CS4342 Advanced Computer Architecture

Take Home Lab 2

Due – June 25 before 11:55 PM

Learning Outcomes

In this lab we will simulate a collection of branch prediction algorithms. At the end of the lab you will be able to:

- understand the concept of branch prediction, its impact on performance, and how to improve branch prediction algorithms to gain better prediction accuracy
- develop a set of branch prediction simulators and conduct a comprehensive performance study

Getting Started

This lab is derived from "Lab Assignment 2: Branch Prediction" by Andy Pimentel, Roy Bakker, and Roeland Douma at University of Amsterdam.

Step 1: Download the aca2014_assignment2_framework.tgz *Framework* given under "Assignment 2: Branch prediction" at https://staff.fnwi.uva.nl/r.bakker/teaching/aca2014/ (also available in Moodle).

Step 2: The downloaded file has a collection of branch traces (in *traces* folder) based on the following benchmarks:

- N-Queens: A chessboard of $N \times N$ tiles and N queens. The queens must be positioned in such a way that no 2 queens are on the same horizontal, vertical or diagonal line
- Fibonacci: Calculate the N-th Fibonacci number recursively
- Matmul: Do a series of matrix multiplications in different ways
- Ray Tracing: A simple ray tracer

We will use only the following branch traces for the performance analysis:

Table 1 – Branch traces to be used for performance analysis.

Benchmark	Total Branches	Unique Branches
12-queens	2,727,424	768
fib(30)	4,241,389	491
matmul	838,802	1,001
ray tracing	42,400,131	797

Step 3: Open up one of the branch trace files and study the file structure.

We are not going the use the Framework given in the .tgz file, instead we will write our own predictors.

Task 1

Step 1: Develop a branch predictor based on an 8,192 entry Branch History Table (BHT). As the BHT has 8,192 entries, table needs to be indexed based on the last 13 suffix bits of the branch address. [2 marks]

Step 2: Develop a 2-bit branch predictor based on a 4,096 entry BHT. As the new BHT has 4,096 entries, table needs to be indexed based on the last 12 suffix bits of the branch address. [2 marks]

Step 3: Develop an (2, 2) bit branch predictor based on a 1,024 entry, 4 BHTs where each entry is a 2-bit predictor. As each new BHT has 1,024 entries, table needs to be indexed based on the last 11 suffix bits of the branch address. [2 marks]

Step 4: Develop all 3 predictors within the same program, e.g., either as separate functions or classes. A user should be able to run your predictors using the following command:

```
BPAnalysis <branch predictor> <trace file>
```

Where *BPAnalysis* is a name of the executable. *branch predictor* will be one of the following:

- 1 – 8,192 BHT
- 2 – 2-bit 4,096 BHT
- 3 – (2, 2) 1024 BHT
- 4 – Custom

trace file is the path to a given input trace.

Step 5: Make sure each of your branch predictors can output the following statistics at the end of each run:

- Total no of branches:
- No of unique branches:
- No of branches correctly predicted:
- No of branches incorrectly predicted:
- Mis-prediction rate: (i.e., no of incorrect predictions / total no of branches)

Step 6: Run the 3 branch predictors developed in Steps 1 to 3 using the 4 branch traces given in Table 1. Fill up Table 2. [3 marks]

Step 7: Plot the results on a suitable graph. Comment on your observations. [3 marks]

Step 8: Develop a custom version of a branch predictor with enhanced prediction accuracy. You may consider the option of optimizing parameters of one of the 3 branch predictors as well as developing a new one. Justify your selected design choices. You have to be reasonable with the increase in hardware complexity or memory consumption. [3 marks]

A good overview of simple branch predictors is available at:

T. Yeh and Y. N. Patt, "A comparison of dynamic branch predictors that use two levels of branch history," In Proc. 20th Annual Intl. Symposium on Computer Architecture (ISCA '93), pp. 257 - 266, 1993.

Step 9: Run the custom branch predictor using the 4 branch traces given in Table 1. Add another column to Table 2 and fill up the table. The marks you gained for this performance analysis is based on how accurate your new predictor is. It will be calculated as follows: [3 marks]

Reduction in average mis-prediction rate	Marks
< 1%	0.5
< 2%	1.0
< 3%	1.5
< 4%	2.5
>= 4.0%	3

Step 10: Plot the results on a suitable graph. Comment on your observations. Justify why your Custom predictor is better. [2 marks]

Table 2 – Performance results.

	8,192 BHT	**2-bit 4,096 BHT**	**(2, 2) 1024 BHT**
12-queens			
Total no of branches:			
No of unique branches:			
No of branches correctly predicted:			
No of branches incorrectly predicted:			
Mis-prediction rate:			
fib(30)			
Total no of branches:			
No of unique branches:			
No of branches correctly predicted:			
No of branches incorrectly predicted:			
Mis-prediction rate:			
matmul			
Total no of branches:			
No of unique branches:			
No of branches correctly predicted:			
No of branches incorrectly predicted:			
Mis-prediction rate:			
ray tracing			
Total no of branches:			
No of unique branches:			
No of branches correctly predicted:			
No of branches incorrectly predicted:			
Mis-prediction rate:			

What to Submit

- Submit following files as a single .zip file
  - README with instructions on how to compile and run your program
  - Commented source code
  - Performance results, graphs, and discussion as a PDF
- Name the .zip file as lab2_<*index no*>.zip.