Final Report

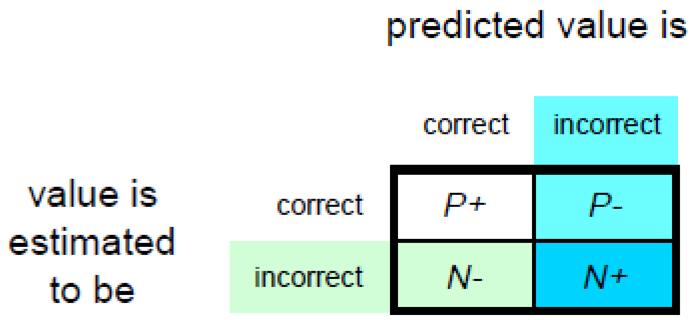
Student Name: Krishna Kabi   Student ID: 862255132
Student Name: Sheenam Gupta   Student ID: 862254462

Below are the assumptions considered for designing of the predictor:

Assumption: An incorrect prediction is considered as:

i) We decide to predict and predicted value (PV) != actual value (AV)
ii) We decide to not predict and predicted value (PV) = actual value (AV)

Metric:



		predicted value is	
		correct	incorrect
value is estimated to be	correct	*P+*	*P-*
	incorrect	*N-*	*N+*

P_corr= no. of times we choose to predict, and the prediction was right

P_incorr= no. of times we choose to predict, and prediction was wrong

NP_corr= no. of times we choose to not predict and we were right (PV!=AV)

NP_incorr= no. of times we choose to not predict, and we were not right (PV=AV)

We found multiple conventions are followed to evaluate the accuracy, so we collated 3 of them to measure our predictor performance.

$$\text{Accuracy (only considering P\_corr)} = \frac{\text{P\_corr}}{(\text{P\_corr} + \text{P\_incorr})} \quad \text{------------(1)}$$

$$\text{Accuracy (P\_corr and P\_incorr)} = \frac{(\text{P\_corr} + \text{NP\_corr})}{(\text{P\_corr} + \text{P\_incorr} + \text{NP\_corr} + \text{NP\_incorr})} \quad \text{----(2)}$$

$$\text{Coverage} = \frac{\text{P\_corr}}{\text{P\_corr} + \text{NP\_incorr}} \quad \text{-----------(3)}$$

Accuracy: A high accuracy represents a high ratio of correct predictions which save cycles) over incorrect predictions (which cost cycles) but usually also means few overall prediction attempts (i.e., the higher the accuracy, the more conservative the predictor).

.

Coverage: A high coverage, on the other hand, represents a good exploitation of the existing potential, which normally translates into many prediction attempts

The accuracy and the coverage are antagonistic, meaning that tuning a predictor to increase either one almost always decreases the other. We try to balance this and achieve both high values.

1. **Your approach towards converging to a load value predictor design (Why did you design it that way?)**

Ans:

Approach 1 (Using Prediction outcome history)

Simple Predictor: We started with implementing a simple predictor according to the paper "Value Locality and Load Value Prediction" with the following configuration. In this we use LVP (Load Value Prediction) Table with 1024 entries and history depth of 1 bit, we also used LCT (Load Classification Table) with 256 entries with 2 bits entry (Saturating Counter) as shown in table below.

	LVP Table		LCT	
LVP Unit Configuration	Entries	History Depth	Entries	Bits/Entry(Saturating Counters)
Simple	1024	1	256	2

Table 1

We did not choose to use CVU (Constant Verification Unit), because in the traces we are using just the load values and CVU entries are kept coherent with the main memory. Had we seen any store instruction, the role of CVU would then be evident by deleting the matching entries from it when store instruction is encountered.

Prediction Outcome History:

Fortunately, making no prediction does not incur a cycle penalty, meaning that it is better not to make a prediction than to make an incorrect prediction. Consequently, identifying predictions that are likely to be incorrect and inhibiting them can reduce the number of penalty cycles and thus improve the predictor performance.

