4} + \text{Time}_{\text{ALU,original}} + \text{Time}_{\text{branches,original}}$$
$$\text{Time}_{\text{revised}} = \frac{11.76}{4} + 2.94 + 5.29 = 11.17 \ \textit{seconds}$$

An alternative solution computes the CPI of the revised machine. The revised machine executes 10% fewer instructions. Thus the new CPI is given by:

$$\begin{aligned}
\text{CPI}_{\text{revised}} &= \frac{01 \times 5 + 0.5 \times 1 + 0.3 \times 3}{0.9} = 2.11 \frac{\textit{cycles}}{\textit{instruction}} \\
\text{Time}_{\text{revised}} &= \frac{2.11 \times 0.9 \times 23.53 \times 10^9}{4 \times 10^9} = 11.17 \ \textit{seconds}
\end{aligned}$$

d. (**3 points**) How much faster is this benchmark in the improved Nindle (with both the revised architecture and the improved compiler) in comparison with the original Nindle?

$$\frac{\text{Time}_{\text{original}}}{\text{Time}_{\text{revised}}} = = \frac{20 \ \textit{seconds}}{11.17 \ \textit{seconds}} = 1.79 \text{ times faster}$$

Alternatively:

$$\frac{\text{Time}_{\text{original}} - \text{Time}_{\text{revised}}}{\text{Time}_{\text{original}}} \times 100 = \frac{20 - 11.17}{20} \times 100 = 44.2\% \text{ faster}$$