Cody Martin

18-447 Homework 4

1. a)

Branch 1: N | TTTT...TTN Branch is predicted 1001 times (0...1000), taken 1000 times, not taken 1 time on the 1001st prediction, incorrect predictions on the first taken and the not taken on the 1001st prediction so accuracy is 999/1001 = 99%

Branch 2: N | TNNNTNNNTNNN...TNNN Branch is predicted 1000 times, branch is taken 250 times, and for each taken, the predictor mispredicts twice. So, there are 250\*2 mispredictions = 500. So the accuracy is (1000 - 500)/1000 = 50%.

Branch 3: N | TNTN...TN Branch is predicted 1000 times, branch is taken 500 times, and mispredicts twice per taken branch. So there are 500\*2 = 1000 mispredictions, so accuracy is (1000 - 1000)/1000 = 0%.

b)

Branch 1: SNT | WNT, WT, ST...ST, WT Branch predictor is wrong 3 times out of 1001 times, so accuracy is (1001 - 3)/1001 = 99%

Branch 2: SNT | WNT, SNT, SNT, SNT...WNT, SNT, SNT, SNT Branch predictor is incorrect on every time the branch is taken, which is 250 times. so (1000-250)/1000 = 75%.

Branch 3: SNT | WNT, SNT...WNT, SNT Branch Predictor is wrong on every taken, which is 500 times. So accuracy is (1000-500)/1000 = 50%.

c)

Branch 2: WNT | WT, WNT, SNT, SNT, WNT, SNT, SNT, SNT...WNT, SNT, SNT, SNT Branch is wrong for all takens AND or the first not taken. so it is wrong for 251 times, so accuracy is (1000 - 251)/1000 = 74.9%.

Branch 3: WNT | WT, WNT, WT, WNT...WT, WNT Branch predictor is wrong every time. So its accuracy is 0%.

d) If we look at the tables below, eventually the branch predictor becomes perfect, predicting each branch correct every time. So if we arent considering the first iterations of the loop, we can see that the accuracy of the branch predictor now is 100%.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch 1: |  |  |  |  |  |
| Loop i: | GHR | Table: |  |  |  |
| init | NT/NT | SNT | SNT | SNT | SNT |
| 0 | NT/T | SNT | SNT | WNT | SNT |
| 1 | T/T | WNT | SNT | WNT | SNT |
| 2 | T/T | WT | SNT | WNT | SNT |
| 3 | T/T | ST | SNT | WNT | SNT |
| 4 | T/T | ST | SNT | WNT | SNT |
| 5 | T/T | ST | SNT | WNT | SNT |
| 6 | T/T | ST | SNT | WNT | SNT |
| 7 | T/T | ST | SNT | WNT | SNT |
| 8 | T/T | ST | SNT | WNT | SNT |
| 9 | T/T | ST | SNT | WNT | SNT |
| 10 | T/T | ST | SNT | WNT | SNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch 2: |  |  |  |  |  |
| Loop i: | GHR | Table: |  |  |  |
| init | NT/NT | SNT | SNT | SNT | SNT |
| 0 | NT/T | SNT | SNT | WNT | SNT |
| 1 | T/NT | SNT | SNT | WNT | SNT |
| 2 | NT/NT | SNT | SNT | WNT | SNT |
| 3 | NT/NT | SNT | SNT | WNT | SNT |
| 4 | NT/T | SNT | SNT | WT | SNT |
| 5 | T/NT | SNT | SNT | WT | SNT |
| 6 | NT/NT | SNT | SNT | WT | SNT |
| 7 | NT/NT | SNT | SNT | WT | SNT |
| 8 | NT/T | SNT | SNT | ST | SNT |
| 9 | T/NT | SNT | SNT | ST | SNT |
| 10 | NT/NT | SNT | SNT | ST | SNT |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch 3: |  |  |  |  |  |
| Loop i: | GHR | Table: |  |  |  |
| init | NT/NT | SNT | SNT | SNT | SNT |
| 0 | NT/T | SNT | SNT | WNT | SNT |
| 1 | T/NT | SNT | SNT | WNT | SNT |
| 2 | NT/T | SNT | SNT | WT | SNT |
| 3 | T/NT | SNT | SNT | WT | SNT |
| 4 | NT/T | SNT | SNT | ST | SNT |
| 5 | T/NT | SNT | SNT | ST | SNT |
| 6 | NT/T | SNT | SNT | ST | SNT |
| 7 | T/NT | SNT | SNT | ST | SNT |
| 8 | NT/T | SNT | SNT | ST | SNT |
| 9 | T/NT | SNT | SNT | ST | SNT |
| 10 | NT/T | SNT | SNT | ST | SNT |

2. (1) This takes 17\*5 + 1 cycles to execute = 86 cycles

(2) Now this takes 60 cycles to execute.