Note that in cache terminology, direct-mapping implies the presence of
tag bits. However, unlike caches load value predictors do not have to be correct all the time, meaning that tags are not mandatory.

Since above one gives considerable accuracy we thought if we append history of our success or failed prediction, our confidence to predict would further increase.

This approach includes choosing the optimal threshold, history depth and no. of lines in LVP Table to increase the accuracy.

In this case, we use Last Value Predictor + Prediction Outcome History

The reason to choose this, is because this gives us more confidence to predict or don't predict as compared to simple predictor.

The new configuration we use is as shown below:

LVP Table		CE (confidence estimator)		Threshold
Entries	History Depth	Entries	Bits/Entry	
2^n	S-bit	2^s	T-Bits	>th

Table 2

In the above table n is the number of lower bits of PC address used for LVP predictor.

2. The design and implementation of your predictor(s) (How did you design it?)

Ans: The architecture of our design is well explained by the below figure

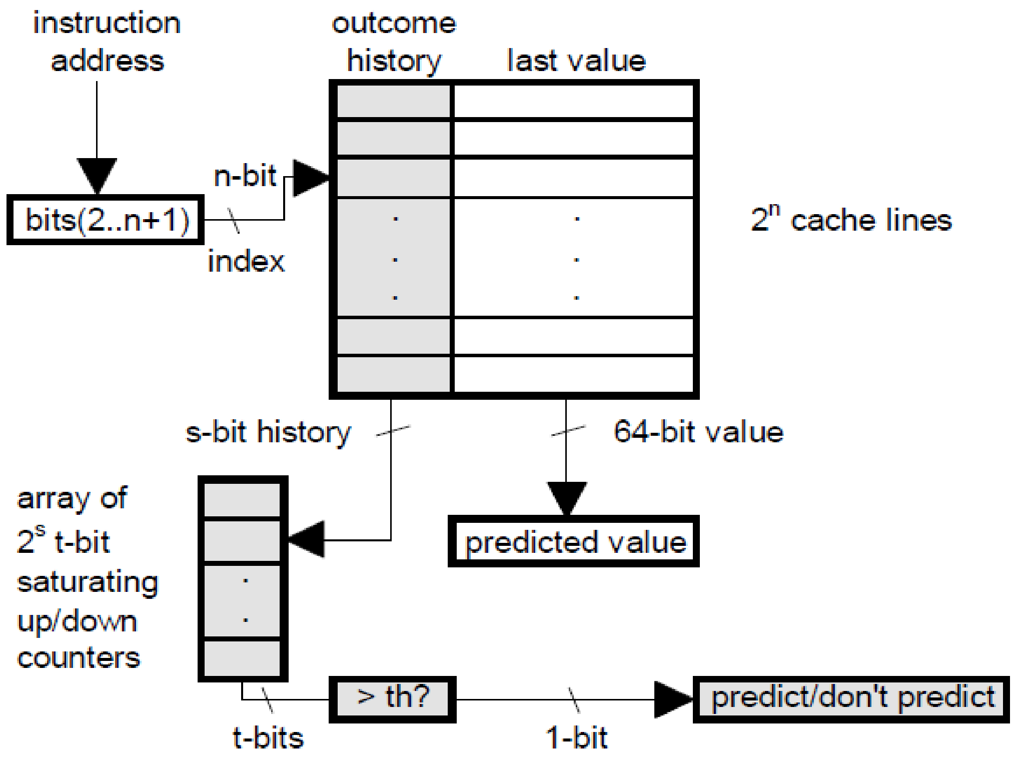


Figure 5

- The predictor consists of Last Value Predictor (LVP) and prediction outcome history-based confidence estimator.
- For each outcome history is a saturating up/down counters are built which are incremented if the prediction was correct and decremented otherwise.
- Prediction is made only if the value of the counter associated with the current load instruction's prediction outcome history is above the preset threshold.
- A new bit is shifted into the prediction outcome history of the selected cache line and the oldest bit is shifted out (lost). If the true value is equal to the value in the cache, a one is shifted in, otherwise a zero is shifted in. Finally, the value in the cache is replaced by the true value.
- Here each line of the predictor contains its own saturating counter, meaning that the confidence of each line is measured separately.
- If multiple loads alias in the same predictor line, the resulting counter value is a combination of the confidences of the loads that share the line. This allows instructions with the same prediction history to share confidence counters.
- By letting saturating counters record the number of correct predictions that followed each history pattern in the recent past, the counter values dynamically assign a confidence to each history and thus continuously adjust which patterns should be followed by a prediction and which ones should not.

