Homework 1

Shenyao Jin

shenyaojin@mines.edu

January 27, 2025

1 Problem 1

Problem: Compute the Clocks Per Instruction (CPI) of a machine, which has an average CPI for ALU operations of 1.1, a CPI for branches/jumps of 3.0, and a hit rate of 60% in the cache. A hit in the cache takes 1 cycle and a cache miss takes 120 cycles. Assume 22% of instructions are loads, 12% are stores, 20% are branches/jumps, and the rest are ALU operations.

Solution:

For the memory instructions, the average CPI is given by

$$\text{CPI}_{\text{memory}} = (1 - \text{hit rate}) \times \text{miss penalty} + \text{hit rate} \times \text{hit time} = (1 - 0.6) \times 120 + 0.6 \times 1 = 48.6 \tag{1}$$

So the average CPI for the machine is: x

$$\text{CPI}_{avg} = 0.22 \times 48.6 + 0.12 \times 48.6 + 0.2 \times 3 + 0.46 \times 1.1 = 17.63 \tag{2}$$

Therefore, the average CPI of the machine is 17.63.

2 Problem 2

Problem: You are a processor designer and have to make a decision between building a processor, which executes at 1GHz and has an average CPI 1.2 and a processor, which executes at 2GHz but has a CPI of 2. Which is better to build and why?

Solution:

We assume the number of instructions executed is the same for both processors, which is marked as N. The execution time for the first processor is

$$T_1 = \frac{N}{\text{Frequency}_1} \times \text{CPI}_1 = \frac{N}{1 \times 10^9} \times 1.2 \tag{3}$$

And the execution time for the second processor is:

$$T_2 = \frac{N}{\text{Frequency}_2} \times \text{CPI}_2 = \frac{N}{2 \times 10^9} \times 2 \tag{4}$$

Therefore, $T_1 = 1.2 \times 10^{-9} N$ and $T_2 = 1 \times 10^{-9} N$. Since $T_1 > T_2$, the second processor is better to build.

3 Problem 3

Problem: A revolutionary new technology in memory improves your memory subsystem so that memory latencies are reduced by a factor of 3.5. After replacing your memory with the new ones, you observe that you now spend half your time waiting for memory. What percentage of the original execution (with the older memory system) was spent waiting for memory?

Solution:

For the memory:

$$T_{memory,new} = \frac{T_{memory,old}}{3.5} \tag{5}$$

And the total time spent waiting for memory is:

$$\frac{T_{memory,new}}{T_{total,new}} = \frac{T_{memory,new}}{T_{memory,new} + T_{non-memory,new}} = \frac{1}{2} \tag{6}$$

So the non-memory time is:

$$T_{non-memory,new} = T_{memory,new} = \frac{T_{memory,old}}{3.5} \tag{7}$$

Therefore, the percentage of the original execution time spent waiting for memory is:

$$\frac{T_{memory,old}}{T_{total,old}} = \frac{T_{memory,old}}{T_{memory,old} + T_{non-memory,old}} = \frac{T_{memory,new} \times 3.5}{T_{memory,new} \times 3.5 + T_{non-memory,new}} = \frac{3.5}{3.5+1} = 77.8\% \tag{8}$$

Therefore, 77.8% of the original execution time was spent waiting for memory.

4 Problem 4

Problem: You're currently using a single core machine but you want to figure out if it's worth investing in a dual-core machine. Assuming your application is 60% parallelizable, by how much could you decrease the frequency and get the same performance?

Solution:

Use Amdahl's Law, the speedup of the dual-core machine is:

$$S = \frac{1}{(1-P) + \frac{P}{N}} \tag{9}$$

Here, $P = 0.6$ and $N = 2$. So the speedup is:

$$S = \frac{1}{(1-0.6) + \frac{0.6}{2}} = \frac{1}{0.7} = 1.43 \tag{10}$$

This means that, compared to the single-core machine, the dual-core machine is 1.43 times faster.

If we want the same performance as the single-core machine, we can scale the dual-core frequency down by a factor of $1/1.43 = 0.7$. In other words, we can reduce the frequency to 0.7 times the original frequency and get the same performance.