LSU EE 4720 **Homework 7** **Due: 28 April 2025 7:30**

Solution *Formatted 10:29, 6 May 2025*

Collaboration Rules

Each student is expected to complete his or her own assignment. It is okay to work with other students and to ask questions in order to get ideas on how to solve the problems or how to overcome some obstacle (be it a question of MIPS or assembler syntax, interpreting error messages, how a part of the problem might be solved, etc.) It is also acceptable to seek out assembly language resources for help on MIPS, etc. It is okay to make use of AI LLM tools such as ChatGPT and Copilot to generate sample code. (Do not assume LLM output is correct. Treat LLM output the same way one might treat legal advice given by a lawyer character in a movie: it may sound impressive, but it can range from sage advice to utter nonsense.)

After availing oneself to these resources **each student is expected to be able to complete the assignment alone**. Test questions will be based on homework questions and **the assumed time needed to complete the question will be for a student who had solved the homework assignment on which it was based**.

Student Expectations

Some of the problems require thought, and students are expected to persevere until they find a solution. It is each student's duty to him or herself to resolve frustrations and roadblocks quickly, hopefully helped along by the satisfaction of making progress. There are plenty of old problems and solutions to look at. One way to resolve issues is to ask Dr. Koppelman or others for help.

Resources

Questions about superscalar MIPS implementations can be found in most final exams.

Problem 1: Locate 2024 Final Exam Problem 4, which asks for analysis of two patterns on bimodal and local predictors.

(*a*) Solve 2024 Final Exam Problem 4(a).

See the posted solution.

(*b*) What is the smallest local history size for which branches B1 and B2 (from 2024 Final Exam Problem 4) are each predicted at 100% accuracy? *This could have been part b on the final exam question.* The branch outcomes are shown below for convenience, they are the same as those in the final exam.

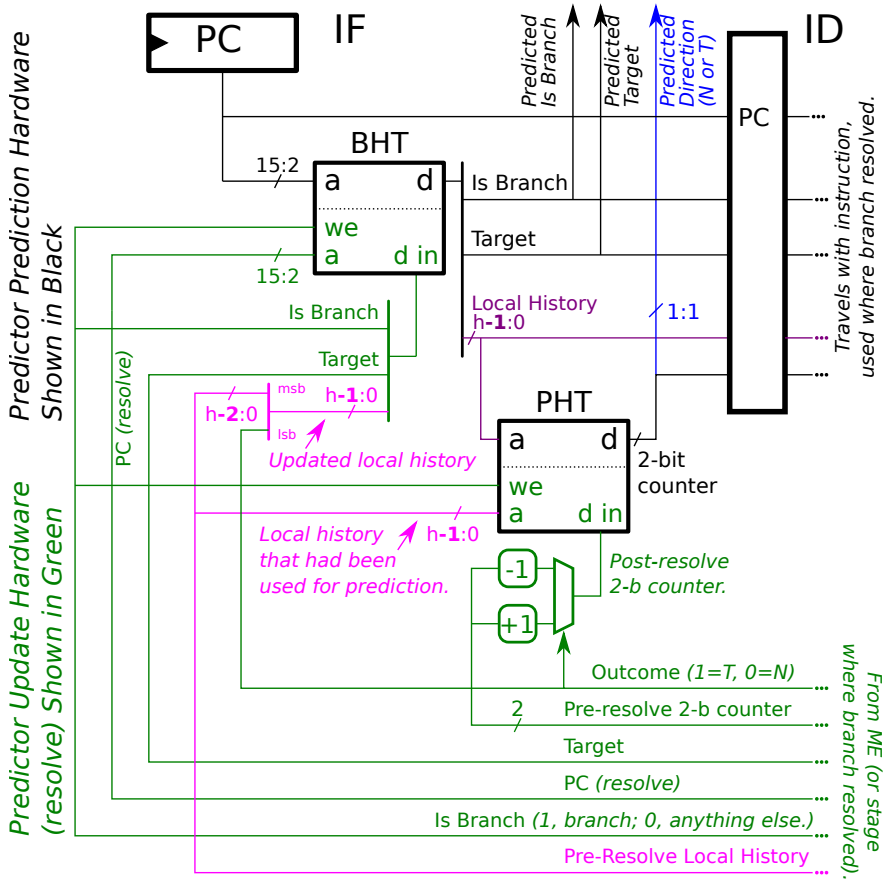
```
B1:  N  N  T  T  T  N  T    N  N  T  T  T  N  T     N  N  T  T  T  N  T   <- Outcome

B2:   N  T  N  T  N  T  N    T  N  T  N  T  N  T     N  T  N  T  N  T  N  <- Outcome
```

The first table below shows that on a 3-outcome local history predictor pattern NTN occurs in both branches and that they *harmfully collide*, meaning they share a PHT entry and their next outcomes are different. After that is a table showing the 7 rotations of the patterns of B1 and the 2 rotations of the patterns of B2. The patterns are sorted to make it easy to find PHT collisions. For the pattern starting NTN four outcomes are needed to distinguish B1 from B2, and for the pattern starting TNT five outcomes are needed to distinguish them. So the minimum local history size is 5.

```
Three-Outcome B1 Local Histories Followed By Next (4th) B1 and B2 Outcomes

Patrn   B1   B2
N N T   T
N T T   T
T T T   N
T T N   T
T N T   N    N
N T N   N    T
T N N   T
```

```
Twelve-Outcome Local History Patterns

N N T T T N T N N T T T   B1

-------
N T N N T T T N T N N T   B1
N T N T N T N T N T N T   B2
-------                  <- Need four outcomes to use separate PHT entries.

N T T T N T N N T T T N   B1
T N N T T T N T N N T T   B1

---------
T N T N N T T T N T N N   B1
T N T N T N T N T N T N   B2
---------                <- Need five outcomes to use separate PHT entries.

T T N T N N T T T N T N   B1
T T T N T N N T T T N T   B1
```