3. Your experimental setup/methodology (How did you test your design?)

Ans: We tested our design by using traces built with Pin tool that gives output of only load value in the trace file.

Traces: We used our Lab1 to generate traces for testing. In this to test the accuracy of our design we identified PC addresses that were repeating after a particular interval. For example, in the trace we used "pinatrace_1st_test.out" we identified that PC Address **0x7faf70f308ac** is repeating after every 19 instructions so, we are using this to test our predictor.

Also, we generated another trace file to generate pattern for particular address in the trace file "pattern_trace4.txt" The patterns were made as shown below.

PC Address	Load Value Pattern
0x7faf70f308ac	0-1-2-3
0x7faf70f30885	0-1-2
0x7faf70f32dc4	0-1-2
0x7faf70f32c67	0-1
0x7faf70f32c52	0-1
0x7faf70f32c43	0-1-2-3

Initially we ran for below configuration:

LVP table: indexed with lower 10 bit of PC address.
LVP history depth: 4 bits
Confidence estimator (Saturating counter): 2^4 entries
Confidence estimator (entry bits): 4

But to get a good measure of accuracy and coverage, we tweaked all parameters and ran 312 different configurations varying each parameter except entry bits of saturating counter (keeping it at 4 bits).

Why saturating counter bits kept constant?
The reason for sticking to 4 bits in saturating counter, is we tried, testing for higher values, but that would have put unnecessary load on the total size of the predictor. And with 4-bits we are anyway getting good accuracy, so we avoided using higher bits for it.

Since, running 312 configurations took more time, for simplicity we switched to 60 different configurations as given below.

LVPT_index: [10, 12, 14, 16]
LVPT_hist_bit: [4,8,10]
Counter_bit: 4
threshold: [6,8,10,12,14]
Penalty: 2

The reports for each configuration for different traces is available in "Reports" folder.

Parameter tuning testing: Since an incorrect prediction generally followed by more incorrect predictions, we thought to try with different values of penalty when our prediction was incorrect. So, instead of decreasing the counter by 1 for each misprediction, we tried to decrease by 2 then 4. The findings for each are showed later.

4. Evaluation and analysis of your predictor (Does your design work?)

Below tables show the accuracy (when we only consider P_corr as given by first equation). A detailed report with other parameters can be found under Report's folder.

For Simple Predictor: (with default penalty i.e 1)

Traces	Configuration	Accu (only using P_corr)	Coverage
pinatrace_dwong.out	LVPT= 10, LCT= 8	78.176	89.469
pintrace3.out	LVPT= 10, LCT= 8	74.484	81.337

pintrace4.out	LVPT= 10, LCT= 8	75.919	82.977
pintrace5.out	LVPT= 10, LCT= 8	73.254	81.092
pattern_trace4.out	LVPT= 10, LCT= 8	77.621	88.695

For Predictor History table: (with default penalty i.e 1)

Traces	Configuration	Accu (only using P_corr)	Coverage
pinatrace_dwong.out	LVPT= 16, LVPT_hist_depth= 10, CE_bit= 4, thresh= 14	99.05	90.707
pintrace3.out	LVPT= 16, LVPT_hist_depth= 10, CE_bit= 4, thresh= 14	98.913	89.446
pintrace4.out	LVPT= 16, LVPT_hist_depth= 10, CE_bit= 4, thresh= 14	99.023	91.058
pintrace5.out	LVPT= 16, LVPT_hist_depth= 10, CE_bit= 4, thresh= 14	98.807	89.321
pattern_trace4.out	LVPT= 16, LVPT_hist_depth= 10, CE_bit= 4, thresh= 14	99.217	88.718

The second predictor shows significant increase in both accuracy and coverage.

