Homework #3

Cavit Cakir 23657

Assigned: 04/05/2021
Due: 12/05/2021

Note: Only hardcopy submissions are acceptable to the instructor during the lecture hour.

1. Assume that your benchmark has the following instruction frequencies:

	R-type	branch	the rest
Benchmark	40%	20%	40%

Also assume the following branch predictor accuracies:

	Always-taken	Always-not-taken	Dynamic predictor
Benchmark	45%	55%	60%

Assume also that CPI = 1 without branch mispredictions. (**25pts**)

a. Stall cycles due to mispredicted branches increase CPI. Assume that the branch outcomes are determined in **`MEM`** stage, that there are no data hazards, and that no delay slots are used. Calculate the CPI after the stalls due to mispredicted branches with "Always-taken", "Always-not-taken", and "Dynamic" predictors? (Hint: When a branch is mispredicted, the instructions in **`IF, ID,`** and **`EX`** stages must be flushed out of the pipeline) (**5 pts**)

misprediction -> if + id + ex => costs 3

"Always-taken"	CPI -> (**0.8** * (1)) + (**0.2** * (0.45 * 1 + 0.55 * 4)) = 1.33
"Always-not-taken"	CPI -> (**0.8** * (1)) + (**0.2** * (0.55 * 1 + 0.45 * 4)) = 1.27
"Dynamic"	CPI -> (**0.8** * (1)) + (**0.2** * (0.60 * 1 + 0.40 * 4)) = 1.24

b. With the "dynamic" predictor, what speedup would be achieved if we eliminated half of the branches by turning each branch into two R-type instructions? Assume that the performance of the predictor improves to 90% after this modification. (**Hint**: `clock count = IC × CPI`) (**10 pts**)

Lets assume there were 100 instructions before elimination.
So, 40 R-type, 20 branch, and 40 the rest instructions.

After elimination of half of the branch instructions and turning to 2 R-type instructions;
60 R-type, 10 branch, and 40 the rest.

After CPI = (**100/110** * (1)) + (**10/110** * (0.9 * 1 + 0.1 * 4)) = 1.027
After Clock Count = 110 * 1.027 = 113
Before clock Count = 100 * 1.24 = 124

Speedup = 124 / 113 = 1.097

2. Consider a program consisting of five conditional branches. Below are the outcomes of each branch for one execution of the program (T for taken, N for not taken).

   Branch 1: T-T-T
   Branch 2: N-N-N-N
   Branch 3: T-N-T-N-T-N
   Branch 4: T-T-T-N-T
   Branch 5: T-T-N-T-T-N-T

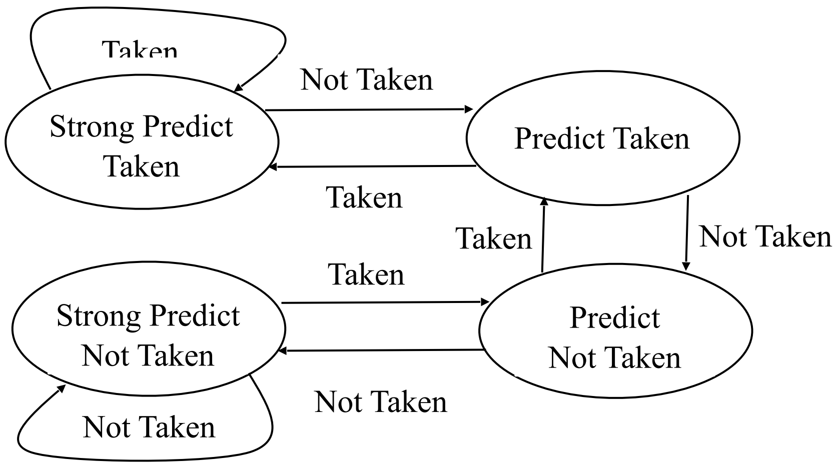
   For dynamic schemes, assume each branch has its own prediction buffer. List the predictions for the following branch prediction schemes for the execution of the code above. What are the total prediction accuracies? (**25 pts**)

   a. There is one 1-bit dynamic predictor for each branch, and each is initialized to "Predict Not Taken". (**5 pts**)

Branch 1: N-T-T	Accuracy: 2/3
Branch 2: N-N-N-N	Accuracy: 4/4
Branch 3: N-T-N-T-N-T	Accuracy: 0/6
Branch 4: N-T-T-T-N	Accuracy: 2/5
Branch 5: N-T-T-N-T-T-N	Accuracy: 2/7

   b. You use 2-bit dynamic predictor and that each branch has its own prediction buffer which is initialized to "Predict Not Taken". Compute the prediction accuracy for each branch and overall branch prediction accuracy. (**10 pts**)

