University of Southern California
Department of Electrical and Computer Engineering
EE557 Spring 2026
Project 1 – Implement Branch Predictors with Pin Tool
Due: 11:59PM., Friday, February 13th
TOTAL SCORE: /6

Branch Predictors

A branch predictor tries to guess the next instruction for a branch before the branch is resolved. The purpose of the branch predictor is to improve the flow in the pipeline. In this project, you are going to implement various branch predictors: one static branch predictor and three dynamic branch predictors.

a) Always Taken
b) 2-bit Global (00, 01 = untaken, 10,11 = taken; one SPB for all, **Figure 1**)
c) 2-bit Bimodal
d) 2-bit Correlated

In EE457, you have learned about these predictors. Static branch predictors always guess the same direction, either always taken or always not taken. On the other hand, dynamic branch predictors guess directions based on previous observations. **Figure 1** shows a 2-bit Saturating Branch Predictor (SBP).

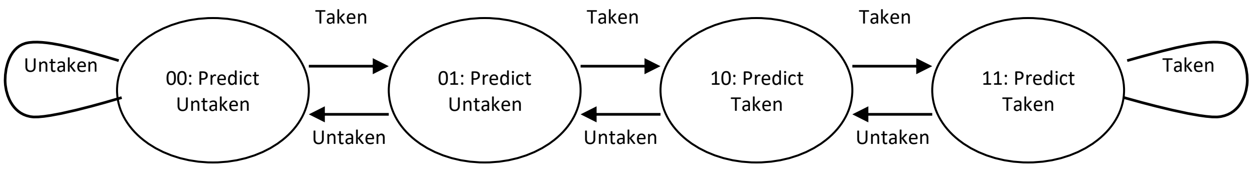

Figure 1. 2-bit SBP

(a) Always taken predictor: all branches are always predicted taken statically.

(b) 2-bit global predictor: one 2-bit SBP is used to predict all branch instructions without separate branch history buffers as shown in **Figure 1**.

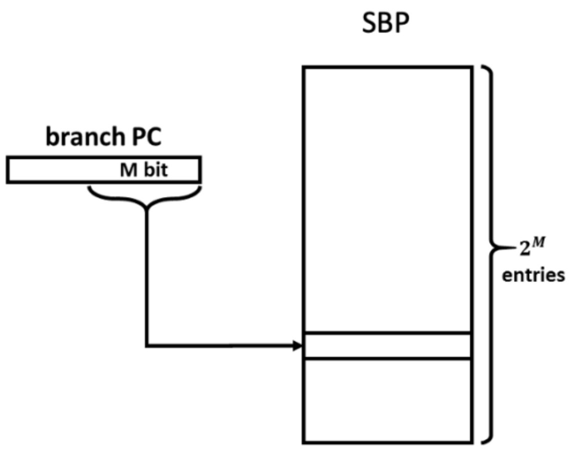

Figure 2. Bimodal Branch predictor

(c) Bimodal predictor: 2-bit SBP is indexed by the branch PC as shown in **Figure 2**. We use 2-bit SBP in each entry. The behavior of the two bits is shown in Figure 1. Initially, two bits are set to 00. If the two bits are 00 or 01, a branch is predicted untaken. If the two bits are 10 or 11, the branch is predicted taken. The two bits are updated once the branch is resolved. If the branch is taken, the bits are increment and decrement otherwise as shown.

(d) **Correlated branch predictor**: it uses both branch PC and recent branches pattern to index the branch prediction buffer as shown in **Figure 3**. The global branch history register is a left-shift register which shifts in the most recently branch result (taken:1 and untaken: 0) and shifts out the oldest branch result. M bit history register and last significant P bit in branch PC are concatenated to index the branch prediction buffer.

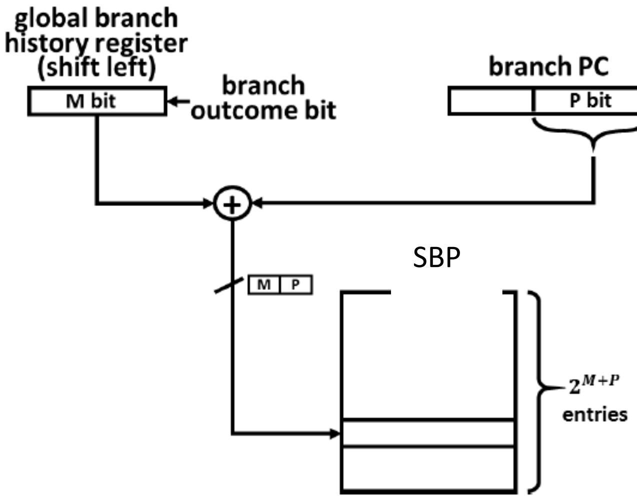

Figure 3. Correlated branch predictor.