The below plot shows how the accuracy varies with different configuration.

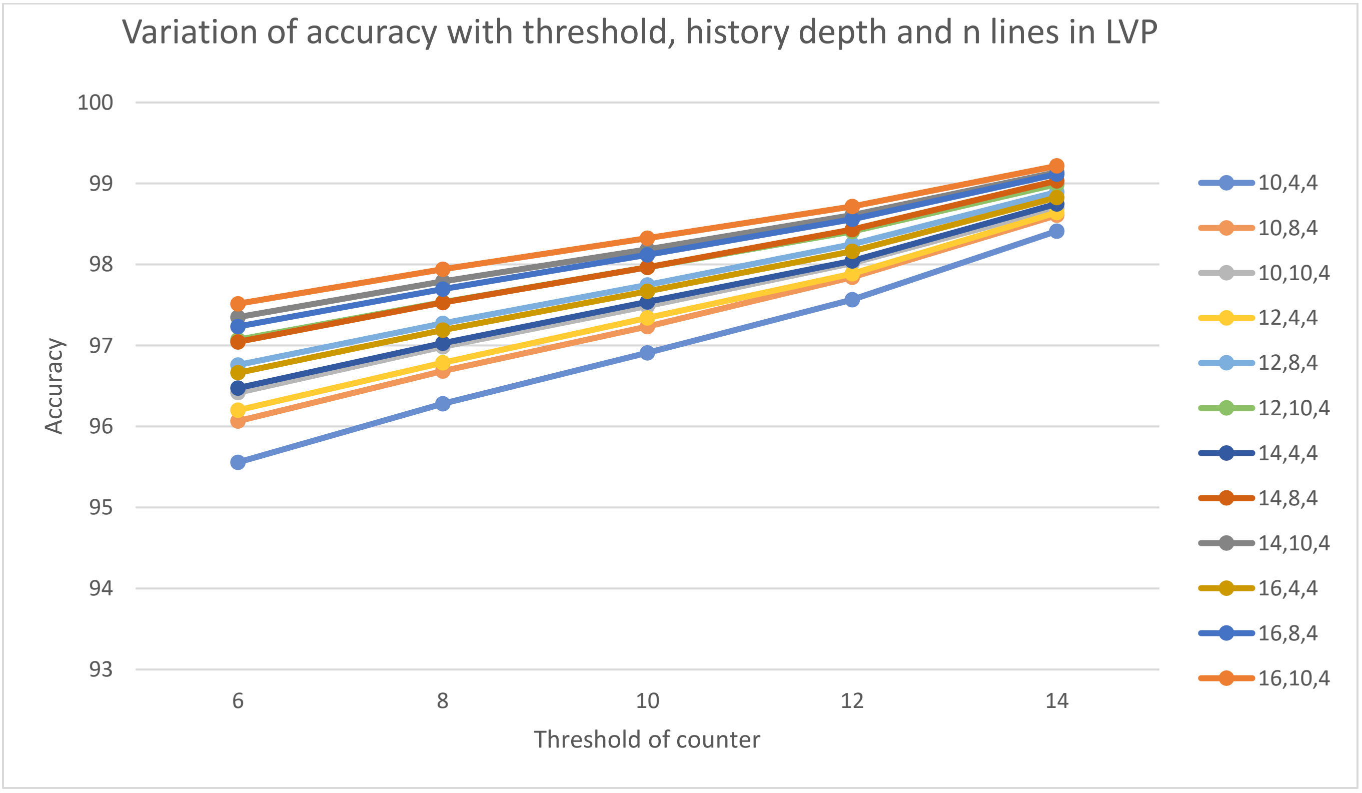


Figure 6: Each point in the plot shows: LSB bits used to index LVPT, depth of history and counter size.

