CS320 Fall 2024: Project 1

Submission Due on BrightSpace: Monday, October 21st by 11:59pm. Project demos: during lab session on Tuesday, October 22nd and during TAs office hours that week. Demo schedule will be posted, you will need to sign up for slots.

The goal of this project is to measure the effectiveness of several branch direction predictors on a number of traces of conditional branch instructions.

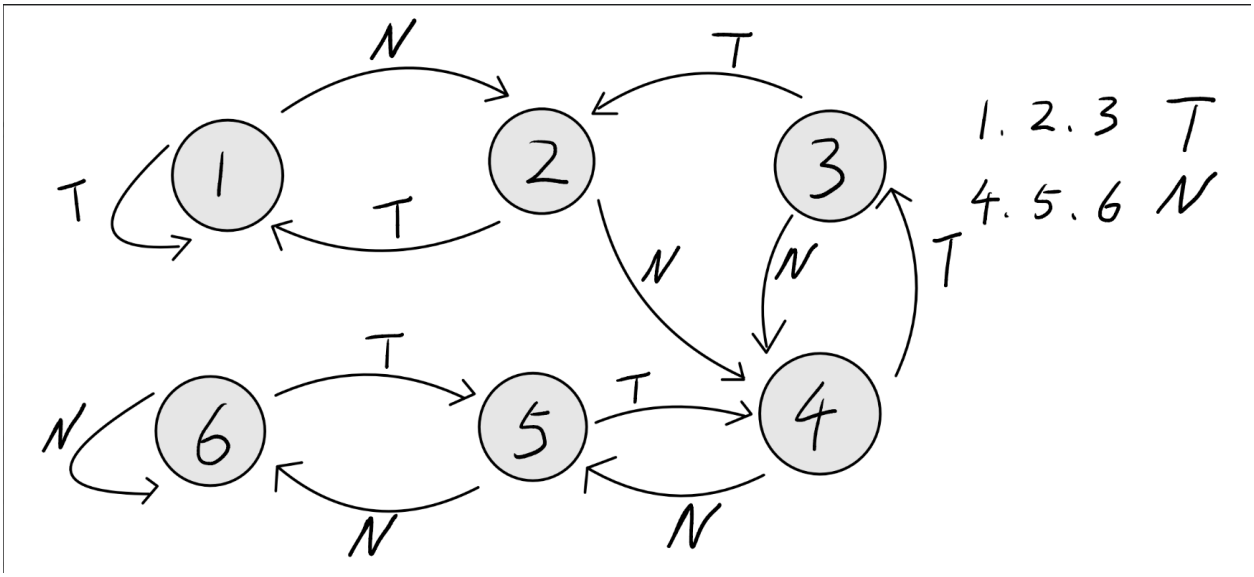
Each trace contains a large number of branch instructions. Each line in the trace contains the following information for each branch: the program counter (expressed as a word address), the actual outcome of the branch. Several trace files are provided for evaluating your predictor designs.

Your goal is to write a program in C or C++ that would use these traces to measure the accuracy of various dynamic branch predictors that we studied in class. The branch outcomes from the trace file should be used to train your predictors. You will need to present results for each trace separately, details of what exactly needs to be submitted are provided below. Note that we also have a secret trace that will be used to assess the correctness of your code.

The following predictors have to be implemented:

1) **[5%] Always Taken.**
2) **[5%] Always Non-Taken**.
3) **[10%] Bimodal Predictor with a single bit of history** stored in each predictor entry. Determine the prediction accuracy of this predictor for the table size of 4, 8, 32, 64, 256, 1024 and 4096 entries. Assume that the initial state of all prediction counters is “Non-Taken” (N)
4) **[20%] Bimodal Predictor with 2-bit saturating counters** stored in each predictor entry. Repeat the same experiments as in part (3) above. Assume that the initial state of all prediction counters is “Strongly Non-Taken” (NN)
5) **[20%] Bimodal Predictor with 3-bit counters** stored in each predictor entry. This predictor has a state machine with six states, as depicted in the figure below. Prediction in states 1, 2 and 3 is “Taken”, and prediction in state 4, 5 and 6 is “Not-taken”. Assume that state machine starts in state 1 (the top left-most state). Repeat

the same experiments as in questions (2) and (3) above for the same table size.



6) **[20%] Gshare predictor**, where the PC is XOR-ed with the global history bits to generate the index into the predictor table. Fix the table size at 4096 entries and determine the prediction accuracy as a function of the number of bits in the global history register. Vary the history length from 2 bits to 12 bits in 1-bit increments. Assume that the initial state of all prediction counters is “Strongly Not Taken” (NN). The global history register should be initialized to one (where 0=NT and 1=T). For example, for 2bits history register, it’s 01, for 5bits, it’s 00001, for 10bits it’s 0000000001. The global history register should be maintained such that the least significant bit of the register represents the result of the most recent branch, and the most significant bit of the register represents the result of the least recent branch in the history. You should use modulo operator before XOR when selecting an index from the table

7) **[20%] Tournament Predictor.** The tournament predictor selects between Gshare and bimodal predictor from question (5) for every branch. Configure Gshare with 4096-entry table and 12 bits of global history, and configure bimodal predictor with a 4096-entry table. Initialize the global history register will all zeroes. Furthermore, configure the selector table with 4096 entries and use the same index as you use for bimodal predictor to index into the selector table (that is, the PC). For each entry in the selector, the two-bit counter encodes the following states: 00 – strongly prefer Gshare, 01 – weakly prefer Gshare, 10 – weakly prefer Bimodal, 11 – strongly prefer bimodal. If the two predictors provide the same prediction, then the corresponding selector counter remains the same. If one of the predictors is correct and the other one is wrong, then the selector’s counter is decremented or

incremented to move towards the predictor that was correct. Initialize all the component predictors to “Strongly Non-Taken” and initialize the selector’s counters to “Strongly Prefer Bimodal”.