Branch 1: N-T-T	Accuracy: 2/3
Branch 2: N-N-N-N	Accuracy: 4/4
Branch 3: N-T-N-T-N-T	Accuracy: 0/6
Branch 4: N-T-T-T-T	Accuracy: 3/5
Branch 5: N-T-T-T-T-T-T	Accuracy: 4/7



3. Consider a processor whose pipeline depth is 20 (i.e. 20-stage pipeline) and issue width is 4 (i.e., four-issue processor). Consider also the following figures for branch instructions and branch prediction scheme adopted. (**25 pts**)

Percentage of branch instructions of all executed instructions	Branch prediction accuracy	Branch outcome known in stage
10%	90%	15

a. If there are no hazards (no structural hazard, no data dependency, and no branch mispredictions), what is the *ideal* performance improvement over a single-issue processor with the classical five-stage pipeline with the same ISA (i.e. ideal speedup)? Assume that the clock cycle time decreases in proportion to the number of pipeline stages. (**5 pts**)

When the issue number increases 1 to 4, performance multiplies by 4x.
When the stage number increases 5 to 20, performance multiplies by 4x.
So the total is 4*4 -> 16x performance increase

b. How many branch instructions can be "in progress" at most (i.e. already fetched from the memory but not completed yet) at any given time? (**10 pts**)

if all the branch predictions are correct,
The maximum instruction can be 'in progress', in the 20th clock time when the last instruction of the first pipeline, and the first instruction of last pipeline.
So, 20 * 4 = 80.

c. How many instructions are fetched from the wrong path for each branch mispredictions and need to be flushed? Calculate the effect of mispredictions on CPI (clock cycle per second). Assume that there is no other hazard. (**10 pts**)

Misprediction costs 14 cycles.

New CPI = (Not branch instruction * old CPI) + branch instructions * (correct predictions% * old CPI + wrong predictions% * penalty)
= (0.9 * 0.25) + 0.1 * (0.9 * 0.25 + 0.1 * 14.25) = 0.39

4. Consider the following 10-bit floating-point number system in which we use 1 bit for the sign, 4 bits for the exponent, and 5 bits for the significand (mantissa). The exponent bias is equal to 7, and the largest and smallest biased exponents are reserved (similar to IEEE 754 Standard). The significand uses a hidden (implicit) 1. The value of a floating-point number can be calculated using the following expression:

$$\text{value} = (-1)^{\text{sign}} \times (1.0 + \text{significand}) \times 2^{\text{exponent-bias}}$$

Find floating-point representation of the following real numbers: (**10 pts**)

x = 45.0 => 32 * (1 + 0.40625)
x = (-1)^0 x 1.01101 x 2^5
exponent - 7 = 5
exponent = 12
Sign, Exponent, Fraction = 0 1100 01101

y = 0.75 => 1/2 * (1 + 1/2)
y = (-1)^0 x 1.1000 x 2^(-1)
exponent - 7 = -1
exponent = 6
Sign, Exponent, Fraction = 0 0110 10000

z = -44.0 => 32 * (1 + 0.375)
z = (-1)^1 x 1.01100 x 2^5
exponent - 7 = 5
exponent = 12
Sign, Exponent, Fraction = 1 1100 01100

5. Consider a processor with the following parameters:

Base CPI, no stalls due to cache misses	0.5
Processor speed	2.5 GHz
Main memory access time	100 ns
First-level cache miss rate per instruction	3%
1st option for the second-level cache	
Direct-mapped: the latency	20 cycles
Local miss rate of the second level cache, direct mapped	50%
2nd option for the second-level cache	
8-way set associative: the latency	50 cycles
Local miss rate of the second level cache, 8-way set associative	40%

(**15 pts**)

a. We are considering two alternatives for second-level cache memory as shown above. What are the global miss rates if we use second-level direct-mapped cache (1st option) and second-level 8-way set-associative cache (2nd option)? (**5 pts**)

Global miss rate if we use 1st option:
3% * 50% = 1.5%
Global miss rate if we use 2nd option:
3% * 40% = 1.2%

b. Calculate the CPI for the processor in the table using: i) only first-level cache, ii) a second-level direct mapped cache, and iii) a second-level 8-way set associative cache. (**10 pts**)

i) effective cpi = base cpi + first-level miss rate * (main memory access time / (1/processor speed))
= 0.5 + 0.03 * (100 / 0.4)
= 8

ii) effective cpi = base cpi + (first-level miss rate * Direct-mapped: the latency) + (global miss * (main memory access time / (1/processor speed)))
= 0.5 + (0.03 * 20) + (0.015 * (100 / 0.4))
= 4.85

ii) effective cpi = base cpi + (first-level miss rate * 8-way set associative: the latency) + (global miss * (main memory access time / (1/processor speed)))
= 0.5 + (0.03 * 50) + (0.012 * (100 / 0.4))
= 5