Lab 2: Branch Prediction

In this lab, you will simulate a **correlating branch predictor** that makes use of 2-bit saturating counters. You are provided with a text file containing a trace of branch instructions consisting of the PC at which each branch occurs, and whether the branch is Taken or Not Taken.

Your goal is write code to evaluate the performance of a correlating branch predictor on this trace. Your output file indicates, for each branch in the trace, whether it was predicted as Taken or Not Taken.

Branch Predictor Architecture
Your design must consist of 2^{m+k} 2-bit saturating counters indexed using *m* LSBs of each branch instruction and a *k* bit Branch History Register (BHR) that records the outcomes of the previous k branches. Each 2-bit saturating predictor starts in the 11 (Predict Taken with High Confidence) state and is updated as per the finite state machine discussed in Lecture 11. Further, you can assume that the BHR is initialized assuming the previous *k* branches were Taken.

The values of m and k are specified in a config file config.txt.

Config File
The config file config.txt contains a two lines, the first with the value of m and the second with the value of k. A sample file for m=12 and k=2 is provided. The largest value of m is 32 (for 32 bit PCs), but we will not input an m larger than 20.

Trace File
The trace file, trace.txt, contains one branch per line. Each line has the PC for the corresponding branch (in hex format) followed by a single bit indicating Taken (1) or Not Taken (0). A sample trace file is provided.

Output Format
The output from your simulator should be a file trace.txt.out that has one line per branch. Each line has a single bit which indicates whether that branch was predicted as Taken (1) or Not Taken (0).

What to Submit

a) Code

1. Your source code. On compilation, we should be able to execute your simulator with the following command
   ./branchsimulator.out config.txt trace.txt
   Your simulator should output a file trace.txt.out in the same directory.
2. A readme.txt with instructions on how to compile your code
3. We will compile and run your source code on Gauss

b) Analysis

For the provided trace.txt, plot a graph of the branch misprediction rate as a function of m with m varying from 10 to 16 for each value of k from 0 to 4 (5 plots in all, one for each value of k). Submit your plot as a PDF file in the same folder as your source code.