CS246 – Homework #3

Matheus S. Farias

School of Engineering and Applied Sciences, Harvard University

Spring, 2023

Abstract

This document presents my solutions for homework 3 of CS246 taught in Spring 2023.

Contents

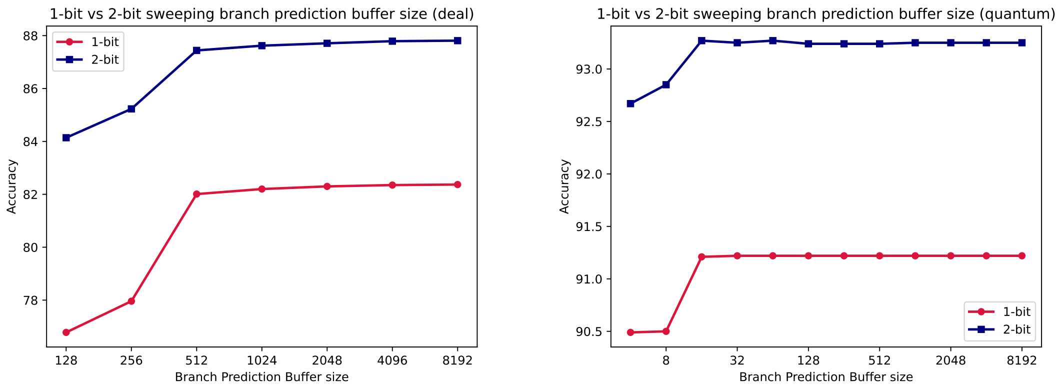
§1 Pin Assignment: Branch Predictor

(a) Code is presented in `hw3_a` folder.

(b) Code is presented in `hw3_b` folder.

(c) The processor model we have in the cluster is Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz. According to this link, Intel Xeon E5-2600v4 “Broadwell-EP” processors use a 10-way Branch Prediction Unit Target Array. This implementation differs from what we did in b) because it uses an associative history register table (AHRT). As explained in the paper suggested as the reading for this problem[1], when a conditional branch is to be predicted, the branch’s entry in the AHRT is located first. If the branch has an entry in the AHRT, the contents of the corresponding history register are used to address the pattern table. If the branch does not have an entry in the AHRT, a new entry is allocated for the branch.

(d) See Figures 1, 2, 3, and 4. To get these plots, I requested 8 cores of Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz node in the FASRC (I requested 8 because it was really hard to get a node, and 1 core didn’t give me enough memory to run the

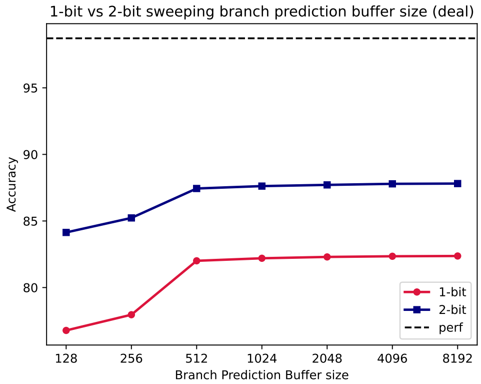


(a) Deal

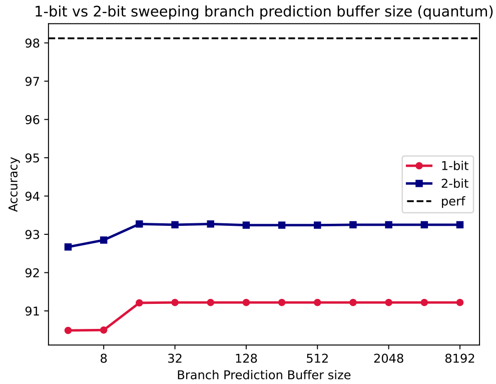
(b) Quantum

Figure 1. 1-bit vs 2-bit sweeping branch prediction buffer size.

experiments). Finally, to get the average branch misprediction ratio, I requested 32 cores as recommended in homework 1 and ran `likwid-perfctr` with the flag `-g BRANCH`. Notice that we should use the metric "ratio" according to this link. This result is added as a dashed line in the plots and can be seen in Figure 5. I decided to separate plots with and without the horizontal line representing the accuracy from perf for a better visualization. First, in Figure 2, we can see that as we increased the number of bits from the 1 to the 2-bit method, the accuracy increased and got closer to the actual prediction method the cluster has, literature says if we use more than 2-bits, we don't have significant improvements with respect to 2-bits. Also, after a certain point, increasing the buffer size doesn't improve accuracy. Now, in Figure 4, as we increased the number of bits of the history table, we increased the accuracy. In the `deal` task, the prediction method used in the cluster is clearly better. However, for the `quantum` case, both our method and the cluster's has very good accuracy. As the analysis from `likwid` is an average of measurements, we would have to get the standard deviation to have a clear understanding of what's happening. In my opinion, probably the uncertainty here would include our method and then we cannot determine which one is better, this argument is based on the small difference in the accuracy between different methods. Also, in the `quantum` case, the accuracy was constant with the number of entries whereas in `deal`, we had accuracy improvements as we increased the number of entries. All these analyses are with respect to the range of values I got and should be generalized with caution.

