CS 422 - Problem Set I
IIT Kanpur

February 5, 2018

Problem 1. You would like to add a branch predictor to your Baseline processor, and you are considering two options: PikaChooser and CharWizard. Evaluate the speedup of each relative to your Baseline if branches are 15% of all instructions. Assume normal CPI is 1, but the branch mispredict penalty is 2 extra stall cycles.

PikaChooser:
10% misprediction rate
Will increase the cycle time by 15%
CharWizard:
12% misprediction rate
Will increase the cycle time by 20%

Problem 2. A pipeline with 18 stages runs a program P having 4,076 instructions. Branches comprise 17 percent of the instructions, and the branch not taken assumption holds for branch prediction. Further assume that 41 percent of the branches are predicted correctly, and there is an average penalty of 1.7 cycles for each mispredicted branch. Additionally, 2 percent of the total instructions incur an average of 1.3 stalls each. Caluclate the CPI of P.

Problem 3. Suppose the branch frequencies (as percentages of all instructions) are as follows:

Conditional branches 15%
Jumps and calls 1%
Taken conditional branches 60% are taken

a) We are examining a four-deep pipeline where the branch is resolved at the end of the second cycle for unconditional branches and at the end of the third cycle for conditional branches. Assuming that only the first pipe stage can always be done independent of whether the branch goes and ignoring other pipeline stalls, how much faster would the machine be without any branch hazards?

b) Now assume a high-performance processor in which we have a 15-deep pipeline where the branch is resolved at the end of the fifth cycle for unconditional branches and at the end of the tenth cycle for conditional branches. Assuming that only the first pipe stage can always be done independent of whether the branch goes and ignoring other pipeline stalls, how much faster would the machine be without any branch hazards.

Problem 4. An (m,n) correlating branch predictor uses the behavior of the mos recent 'm' executed branches to choose from 2^m predictors, each of which is an n-bit predictor. A two-level local predictor works in similar fashion , but only keeps track of the past behavior of each individual branch to predict future behavior. There is a design trade-off involved with such predictors: Correlating predictors require little memory for history which allows them to maintain 2-bit predictors for a large number of individual branches (reducing the probability of branch instructions reusing the same predictor), while local predictors require substantially more memory to keep history and are thus limited to tracking a relatively small number of branch instructions. For this exercise, consider (1,2) correlating predictor that can track four branches (requiring 16 bits) versus a (1,2) local predictor that can track two branches using the same amount of memory . For the following branch outcomes, provide each prediction, the table entry used to make the prediction , any updates to the table as a result of the prediction and the final misprediction rate of each predictor. Assume that all branches up to this point have been taken. Initialize each predictor to the following:

Correlating predictor

Entry	Branch	Last outcome	Prediction
0	0	T	T with one misprediction
1	0	NT	NT
2	1	T	NT
3	1	NT	T
4	2	T	T
5	2	NT	T
6	3	T	NT with one misprediction
7	3	NT	NT

Local predictor

Entry	Branch	Last 2 outcomes (right is most recent)	Prediction
0	0	T,T	T with one misprediction
1	0	T,NT	NT
2	1	NT,T	NT
3	1	NT	T
4	2	T,T	T
5	2	T,NT	T
6	3	NT,T	NT with one misprediction
7	3	NT,NT	NT

Branch PC (word address)	Outcome
454	T
543	NT
777	NT
543	NT
777	NT
454	T
777	NT
454	T
543	T

Problem 5. Suppose we have a deeply pipelined processor, for which we implement a branch-target buffer for the conditional branches only. Assume that the misprediction penalty is always four cycles and the buffer miss penalty is always three cycles. Assume a 90% hit rate, 90% accuracy and 15% branch frequency. How much faster is the processor with the branch-target buffer versus a processor that has a fixed two-cycle branch penalty? Assume a base clock cycle per instruction (CPI) without branch stalls of one.

Problem 6. Consider a branch-target buffer that has penalties of zero, two, and two clock cycles for correct conditional branch prediction, incorrect prediction, and a buffer miss, respectively. Consider a branch-target buffer design that distinguishes conditional and unconditional branches, storing the target address for a conditional branch and the target instruction for an unconditional branch.

a) What is the penalty in clock cycles when an unconditional branch is found in the buffer?

b) Determine the improvement from branch folding for unconditional branches. Assume a 90% hit rate, an unconditional branch frequency of 5%, and a two-cycle penalty for a buffer miss. How much improvement is gained by this enhancement? How high must the hit rate be for this enhancement to provide a performance gain?