Implement with Pin Tool

In project 0, you have tried to run ***inscount0*** to count number of executed instructions. In this project, you are asked to implement the four branch predictors using Pin tool. All your codes should be saved inside the **/home/ee557/Desktop/p1** directory. Currently, there are four files in the **p1** directory.

```
ee557@ ee557:~/Desktop/p1$ ls
bimodal.H  bpred.cpp  makefile  makefile.rules
```

A bimodal predictor example code is provided (/home/ee557/Desktop/p1/bimodal.H). It can help you to understand how to catch the branch instructions and their results under PIN tool. The example implements the bimodal predictor as a library and can be called by the main program (/home/ee557/Desktop/p1/bpred.cpp). Use the following command inside the ***p1*** directory to compile the program.

```
make bpred.test
```

Go to the ***obj-intel64*** directory, you should see the compiled ***bpred.so***.

```
cd obj-intel64
```

We are benchmarking the branch predictors using the "tar" program. To run the program, use the following command that creates a compressed tar archive file of the Pin Tool files in Project 0:

```
pin -t bpred.so -- tar -czPf /home/ee557/Desktop/p0/pin.tar.gz /home/ee557/Desktop/p0/pin-3.17-98314-g0c048d619-gcc-linux
```

Note that the above is one single command, no line breaks in it; there is a double-hyphen with spaces before and after "tar". Please type the command into your terminal instead of using copy-paste to avoid format problems.

For those of you who are using USC CARC or UTM, use the following command to download Pin tool for Linux X86_64. You must also change the PIN_ROOT designated in the makefile to compile.

```
wget https://software.intel.com/sites/landingpage/pintool/downloads/pin-external-3.31-98869-gfa6f126a8-gcc-linux.tar.gz
```

Pin tool will instrument and analyze the branch prediction results. The results are written to ***obj-***

intel64/bpred.out as shown below.

```
===== USC EE557 Fall 2025 - Project 1 =====
Name: [Put your name]
Email: [Put yuor USC email]

Always Taken
Total Prediction: [Number of prediction]
Prediction Rate: [Prediction rate]

2-bit Global
Total Prediction: [Number of prediction]
Prediction Rate: [Prediction rate]

Bimodal Predictor
Total Prediction: ~5e7
Prediction Rate: ~0.8

Correlated Predictor
Total Prediction: [Number of prediction]
Prediction Rate: [Prediction rate]
===========================================
```

Because the program only implements the bimodal predictor, you can only see the prediction result for bimodal. You are asked to implement the four branch predictors. You may modify ***bpred.cpp*** and ***bimodal.H*** to serve your needs.

Though we provide you an example bimodal predictor code, you may need to change the parameters to meet the requirements. For bimodal predictor, we use the ***last 5 bits*** of branch PC to index the branch prediction buffer (BPB). Each entry in BPB is a 2-bit predictor. For correlated predictor, the least significant ***4 bits of branch PC*** **<u>is concatenated to the right side of the</u>** ***global branch history register (4 bits)*** to form ***total 8 bits*** and then index the BPB as shown in Figure 3. The global branch history register is a left-shift register. Right most bit is the newest branch outcome and left most bit is the oldest branch outcome (1: Taken and 0: Not-Taken). Initially, all bits in BPB and global branch history register are set to zeros.

<u>Format of your submission</u>

You must submit **one file named LastnameFirstnameProj1.tar** which includes all your code in a *p1* directory and a report named **LastnameFirstnameProj1.pdf** under the same *p1* directory. In your submission, there should be the ***bpred.cpp*** which contains the main(). The report should contain four CondBranch functions for the four branch predictors with **proper comments**. You also need to show a screenshot of execution of the Pin tool and the result (**cat bpred.out**) similar to P0 in the report.

You need submit a report (LastnameFirstnameProj1.pdf) and the following codes

- bimodal.H
- bpred.cpp
- makefile
- makefile.rules
- .H files that has your implementation of other branch predictors

Submit the tar file (and only the tar file) to the Dropbox on the DEN site.

<u>This project can be done individually or in teams of two</u>