Analysis:

1. As the threshold increases, we take less risk and only predict at more certainty, so our accuracy increases.
2. Furthermore, each with line increasing from bottom-up, we see when more no. of lower bits is used to index into table, accuracy increases, due to less aliasing effect and more prediction.
3. We also notice, even with same no. of LSB indexed, with increase in history depth the accuracy increases. This is obvious, since with more history we become more certain when to predict or not.

We also tested for Penalty, how tuning the penalty parameter affects accuracy.

The below plot shows how accuracy varies with increase in penalty. We can see that, as penalty increases from 1 to 2, accuracy also increase but when penalty is 4, it slowly starts to decrease meaning with more penalty the accuracy at higher threshold results in more mispredictions.

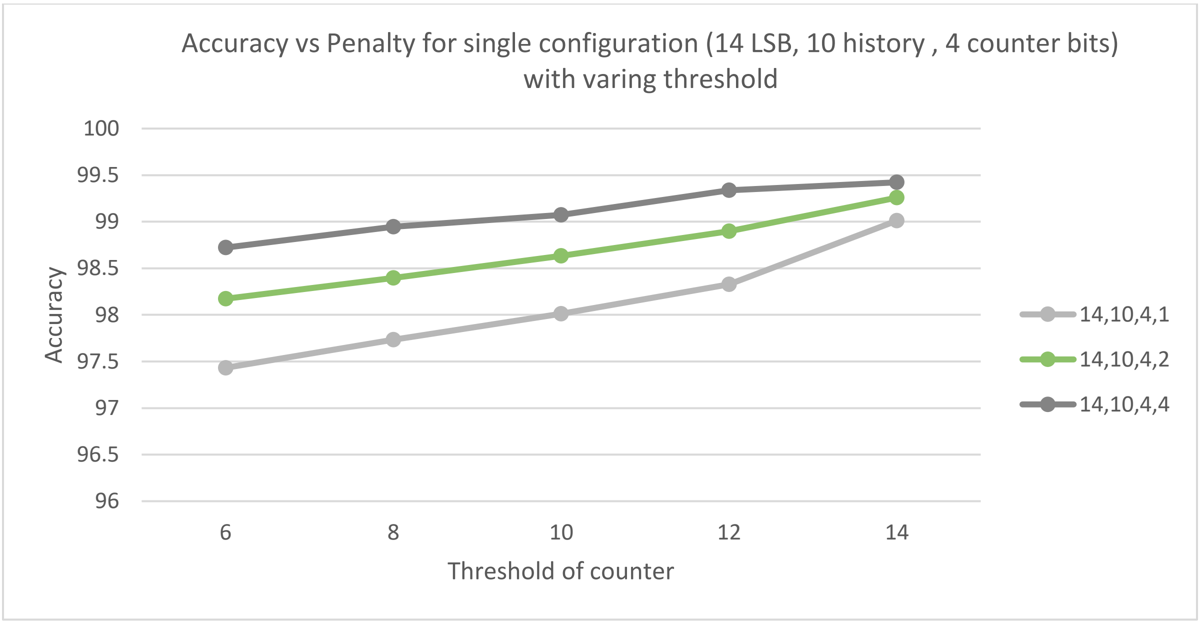


Figure 7

Similarly, below plot shows, how penalty affects for different configuration of predictors.

