EECE 7352: Computer Architecture
Quiz 4

Assume our traditional 5-stage pipeline (single instruction issue, in-order pipeline) with multi-port register file and memory. You can assume that bypassing is always enabled, and the register file is written in the first half of the cycle and read in the second half. The branch instruction (denoted with BR) is resolved in the second stage of the pipeline (ID stage) if the operands are ready.

Q1. (8 points) Assume that a program consists of 20% branch instructions, 40% loads, 20% stores, and 20% ALU instructions. Assume that each branch instruction is always dependent on the instruction immediately before it. In this program, the instruction before the branch instruction is always a LD instruction (load instruction). Answer the following:

(a) What is the CPI for this program if 20% of the branches are taken and 80% are not-taken? Assume that there are no other data dependencies among instructions, and we do not have any branch prediction (or branch delay slots).

(b) Assume that there are no other data dependencies among instructions, and our hardware branch predictor makes 90% accurate predictions. What is the CPI for this program if 20% of the branches are taken and 80% are not-taken? Assume that delay slot mechanism or software-based branch prediction are not available.

Q2 (2 points). I have a compiler-based branch predictor that sets the "likely bit" in the instructions after profiling the whole application. It turns out for ALL the applications I care about and run on my processor, the branch prediction accuracy is ALWAYS between 35% (min accuracy, in the worst-case application) and 40% (max accuracy, in the best-case application). That is, the branches are always predicted accurately at least 35% of the time, but never more than 40% of the time. I cannot employ hardware branch prediction techniques or devise any other software-based branch predictor. Do you think it is possible to achieve higher accuracy using the current compiler-based branch predictor? Please justify your answer.

Q1. (a) CPI = $1 + 0.2 \times 0.2 \times 3$ → 3 cycles wasted

0.2 → % of Br; 0.2 → taken

$+ 0.2 \times 0.8 \times 2$ → 2 cycles wasted

0.2 → % of Branches; 0.8 → not-taken

$= 1 + 0.12 + 0.32 = 1.44$

when there is no Branch prediction you waste 3 cycles for all "taken"

"branch".

Even for "non-taken" Branches, you have 2 cycle penalty due to resource hazard.

CPI = $1 + 0.2 \times 0.1 \times 3$

(0.1 → mis prediction)

$+ 0.2 \times 0.9 \times 2$

(0.9 → correct prediction)

$= 1 + 0.06 + 0.36$

$= 1.42$

3. compiler bugs.

flip the bit :->