Pipeline Performance Homework

Troy Veldhuizen

1. In theory, what is the maximum speedup you can possibly achieve by converting the book's single-cycle CPU into a 5-stage pipelined CPU? Assume

First calculate the new p' speed.

$$\begin{aligned} t' &= n * p * cpi \\ p' &= 925/5 \\ p' &= 185 + 50 \\ p' &= 235 \end{aligned}$$

Calculate speedup

$$\begin{aligned} speedup &= \frac{oldtime}{newtime} \\ speedup &= \frac{n * p}{n * p' * cpi} \\ speedup &= \frac{925 * n}{235 * n * 1} \\ speedup &= \frac{925}{235} \\ speedup &= 3.94 \end{aligned}$$

2. In theory, what is the maximum speedup you can possibly achieve by converting the book's single-cycle CPU into a pipelined CPU of any depth? Assume

First find general equation for a cycle speedup

$$speedup = \frac{925}{\frac{925}{k} + 50}$$

Then find the speedup as k approaches ∞

$$\begin{aligned} speedup &= \frac{925}{\frac{925}{\infty} + 50} \\ speedup &= \frac{925}{0 + 50} \\ speedup &= \frac{925}{50} \\ speedup &= 18.5 \end{aligned}$$

The maximum theoretical speedup is 18.5.

3. In addition to the assumptions given in Problem 2, also assume that the CPI for a 5-stage pipeline is 1.035 and that the number of stalls increases by 50% with every additional stage. (In other words, assume that the CPI for a 6-stage pipeline is 1+ (.035)(1.5), the CPI for a 7-stage pipeline is 1 + (.035)(1.5)(1.5), etc.)

a. Develop a formula for the performance of this k-stage pipeline. Your formula should combine the cycle time formula from Problem 2, with the model for stalls discussed above.

First find the *cpi* adjusted for stalls

$$cpi = 1 + 0.035(1.5^{k-5})$$

Total speed formula

$$t = (\frac{925}{k} + 50) * n * (1 + 0.035(1.5^{k-5}))$$

b. Use your formula to determine the optimal number of pipeline stages. (The math for this problem gets complex. I suggest using gnuplot, Maple, or Wolframalpha.com to graph the solution and make a close visual guess.)

After plotting the equation, the minimum time was when k was 9.418, so roughly 9 or 10 pipeline stages will give you the optimal performance assuming the above parameters.

4. You are attempting to improve a pipelined CPU's performance by splitting the data memory stage (the slowest stage) into two stages, thereby increasing the clock speed. However, splitting data memory into two stages introduces the possibility of some lw instructions generating two stalls instead of one.

a. Explain how splitting the data memory section into two stages might result in extra stalls.

The objective of forwarding is to get the correct input into the ALU in the execute stage. In the five stage pipeline outlined in the book, the memory stage is in between the execute and the write stage. This causes the need for one stall. If two memory stages are between the write and the execute stage, then we would need two stalls.

b. Will sw instructions generate any additional stalls? Why or why not?

It depends if the machine is in a state capable to start on the first stage of reading from memory after the first stage of writing is completed. If writing and reading from memory is analogous to hanging drywall and painting the drywall. If half of a new house has drywall installed, it is plausible that one could begin painting that half of the house after only half of the house has drywall. The sw instruction would not need any stalls, because the first stage store effectively "prepares" for next read. But if the memory is implemented in a way in which read and write are atomic procedures that do not have any

usable intermediate stage, a stall would be needed to allow a store to complete before the next memory operation.

c. On average, how many stalls can each lw generate and still have the new CPU outperform the old CPU?

$$\begin{aligned} 600 * n * (1 + (0.25)(0.15)) &= 580 * n * (1 + (0.25)(stall)) \\ 622.5n &= 580n * (1 + (0.25)(stall)) \\ 1.0733 &= 1 + (0.25)(stall) \\ 0.0733 &= (0.25)(stall) \\ 0.293 &= stall \end{aligned}$$

Roughly 29.3% of lw instructions can be stalls before the new CPU will perform worse than the old CPU. So on average, each lw can have 0.293 stalls.

d. At most, what percentage of lw instructions can generate a stall before the new CPU fails to outperform the old CPU?

$$\begin{aligned} 600 * n * (1 + (0.25)(0.15)) &= 580 * n * (1 + (0.25)(stall)) \\ 622.5n &= 580n * (1 + (0.25)(stall)) \\ 1.0733 &= 1 + (0.25)(stall) \\ 0.0733 &= (0.25)(stall) \\ 0.293 &= stall \end{aligned}$$

Roughly 29.3% of lw instructions can be stalls before the new CPU will perform worse than the old CPU.

e. At most, what percentage of lw instructions can generate two stalls before the new CPU fails to outperform the old CPU?

$$\begin{aligned} 600 * n * (1 + (0.25)(0.15)) &= 580 * n * (1 + (2)(0.25)(stall)) \\ 622.5n &= 580n * (1 + (2)(0.25)(stall)) \\ 1.0733 &= 1 + (2)(0.25)(stall) \\ 0.0733 &= (2)(0.25)(stall) \\ 0.037 &= stall \end{aligned}$$

Roughly 3.7% of of the lw instructions can generate two stalls before the new CPU fails to outperform the old CPU.

f. Suppose that lw instructions generate one stall three times as often as they generate two stalls. What percentage of lw instructions can generate one stall before the new CPU fails to outperform the

old CPU?

$$
\begin{aligned}
600 * n * (1 + (0.25)(0.15)) &= 580 * n * (1 + (2)(0.25)(0.66)(stall) + (1)(0.25)(0.33)(stall) \\
622.5n &= 580n * (1 + (2)(0.25)(0.33)(stall) + (1)(0.25)(0.66)(stall)) \\
1.0733 &= 1 + (2)(0.25)(0.33)(stall) + (1)(0.25)(0.66)(stall)) \\
0.0733 &= (2)(0.25)(0.33)(stall) + (1)(0.25)(0.66)(stall) \\
0.0733 &= (0.165)(stall) + (0.165)(stall) \\
0.0733 &= (0.33)(stall) \\
0.222 &= stall
\end{aligned}
$$

Now we need to find the actual percent of the one stall lw instructions

$$0.66 * 0.222 = 0.14$$

14% of lw instructions can generate one stall before the new CPU fails to outperform the old CPU.

The following is to check my answer:

$$
\begin{aligned}
600 * n * (1 + (0.25)(0.15)) &= 580 * n * (1 + (2)(0.25)(0.0732)) + (1)(0.25)(0.147) \\
622.5n &= 580 * n * (1 + (0.0366) + (0.0367)) \\
622.5n &= 580 * n * (1.0733) \\
622.5n &= 622.5n
\end{aligned}
$$

Everything looks good!