1. (20 pts) What is branch prediction, and what are the reasons why a superscalar CPU should use branch prediction?

Branch prediction is the process done by a piece of hardware in the superscalar CPU called the branch predictor. This process involves predicting which branch an operation will take if it can choose between two different branches in an instruction. The predictions occur when instructions from one block of code try to jump to a different block of code or keep going straight to the next block after the original block. This can usually occur in a loop which is where this prediction tends to be used since the same block that is given the option to jump will be run multiple times inside of a loop causing the decision on whether to jump or not to come up multiple times during the run time of the code. Because this block and its decision to jump or not jump are run so many times the predictor can use past decisions made by the block to guess whether or not it will choose to jump or not jump. This early guessing before the block can get to the point where it can make its decision speeds up the run time if the predictor guesses correctly since it will be able to anticipate the resources that it will need to get from the next block the code is jumping to. These predictions are done when the CPU would be idling and waiting for a retrieval of an uncached value located in external physical memory. This means that even if the prediction made by predictor is wrong then there is no slow down to the overall run time since it would just discard its prediction and would have been as if it was idling like it was originally.

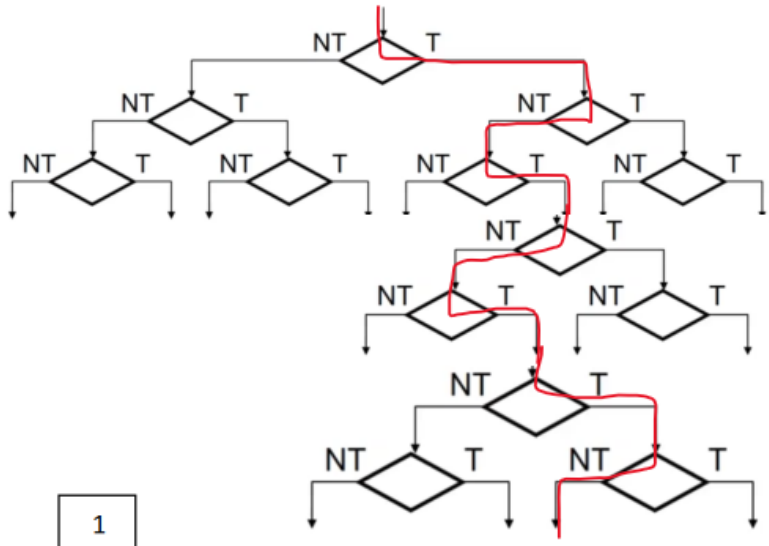
2. (40 pts) The control flow chart (CFG) of a program is given in Figure 1, and the sequence of basic blocks executed is also given below. Please record the execution pattern for the branch in the loop of this CFG (N for branch not taken, and T for branch taken) by assuming no branch prediction. Then, do the branch prediction for the branch in the loop (i.e. the branch jumping from BB3 to BB4 or BB5) based on the algorithm discussed in the class (shown in Figure 2), and calculate the mis-prediction rate (Assume the branch prediction for the first two dynamic execution is always taken).

Execution Trace: Sequence of Basic Blocks Executed:

1 2 3 5 6 2 3 4 5 6 2 3 5 6 2 3 4 5 6 2 3 5 6 2 3 5 6 2 3 4 5 6

Start of loop 1

2 3 5 6 T, 2 3 4 5 6 N, 2 3 5 6 T, 2 3 4 5 6 N, 2 3 5 6 T, 2 3 5 6 T, 2 3 4 5 6 N



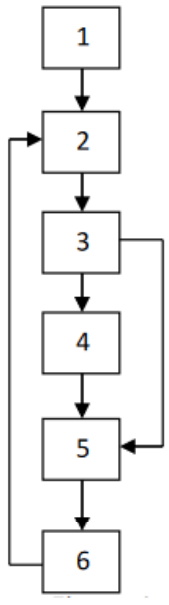


Figure 1. CFG

Second old branch	First old branch	Branch prediction for next branch	Is prediction correct
?	T	T	No
T	N	T	Yes
N	T	T	No
T	N	T	Yes
N	T	T	Yes
T	T	T	No
T	N	T	?

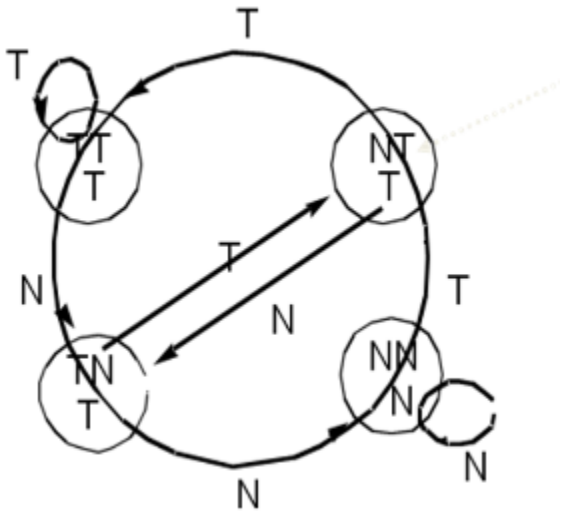


Figure 2. Branch Prediction Algorithm

We technically can ignore the first two branch predictions since the question asked us to do this. So, we can ignore whether they were right or wrong. This means that there was a total of 4 predictions made by the branch predictor and 2 of these predictions were correct. This means the mis-prediction rate would for these instructions using the predictor would be about 50%.

3. (40 pts) Read the paper entitled "Spectre Attacks: Exploiting Speculative Execution" by Paul Kocher et al. (https://spectreattack.com/spectre.pdf), and summarize in your own words what is a spectre attack, and what microarchitectural features make spectre attacks possible.

A spectre attack is microarchitectural attack that exploits the CPU's ability to run speculative executions in an attempt to speed up performance by not wasting time idling. The way this

attack exploits this feature is by relying on the CPU predictor's inability to check if instructions being run are correct. The attack looks through a systems memory somehow to find instruction sequences that could be used to leak private information and then has the CPU use these instructions during its speculative execution process for process like branch prediction. Because the CPU never checks if the speculative execution it is running is the actual execution it should be speculating, the instruction can be run leaking information to the attacker without the CPU knowing that it leaked any information. This is all possible because of the microarchitectural feature that allows a CPU to perform this speculative execution without checking if the instruction actually being run is correct and the only hypothetical way to eliminate this attack as stated in the paper by Paul Kocher is to "change processor designs and update instruction set architectures to give hardware architects and software developers a common understanding as to what computation state CPU implementations are (and are not) permitted to leak".