Homework 3

Shenyao Jin

shenyaojin@mines.edu

April 8, 2025

Problem 1

Q1: Describe a T-NT (Taken–Not Taken) pattern for two branches where aliasing results in negative interference.

Answer: Suppose branch A always follows the pattern Taken–Taken (T–T), while branch B always follows the pattern Not Taken–Not Taken (NT–NT). If both branches map (alias) to the same predictor entry, the predictor's state is constantly being shifted in opposite directions (T vs. NT), degrading its accuracy. As a simple example, if the 2-bit predictor saturates at 3 ("strongly taken") after seeing repeated T from branch A, then branch B (which is actually NT) will incorrectly shift it toward 2 ("weakly taken") and continue to push it toward 0 ("strongly not taken"), causing both branches to be mispredicted more often.

Q2: Describe a T-NT (Taken–Not Taken) pattern for two branches where aliasing results in positive interference.

Answer: Suppose branch A's pattern is T–T and branch B's pattern is T–T as well (or some pattern that ultimately leads the predictor to favor the same direction, e.g., T–NT but both converge to "taken" in the predictor's finite state machine). When both map to the same entry, any history from either branch encourages the predictor to move in the same direction (usually "taken"), reinforcing correct predictions for both branches.

Q3: Why does aliasing usually result in negative interference?

Answer: Most of the time, distinct branches have different outcomes. When they alias to the same predictor state, one branch's usage can overwrite or distort the history needed for the other branch's correct prediction. Because it is more common that two unrelated branches do not share exactly the same behavior, aliasing usually shifts the predictor state away from one branch's correct pattern, causing negative interference.

Problem 2

Assume a machine with a typical MIPS 5-stage pipeline that uses branch prediction **without** branch delay slots and has a branch misprediction penalty of 3 cycles. For a certain program:

- 1 in every 5 instructions is a branch.
- 80% of these branches are predicted correctly.

(a) How many cycles would it take to execute n instructions?

(b) Now imagine we have a Pentium 4 with a 20-stage pipeline, where the branch misprediction penalty is 19 cycles. To have the same performance as in part (a), what must the new branch prediction accuracy be?

Solution

(a) MIPS Pipeline

1. **Base CPI (ideal):** Each instruction completes in 1 cycle in a 5-stage pipeline (ignoring stalls). Executing n instructions ideally takes n cycles.
2. **Branch Frequency:** 1 in every 5 instructions is a branch, so out of n instructions, $\frac{n}{5}$ are branches.
3. **Branch Prediction Accuracy:** 80% of branches are correct, thus 20% are mispredicted.
4. **Misprediction Penalty:** Each misprediction costs 3 extra cycles.
5. **Extra Cycles from Mispredictions:**

$$\text{Mispredicted branches} = \frac{n}{5} \times 0.20 = 0.04n.$$

$$\text{Total penalty} = 0.04n \times 3 = 0.12n.$$

6. **Total Cycles:**

$$\text{Base} + \text{Penalty} = n + 0.12n = 1.12n.$$

Hence, it takes $\boxed{1.12n}$ cycles to execute n instructions.

(b) Pentium 4 Pipeline

1. **Branch Misprediction Penalty:** With a 20-stage pipeline, the penalty is now 19 cycles.
2. **Required Performance Match:** We want the Pentium 4 to have the same total cycle count as part (a), i.e. $1.12n$.
3. **Set Up the Equation:**

$$\text{Total Cycles} = n + (\tfrac{n}{5} \times (1-p)) \times 19 = n\Big[1 + \tfrac{19}{5}(1-p)\Big],$$

where p is the new branch prediction accuracy ($0 \le p \le 1$).

4. **Equate to $1.12n$:**

$$n\Big[1 + \tfrac{19}{5}(1-p)\Big] = 1.12n.$$

Cancel n on both sides:

$$1 + \frac{19}{5}(1-p) = 1.12.$$

5. **Solve for p:**

$$\frac{19}{5}(1-p) = 0.12 \quad \Longrightarrow \quad 1-p = \frac{0.12 \times 5}{19} \approx 0.03158 \quad \Longrightarrow \quad p = 1 - 0.03158 \approx 0.96842.$$

Thus, the required prediction accuracy is about $\boxed{96.8\%}$.