CSE240A Fall 2016

Branch Predictor Project

Issued: November 03, 2016
Due: November 19, 2016, 11:59PM

1 Introduction

This project is based on the Championship Branch Predictor contest [1]. The goal of this project is to evaluate the impact of the storage budget on the performance of branch predictors.

Each group will write a set of branch predictors that will consume a trace of branches (generated from real execution), based on a standard interface we will provide. Each predictor will be tested using different storage budgets, (e.g. 32K + 256 bits). The budget encompasses all storage (all tables, plus GHR, if necessary – but not PC) used by the predictor[1]. The output of your simulators will show number of branches, number of taken branches, and mispredict rate (shown as mispredicts per 1K instructions).

Each group (2 students max) will deliver a written report in which they will evaluate the impact of using different storage budgets for each predictor type, as well as analyzing which (and why) predictors work better for each budget size and trace type.

2 Scope

Branch Predictors	**Budget Sizes**	**Trace Types**
• 2-Level Local Predictor	• 8K + 64 bits	• Floating Point
• *Alpha 21264*-like Predictor	• 16K + 128 bits	• Integer
• Perceptron Predictor	• 32K + 256 bits	• Multimedia
• G-Share Predictor	• 64K + 512 bits	• Server
	• 128K + 1K bits	
	• 1M + 4K bits	

[1]That may not be the amount of storage your predictor simulator actually uses, however – for example, you may implement 2-bit predictors with a table of ints, in which case the simulator will use more memory – that's okay, we're only concerned about the memory used by the simulated branch predictor.

3 Deliverables

Written Report

- Must be formatted as an IEEE workshop short paper style and delivered as a PDF file. Maximum of 5 pages (including references). You can download the template from here: [4]
- Must be submitted to Gradescope by due date.
- Must effectively address the following aspects:
  - [+] What do you expect from increasing/reducing the budget for every predictor?
  - [+] Did the experiments confirm your expectations?
  - [+] Did you find any unexpected results? Describe them.
  - [+] Compare the performance across predictors for each budget size.
- The report should contain a plot showing the misprediction rates for all your predictors at different budget sizes. Your report must contain 4 of these plots, one per trace type.
- Do not reference or look for the predictors from the contest (either the code or the descriptions). Instead, refer to the predictors cited below that should prove helpful [8, 6, 3, 7, 9, 5]. Remember to cite the papers you read in your report.
- *Pro Tip 1*: Do not wait until to the very last moment to write the report. It is the most important aspect of the project and we will evaluate it based on the strength of your arguments/conclusions as well as the quality of presentation.
- *Pro Tip 2*: You may follow a typical paper structure: Introduction → Conceptual description of the subject(s) of analysis → Description of the experiments and results → Discussion of the results → Conclusion → References

Code

- Must be submitted via email (to the TAs) compressed in *.tar.gz* format.
- We will evaluate the last version you send us before the deadline.
- All predictors must compile and execute correctly.
- Create one single code file per predictor (4, in total).
- You should strive for the best performance for each predictor before gathering results for your report. Unusually low performing predictors will result in loss of points.
- Include a data sheet (raw, CSV format is fine) with all the results you got (4 predictors x 6 budgets x 4 traces = 96 results).
- Use of C/C++ is highly recommended, although not enforced. Your code for all your predictors should be functional and compile without errors. You should not make changes to the Makefiles we provide unless necessary. In such case, provide a justification.
- Write your own code and report. Do NOT look at code from other groups, either from the current year or previous years. Do NOT look for code in the internet. There are plenty of places you can look for predictors, but to avoid violating UCSD Policy on Integrity of Scholarship [10], please refrain from doing so.

4 Downloads

You will be provided with a starter code with an *always-predict-not-taken* dummy predictor you can use as template. You will also be provided with 4 trace files containing the branch behavior of *real* applications. All files will be made available for download in the course's webpage [2].

Acknowledgments

This project is partially based on the CSE240A Fall 2009 project '*Branch prediction contest*' by Prof. Tullsen and TA Matt DeVuyst. Traces and their interfaces were generously provided by Matt DeVuyst and Leo Porter. Additional traces generated by an execution analysis software created by Andreas Prodomou.

References

[1] *Championship Branch Predictor Contest.* URL: `http://www.jilp.org/cbp2016/`.

[2] *CSE240A - Fall 2016 Webpage.* URL: `http://cseweb.ucsd.edu/classes/fa16/cse240A-a/`.

[3] Avinoam N Eden and Trevor Mudge. "The YAGS branch prediction scheme". In: *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture.* IEEE Computer Society Press. 1998, pp. 69–77.

[4] *IEEE Manuscript Templates for Conference Proceedings.* URL: `http://www.ieee.org/conferences_events/conferences/publishing/templates.html`.

[5] Daniel A Jiménez and Calvin Lin. "Neural methods for dynamic branch prediction". In: *ACM Transactions on Computer Systems (TOCS)* 20.4 (2002), pp. 369–397.

[6] Richard E Kessler. "The Alpha 21264 Microprocessor". In: *IEEE micro* 19.2 (1999), pp. 24–36.

[7] Chih-Chieh Lee, I-CK Chen, and Trevor N Mudge. "The bi-mode branch predictor". In: *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on.* IEEE. 1997, pp. 4–13.

[8] Scott McFarling. *Combining branch predictors.* Tech. rep. Technical Report TN-36, Digital Western Research Laboratory, 1993.

[9] André Seznec et al. "Design tradeoffs for the Alpha EV8 conditional branch predictor". In: *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on.* IEEE. 2002, pp. 295–306.

[10] *UCSD Policy on Integrity of Scholarship.* URL: `https://students.ucsd.edu/academics/academic-integrity/policy.html`.