(*c*) Someone foolishly argues that limiting the local history to only three outcomes keeps the cost of the branch prediction hardware a tiny fraction of the cost of one with a 16-outcome local history. Explain why that argument is foolish based on the diagram below. Use the sizes, in bits, of the BHT and PHT in your explanation. *Note: the "tiny fraction" phrase was not in the original problem.*



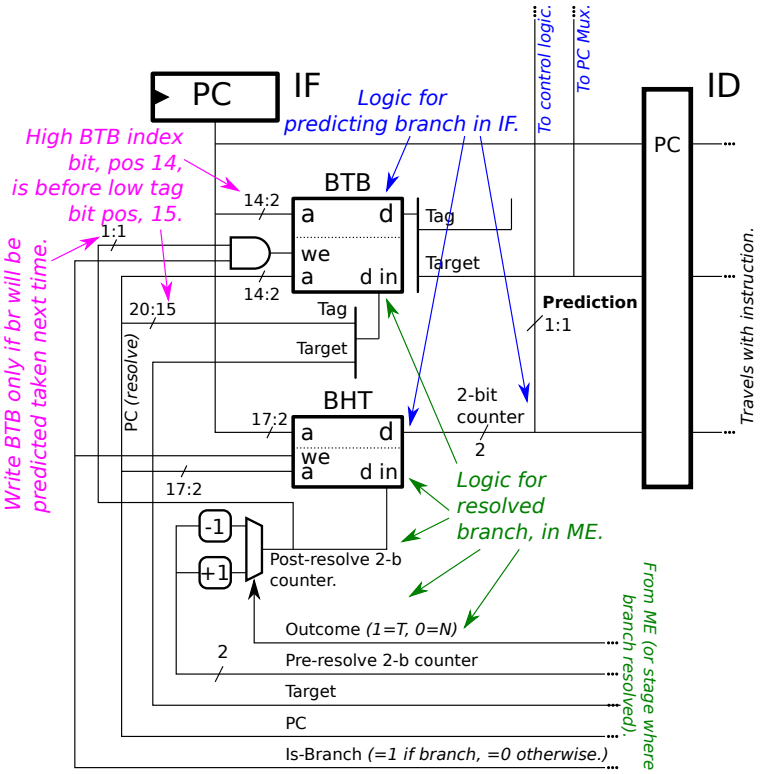
The point of the problem was to notice that the BHT size includes the target and that doesn't change with local history size.

For an h-outcome local history the PHT size is 2×2^h. Consider a change from 16 outcomes to 3. That would reduce the cost of the PHT from $2^{17} = 131072\,\text{b}$ to $2^4 = 16\,\text{b}$. That's a big improvement. But one must also consider the BHT. Assuming the Target field is 16 bits and 1 for the Is Branch field, the size of an entry is $1+16+h$. The number of entries in the illustrated BHT is $2^{16-2} = 2^{14} = 16834$. For $h = 16$ the BHT size is $2^{14}(1+16+16) = 540672\,\text{b}$ and the total predictor size is $671744\,\text{b}$. Dropping to $h = 3$ the BHT cost is about half, $2^{14}(1+16+3) = 327680\,\text{b}$ and the total cost is $327696\,\text{b}$, about half. However the prediction accuracy would be much lower.

To better see the comparison look at $h = 8$. The size of the PHT and BHT would be $410112\,\text{b}$, only 25% higher than the 3-outcome predictor, but the accuracy would be much higher.

There is another problem on the next page.

Problem 2: The diagram below is of a bimodal predictor in which the BHT (branch history table) can keep track of eight times as many branches as the BTB (branch target buffer). Some purple text on the left explains that the BTB should only be updated for a branch that will be predicted taken next time.



Show two sets of branches. In the first set heeding the advice of that text improves performance. In the other set a branch predictor that updates every time would do as well as one that updates only for a branch that will be predicted taken the next time it is encountered. The solution should look something like the sample below, but with the branch addresses, such as `0x12340`, and outcome patterns changed. Additional branches can be added to each set.

Solution on next page.

First, the two branches need to collide, so change their addresses so they use the same BTB entry. That's the case for both sets below. (They use the same BTB entriy if bits 14:2 in the two are the same.) For that purple text to be effective the predictor needs to know whether a colliding branch will be predicted taken and so the two branches need to use a different BHT entry. So in Set 1 make sure bits 17:15 of the two branches are different (while keeping bits 14:2 the same). Bias the first branch, B11, not-taken and the second branch, B12, taken. If the purple text is heeded then the predictor won't waste a BTB entry on B11 and so B12 can use the entry, which it needs for its target. If the purple text is ignored on the Set 1 branches the predictor will correctly predict B12 taken, but the BTB entry will be for B11 and so there will be no target (at least not a correct one).

Next consider Set 2. Because the two branches use the same BHT entry there is no way to reliably correctly predict that B21 will be not-taken next time and so no way to prevent the B21 target from overwriting the B22 target in the BTB.

```
# Set 1: Branches have same BTB entry but different BHT entries.
0x10000 B11:  N  N  N  N ...
0x18000 B12:   T  T  T  T ...

# Set 2: Branches have same BTB entry and same BHT entries.
0x100000 B21:  N  N  N  N ...
0x200000 B22:   T  T  T  T ...
```