ECE552: Computer Architecture

University of Toronto
Faculty of Applied Science and Engineering
Lab 2

Prepared By	Danja Papajani (1002136217), Kathryn Tremblay (1003286639)

Our microbenchmark consisted of two separate inner for-loops, one with 6 iterations and one with 10 iterations, within an outer for-loop of 100,000 iterations. The first inner for-loop of 6 iterations has an expected pattern of TTTTTTN. Since the width of our 2-level BHT is 6 bits, we categorize this as a "steady state" and expect an MPKI close 0, excluding the addition of the predictor training. The resulting MPKI was 0.557, which confirms our estimate. The second inner for-loop of 10 iterations has an expected pattern of TTTTTTTTTN, with a total instruction count of 4807205. Since the width of our 2-level BHT is 6 bits, but we iterate 10 times, we expect about 1 mispredictions and MPKI of about ((1*100,000)/4807205)*1000 = 20.80, excluding the addition of the predictor training. The resulting MPKI was 21.73. Including both loops, with 7607205 total instructions, we expect a final MPKI of about (1*100,000)/7607205)*1000 = 13.14. Our resulting MPKI for the 2-level predictor is 13.380, so our microbenchmark has verified our 2-level predictor. We used the flag **-O0** to compile our microbenchmark, in order not to perform any optimizations, and we iterated our microbenchmark 100000 times to get accurate statistics.

Table 1: Number of Mispredicted Branches and MPKI for each branch predictor and benchmark

	2-bit		**2-level**		**Open-ended**	
Benchmark	NMP	MPKI	NMP	MPKI	NMP	MPKI
astar	3695830	24.639	1785464	11.903	946837	6.312
bwaves	1182969	7.886	1071909	7.146	903071	6.020
bzip2	1224967	8.166	1297677	8.651	1232239	8.215
gcc	3161868	21.079	2223671	14.824	768064	5.120
gromacs	1363248	9.088	1122586	7.484	886792	5.912
hmmer	2035080	13.567	2230774	14.872	1739404	11.596
mcf	3657986	24.387	2024172	13.494	1515518	10.103
soplex	1065988	7.107	1022869	6.819	751395	5.009

Open Ended Branch Predictor

For our open ended branch predictor design, we decided to implement a Dynamic Branch Predictor with Perceptrons [1]. We use two tables to implement this design, one is the perceptron table which contains the perceptrons and the other is the global branch history register (GHR) which contains history of branch predictions. We initialize all weights in the perceptron table to 0 and all GHR elements to 1. We first index into the perceptron table using the hash of PC%[NUM_OF_PERCEPTRONS] as recommended by

[1]. The output perceptron, y, is calculated by executing the dot product of the weights of the (extracted) perceptron with the GHR predictions. If this output is negative we predict NOT_TAKEN, otherwise we predict TAKEN. If our prediction is incorrect, the weight of the perceptron is updated by adding its current value to the product of the actual outcome (resolveDir, where taken = 1 and not taken = -1) with the value indexed at the GHR. The storage requirements were calculated by adding the number of bits used in both the GHR and the perceptron table. As per the recommendation of Jiminez and Lin, we decided on a GHR of 36 bytes [1]. Similarly, we decided on 8 bits for an individual weight vs the recommended 16 from the paper's Best History Length table. This is because we wanted to use more storage to create a larger (more entries) perceptron table in an attempt to improve accuracy. This gives a total weight of (400 entries)*(36 weights)*(8 bits/weight) + (36 elements)*(8 bits/element) = 115488 bits.

2-Level Branch Predictor CACTI Results

We modified cache.cfg to model the PHT and Private predictor tables, because they are similar to a cache in that they are indexed by the branch PC

Configuration File	Area (mm^2)	Access Latency (ns)	Leakage Power (mW)	Modified Config Param Size
2-level-bred-1.cfg	0.0020849	0.331825	0.404632	size (512B)
2-level-bred-2.cfg	0.000541368	0.260657	0.0882322	size (128B), block size (2B)
TOTAL	***0.0026227***	***0.5925***	***0.49286***	

Open-ended Branch Predictor CACTI Results

We modified pureRAM.cfg to represent the Global History Register in the Perceptron Branch Predictor since it is not tagged like a cache. We changed the size to 64B because that's the minimum size for the CACTI, but we only used 36B. We modified cache.cfg to represent the table of perceptrons, since it is indexed by the branch PC, similar to a cache.

Configuration File	Area (mm^2)	Access Latency (ns)	Leakage Power (mW)	Modified Config Param Size
open-ended-bpred-1.cfg	0.00019164	0.1178	0.029967	size (64B)
open-ended-bpred-2.cfg	0.04281	0.5392	9.96934	size (14400B)
TOTAL	***0.043002***	***0.5392***	***9.9993***	

Division of Work

KC Tremblay: worked on all deliverables (½ of workload)

Danja Papajani: worked on all deliverables (½ of workload)

[1] https://www.cs.utexas.edu/~lin/papers/hpca01.pdf