(3) Now this takes 74 cycles to execute.

3. (1) The scoreboard will show that R1, R2, and R3 and not valid while the pipeline is executing the first instruction. Therefore, when the second instruction goes to execute, the pipeline will look to the scoreboard to see if all of its registers (R2, R4, R5) are valid. It will see that R2 is not valid because instruction one is using it. The scoreboard does not keep track of what instructions are reading registers and what instructions are write to registers.

(2) The only data dependency this would stall on would be a read after write dependency. This new architecture would avoid stalling for the write after read dependency and the read after read dependency. The table below illustrates the scoreboard for this new implementation:

|  |  |  |
| --- | --- | --- |
| Register Number | Valid | R/W |
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 1 |
| 4 | 1 | 1 |
| 5 | 0 | 1 |
| 6 | 1 | 1 |
| 7 | 0 | 0 |

The valid bit will signify whether or not the register is being used, and the R/W bit will signify whether the register is being read or written. The pipeline will only stall if the former instruction’s R/W bit is WRITE and the the valid bit is 0 on an instruction using that given register that the current instruction to which the current instruction is asking access.

4. (1) This will happen when the single pattern history table has less capacity than the number of branches in the program. If you have too many branches for the number of slots in the table, then two branches will identify with the same place in the table. This can also happen if the sequence of taken and not taken branches before 2 different branches is very similar. If this happens, then that sequence given by the GHR can have multiple branches pertain to it.

(2) a. Predictor A will have a lower prediction accuracy than predictor B when you have a lot of branches. This is because The branches can interfere with other. If interference happens, then the table will take longer reach its equilibrium, or not reach it at all. This will cause less prediction accuracy, especially in shorter loops.

b.

c. Yes there is. If two branches that are interfering do not change the table slot that they correspond to, then they can work together and not impact the branch prediction acuracy.

5.

(1) For Machine A, we have a P% chance of guessing it right. That means we have a (1-P)% chance of guessing wrong, and wasting 7 cycles doing not needed instructions if it should be taken but we dont take it, and wasting 12 cycles doing not needed instructions if it should not be taken and we take it. As a weighted average, our expected number of cycles missed is (1-P)\*(0.4)\*7 + P\*(0.6)\*12 = 2.8 - 2.8P +7.2P = 2.8 - 4.4P. expected number of cycles Machine be misses is always 3. So 2.8 + 4.4P > 3, so P > 4.5%.

(2) I quit.

6.

a)

ADD R7 <- R6, R7

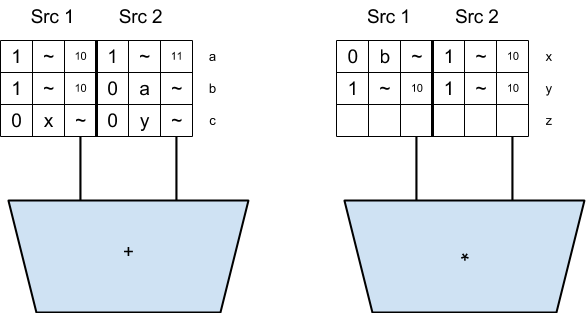
ADD R3 <- R6, R7

MUL R0 <- R3, R6

MUL R2 <- R6, R6

ADD R2 <- R0, R2

b)



c)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Valid | Tag | Value |
| R0 | 0 | x | ~ |
| R1 | 1 | ~ | 5 |
| R2 | 0 | y | ~ |
| R3 | 0 | b | ~ |
| R4 | 1 | ~ | 8 |
| R5 | 1 | ~ | 9 |
| R6 | 1 | ~ | 10 |
| R7 | 0 | a | ~ |

7. This Data Flow Diagram executes k^N. N is a backwards counter that counts how many times k multiplies by itself.