Computer Architecture

Project proposal:

Exploring Predictive Replacement Policies for Instruction Cache and Branch Target Buffer

Diane Uwacu and Fatmaelzahraa Elsheimy

Abstract

For our final project, we will implement a global history replacement policy for instruction cache and branch target buffer. The method was shown to lower instruction cache MPKI by an average of 18% over the least recently-used policy, and it showed similar improvements over several other policies. We plan to implement and validate the results in the original work by comparing against at least one of the stated methods.

1 Introduction

The instruction cache (I-cache) stores blocks of recently used instructions, and the branch target buffer (BTB) caches targets of previously taken branches. This means that the I-cache improves throughput and latency while the BTB latency due to branch target re-computation, making them both essential. The work in [3] explores efficient replacement policies at the I-cache and BTB levels. Authors propose a history-based algorithm that predicts dead blocks that should be evicted with no penalty since they are guaranteed that they won't be reused before eviction.

Authors surveyed recent work on cache replacement and noticed that a common obervation in this research area is that sequences of recently accessed instructions correlate with the likelihood of block reuse. With this knowledge, they developed the Global History Reuse Prediction (GHRP) to predict reuse behaviors in I-cache and BTB.

Authors evaluated seven recent replacement policies and discussed their effect on the I-cache and BTB hit rates and did a comparison based average MPKI values. Authors validated their work and run comparative analysis using benchmarks from the fifth Championship Branch Prediction Competition (CBP-5) [1]. With their experimental evaluation, they were able to explain why sampling-based replacement policies failed to improve things on the I-cache and BTB level.

For our final project, we would like to implement the GHRP replacement policy and run it through the CBP-5 benchmarks. Time allowing, we would like to compare our results to one of the methods that the original paper compared against.

2 Background

According to authors, not much work has been done to evaluate replacement policies on the I-cache and BTB level. Except for the work in [9] that was done in the 80s with simpler policies like FIFO, and the work by Perleberg et al [7] from the 90s, this is the first recent work to look at this problem from this angle.

Authors discuss how different prediction replacement policies affect and are affected by the I-cache and BTB. In particulat, they showed how a sampling-based prediction policy that works well for PC-based dead block prediction, does not work as well in this area, due to the fact that a given PC only accesses one set at a time. In addition to the sampling-based method, authors compare their work againt the least recently-used (LRU) policy, the random policy, and the static re-reference predictor (SRRIP).

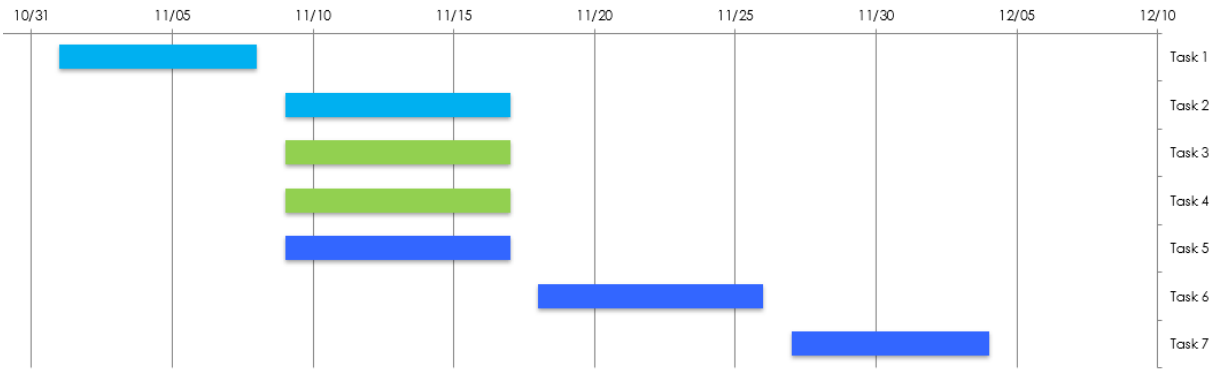
3 Motivation

Most other publications has only considered enhancing prefetching methods or software-based approaches. We are focusing our effort on improving replacement policies in I-cache and BTB based on dead-block prediction. When looking at recent work studying the effect of replacement policy, not many were found or mostly gave poor results [11, 8] . Most of them focused on different approaches such as recent static prediction [5] and using a hashed value for indexing branch predictors table [10]. Other works that studied dead-block prediction mainly focused on using it in power reduction [2], cache optimization [4] or bypassing [6]. Hence, our efforts will be focus on studying the use of dead block prediction in the I-cache and BTB. Different algorithms will be tested to be able to adapt the dead block information to BTB and I-cache replacement using open-source simulator.

4 Proposed technique

1. Find a proper open-source simulator that could be used to study BTB and I-cache.
2. Modify the simulator by adding appropriate code to collect the necessary data for studying
3. Implement a replacement algorithm that uses the past history to predict dead-blocks in cache and BTB
4. Implement a 64kB I-cache with a 64B block size.
5. Implement three prediction tables with 4096 entires and a two-bit counter per entry- it will be used for predicting the dead blocks and dead entries in BTB.
6. Measure performance of our modified simulator using LRU as baseline and studying MPKI values
7. Compare our modified simulator with another state of art approach by studying the MPKI values

5 Estimated timeline



References

[1] The 5th jilp championship branch prediction competition (cbp-5). 2016.

[2] J. Abella, A. González, X. Vera, and M. F. P. O'Boyle. Iatac: A smart predictor to turn-off l2 cache lines. *ACM Trans. Archit. Code Optim.*, 2(1):55–77, Mar. 2005.

[3] S. M. Ajorpaz, E. Garza, S. Jindal, and D. A. Jiménez. Exploring predictive replacement policies for instruction cache and branch target buffer. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, page 519–532. IEEE Press, 2018.

[4] An-Chow Lai and B. Falsafi. Selective, accurate, and timely self-invalidation using last-touch prediction. In *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, pages 139–148, 2000.

[5] A. Jaleel, K. B. Theobald, S. C. Steely, and J. Emer. High performance cache replacement using re-reference interval prediction (rrip). *SIGARCH Comput. Archit. News*, 38(3):60–71, June 2010.

[6] T. L. Johnson, D. A. Connors, M. C. Merten, and W. . W. Hwu. Run-time cache bypassing. *IEEE Transactions on Computers*, 48(12):1338–1354, 1999.

[7] C. H. Perleberg and A. J. Smith. Branch target buffer design and optimization. *IEEE Trans. Comput.*, 42(4):396–412, Apr. 1993.

[8] M. K. Qureshi, D. N. Lynch, O. Mutlu, and Y. N. Patt. A case for mlp-aware cache replacement. In *33rd International Symposium on Computer Architecture (ISCA'06)*, pages 167–178, 2006.

[9] J. Smith and J. Goodman. Instruction cache replacement policies and organizations. *IEEE Transactions on Computers*, 34(03):234–241, mar 1985.

[10] D. Tarjan and K. Skadron. Merging path and gshare indexing in perceptron branch prediction. *ACM Trans. Archit. Code Optim.*, 2(3):280–300, Sept. 2005.

[11] C. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, and J. Emer. Ship: Signature-based hit predictor for high performance caching. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 430–441, 2011.