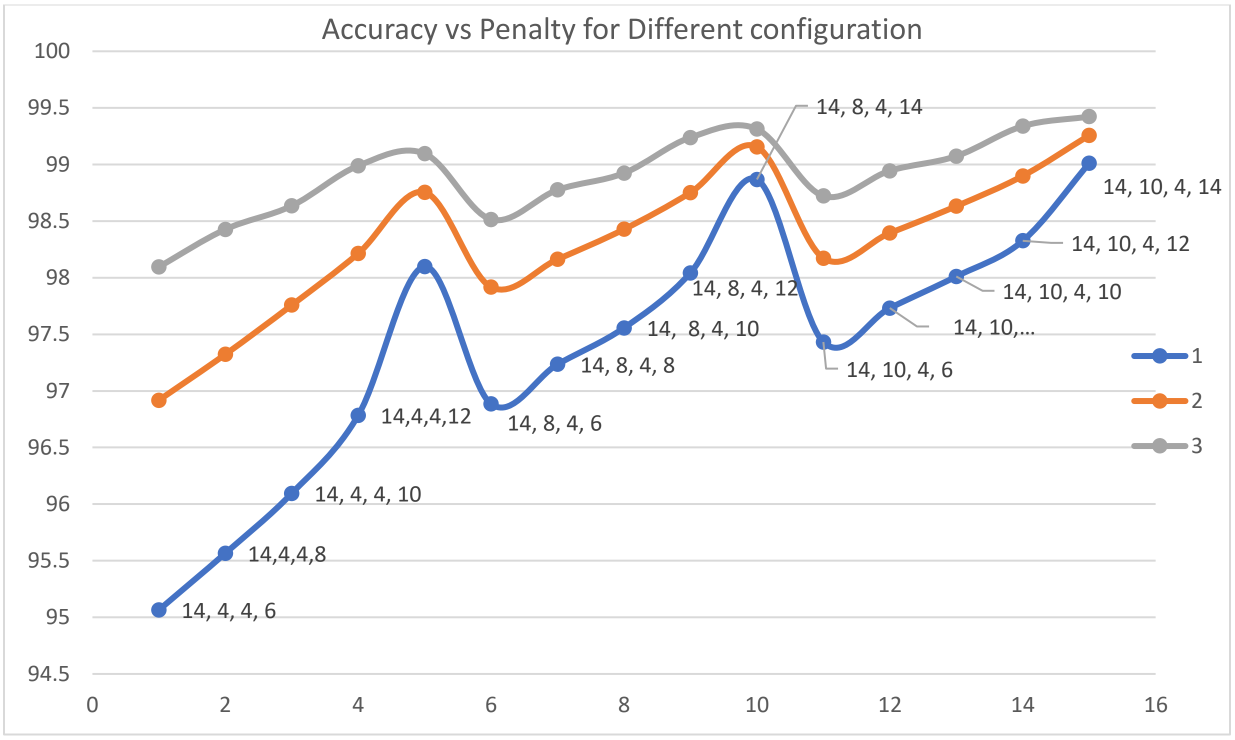


Figure 8: The labels highlighted on plot are same for each line vertically and not shown for clarity. Each configuration shows LSB bits, history bits, counter bits and threshold.

Conclusion:

- Our predictor shows both high accuracy and high coverage. While all configurations give good accuracy, there is usually a tradeoff between choosing what threshold, what lower bits we want to pick. As shown in the table, a high accuracy is always ensured with larger bits, but this might have a space overhead, so we can think of choosing lower 10 bits and having higher threshold and greater depth to achieve nearly same amount of accuracy.

- Threshold: A wise pick for threshold is critical here. Since, too high threshold will enable the predictor to take less risk so NP_incorr would increase since, we end of not predicting all the correct ones too.

  Similarly picking too low threshold would increase P_incorr, we end up more values which should not have been predicted.

Future Scope: While our predictor gave a good accuracy, it mostly depends on the kind of trace we run. Our predictor will work very well with constant sequences. For frequent pattern sequence like 12341234.. or 123123… etc , a Last 4 value predictor, would give better result than last value predictor. The architecture would be same as we did in our case but extended further to replicate 4 such predictors having 4 LVP tables and 4 confidence estimators 1 for each. A predictor in this case is only chosen when the confidence of it is > threshold. Such predictor was in progress but could not be included due to time constraint.

References:

[1] Lipasti, Mikko & Wilkerson, Chris & Shen, John. (1996). Value Locality and Load Value Prediction.. Operating Systems Review - SIGOPS. 30. 138-147. 10.1145/248208.237173.

[2] Burtscher, Martin & Zorn, Benjamin. (1999). Load Value Prediction Using Prediction Outcome Histories.

[3] Improving Context-Based Load Value Prediction, by Martin Burtscher Dipl.-Ing., Eidgenössische Technische Hochschule, 1996