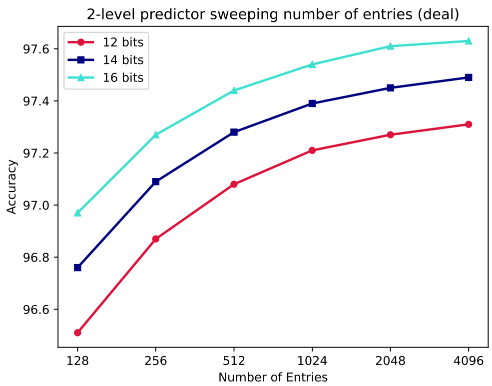


(a) Deal

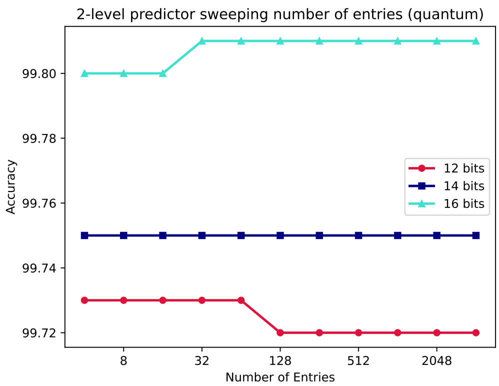


(b) Quantum

Figure 2. 1-bit vs 2-bit sweeping branch prediction buffer size with perf value.

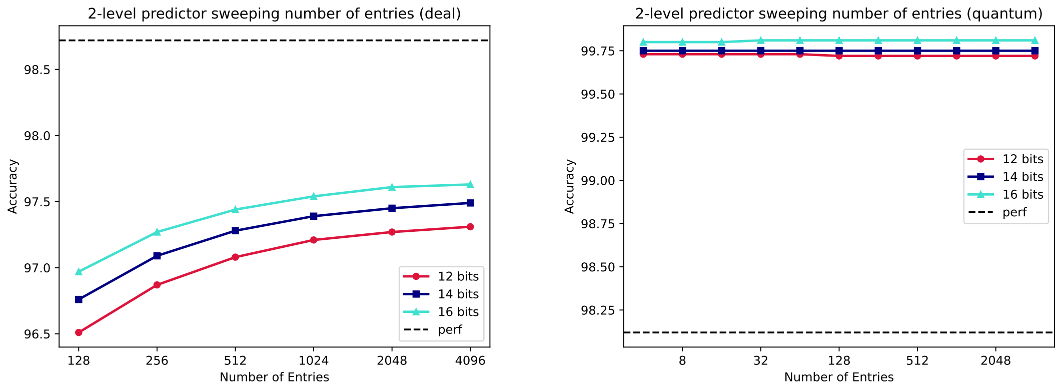


(a) Deal



(b) Quantum

Figure 3. 2-level predictor sweeping number of entries.



(a) Deal **(b)** Quantum

Figure 4. 2-level predictor sweeping number of entries with perf value.

Metric	Sum	Min	Max	Avg
Runtime (RDTSC) [s] STAT	583.7344	18.2417	18.2417	18.2417
Runtime unhalted [s] STAT	25.4150	1.930316e-06	25.3931	0.7942
Clock [MHz] STAT	41087.9516	1193.4309	2983.7648	1283.9985
CPI STAT	136.9076	0.4204	10.6361	4.2784
Branch rate STAT	7.9655	0.1359	0.2904	0.2489
Branch misprediction rate STAT	0.8411	0.0017	0.0615	0.0263
Branch misprediction ratio STAT	3.2493	0.0121	0.2189	0.1015
Instructions per branch STAT	131.8206	3.4433	7.3608	4.1194

(a) Deal

Metric	Sum	Min	Max	Avg
Runtime (RDTSC) [s] STAT	592.1120	18.5035	18.5035	18.5035
Runtime unhalted [s] STAT	25.7373	7.017766e-06	25.7275	0.8043
Clock [MHz] STAT	40576.4137	1195.6181	2923.7036	1268.0129
CPI STAT	137.5969	0.4240	8.9145	4.2999
Branch rate STAT	8.1982	0.1883	0.3356	0.2562
Branch misprediction rate STAT	0.9941	0.0027	0.0542	0.0311
Branch misprediction ratio STAT	3.8807	0.0112	0.2200	0.1213
Instructions per branch STAT	127.9005	2.9795	5.3105	3.9969

(b) Quantum

Figure 5. `likwid-perfctr` results for branch misprediction rate.

Cycle 74

	Reorder Buffer / Instruction Status Iter 1					Reorder Buffer / Instruction Status Iter 2				
Instructions	**ROB Entry #**	**IS**	**EX**	**WB**	**CMT**	**ROB Entry #**	**IS**	**EX**	**WB**	**CMT**
LD F0, 0(R1)	1	1	2-3	4	5	8	10	11-12	13	43
DIVD F2, F0, F6	2	2	5-19	20	21	9	11	37-51	52	53
LD F6, 8(R1)	3	3	4-5	6	22	10	12	13-14	15	54
DIVD F6, F6, F2	4	4	21-35	36	37	11	13	53-67	68	69
SD F6, 16(R1)	5	5	37-38	39	40	12	14	69-70	71	72
DADDI R1, R1, #-32	6	6	7	8	41	13	15	16	17	73
BNEQZ R1, LOOP	7	7	9	10	42	14	16	18	19	74

Figure 6. Instruction status at the last cycle (cycle 74)

§2 Tomasulo with Speculation

You can see the solution cycle-per-cycle in the file `cycle_solution.pdf`. Figure 6 presents the instruction status at the last cycle (cycle 74).

References

[1] Tse-Yu Yeh and Yale N. Patt. Two-level adaptive training branch prediction. In *Proceedings of the 24th Annual International Symposium on Microarchitecture*, MICRO 24, page 51–61, New York, NY, USA, 1991. Association for Computing Machinery.