Materials on BrightSpace

There is a tar/gzipped archive of materials called `project1.tar.gz` on BrightSpace that contains the following:

1.) A sample output file called `sample_output.txt`, complete with comments

2.) A directory called `examples`, containing code that shows how to read the input

3.) A directory called `traces`, containing the following 6 trace files:

`long_trace1.txt` (15 million branch instructions)

`long_trace2.txt` (20 million branch instructions)

`long_trace3.txt` (25 million branch instructions)

`short_trace1.txt` (2 million branch instructions)

`short_trace2.txt` (3 million branch instructions)

`short_trace3.txt` (4 million branch instructions)

4.) A directory called `correct_outputs`, containing correct outputs for the given traces. These can and should be used to check that your program works correctly and outputs the results in the required format.

To access these materials, download a copy of `project1.tar.gz` from BrightSpace, `cd` into the directory where you placed the tar/gzipped archive and issue the command:

```
tar -xzvf project1.tar.gz
```

This will create a new directory (named `project1`) containing the files mentioned above.

Submission Requirements

NOTE: Please carefully read and follow all directions while preparing your submissions. Submissions will be graded using a script. Failure to follow these directions will likely crash the script, causing the TAs to have to manually grade your

submission, resulting in **deducting points from your grade.** Examples of things to watch out for: the directory inside your tar archive isn't named with your BU-ID, incorrectly named executable after `make`, using standard input and output for I/O, and not using command-line arguments for the names of the input and output files.

You need to submit your source code, so that we can compile it and test for correctness. For checking your code, we will be using the same traces that you used for generating your results, plus some more traces that you will not have access to.

The code that you submit should compile into a single executable called **`predictors`** with a simple `make` command. **Projects that fail to compile will result in a zero grade.** This executable should run all of the predictors on the given trace, which will be specified via command line options as follows:

```
./predictors input_trace.txt output.txt
```

Where `input_trace.txt` is one of the provided branch trace files and `output.txt` stores the branch statistics of your simulation.

The output file should have the following format: (an example text file is on BrightSpace too, with comments, which should not be output by your program)

#,@;

#,@;

#,@; #,@; #,@; #,@; #,@; #,@; #,@;

#,@; #,@; #,@; #,@; #,@; #,@; #,@;

#,@; #,@; #,@; #,@; #,@; #,@; #,@;

#,@; #,@; #,@; #,@; #,@; #,@; #,@; #,@; #,@; #,@; #,@;

#,@;

Where each

corresponds to the number of correct predictions made by each of the predictors

@ corresponds to the number of branches.

First line: Provides the number of correct predictions for the **always taken predictor**

Second line: Provides the number of correct predictions for the **always non-taken predictor**

Third line: Gives the number of correct predictions for all seven variations of the bimodal predictor with a single bit of history (table of size 4,8,32…. etc). Check the first page for required variations.

Fourth line: Repeats the third line for the two-bit saturating counter based bimodal predictor

Fifth line: Repeats the third line for the three-bit counter based bimodal predictor

Sixth Line: shows the number of correct predictions for the 11 variations of Gshare predictor.

Seventh line: The number of correct predictions for tournament predictor tournament predictor.

The number of correct predictions and branches should be separated by a comma.

Every configuration of predictor should be separated by a **semicolon(;)** and a **space**.

Submissions will be checked using a script that will compare your output file to the correct output file using the UNIX `diff` tool, so if your output does not **EXACTLY** match the correct output the grading program will mark it as wrong. The TA will have to check such submissions by hand, which will result in points being deducted.

The BrightSpace submissions are due on October 21st at 11:59pm.

Another requirement is to present your project to the TA either during the lab session on **October 22nd** or during TA's office hours this week. An online signup sheet will be created for demos. **Projects without a demo will result in a zero grade.**

How to Submit

You must submit all of the following:

1.) All source code

2.) A `Makefile`

3.) A `README`, which minimally contains your Name, BU-ID (everything before the @ in your Binghamton University e-mail), B Number. Other things to include might be: what works/what doesn't, things you found interesting, etc.

These materials should be turned in as follows: (using my name and BU-ID as an example):

Dmitry's e-mail is: dponomar@binghamton.edu so his BU-ID is `dponomar`

1.) Create a new directory whose name is your BU-ID:

```
mkdir dponomar/
```

2.) Copy all relevant files into this new directory (please do not include any `.o` files, executables, **or copies of the traces**)

3.) Create a tar/gzipped archive whose name is also your BU-ID from the directory as follows:

```
tar -czvf dponomar.tar.gz dponomar/
```

(This should output name of all archived files. **Ensure that are no `.o` files, executables or trace files in this list before submission**)

4.) Submit tar/gzipped archive via BrightSpace