ECE552 – Lab Assignment 3 Report: Dynamic Branch Prediction

Andreea Varlan & Ben Pak

Microbenchmark (detailed description found in comments in mb.c file)

I decided to use the compilation option 'O0' when compiling my benchmark since with any compilation flags set, the optimizations would filter out the branch/jump instructions that I wanted to analyze. To test the performance of the 2Level BP, I used the following loops:

- For loop iterated through a large number to see a traceable amount of conditional branches
- If statement that runs 6 iterations and branches on the 7th; as the 2Level BP uses 6 history bits, it will await 6 mispredictions before it changes its mind, and therefore have no difficulty handling this loop (MPKI = 0.095, close to **no** mispredictions)
- If statement that runs 8 times and branches on the 9th; in this case, the T/N sequence pattern is too long for 2Level BP to handle, and it will mis predict more often which is reflected in an MPKI=9.921

Another thing to note is the fact that the 2Level uses private tables with private history registers, thus we see no biasing and addition of taken branches when analyzing the performance in the first if statement. If it had uses global history registers, we would see two Taken branches in the beginning, as the for loop branched once and then the first if statement branches right after it. In that case, the pattern would be too long to be captured by the 2Level BP.

Table 1: Misprediction Rate and Num_mispredictions of each Branch Predictor across the 8 Benchmarks

	Astar	Bwaves	Bzip2	Gcc	Gromacs	Hmmer	Mcf	soplex
2bitsat	3697270 24.648	1186188 7.908	1227534 8.166	3179496 21.079	1363971 9.088	2035534 13.567	3658361 24.387	1066980 7.107
2level	1790480 11.937	1093858 7.292	1386521 8.651	2305979 14.824	1174680 7.484	2232325 14.872	2033681 13.494	1023862 6.819
openend	800021 5.333	911863 6.079	1103983 7.36	760818 5.072	808725 5.391	1852187 12.348	1575464 10.503	781122 5.207

The average MPKIs for each BP are **14.49, 10.65 and 7.162.**

CACTI

2level: There are two configuration files for the 2Level BP. First file uses the untagged pureRAM structure to describe the BHT, while the second file uses a tagged cache structure for the 8 PHT tables. The BHT includes 512 entries, each entry being 6-bit wide. Due to the constraint of the program, we must round the 6-bit entries to 1Byte, thus the total size of the configured RAM module for BHT is 512Bytes (for this we only changed the cache size parameter). The second file describes all 8 PHT table, with 64 entries, each containing a 2-bit counter. We can represent the whole structure using only the tagged array, and thus we set a cache size to 512B (8 * 1B * 64), the block size kept at 1B, but the tag size set to 2 as we use saturated counters with 2 bits.

open-ended O-GEHL: The open-ended predictor is based on the O-GEHL branch predictor that utilizes multiple predictor tables, each providing a prediction that contributes to a total signed sum that is used to generate a new prediction. For this implementation, I am using 5 prediction tables with 4096 entries containing a 5-bit signed saturated counter. The first table is indexed by the first 12 bits of PC (2^12 = 4096 – max idx), while the other 4 are indexed in using a hash function that computes an XOR between 12 PC bits and a variable number of bits from the stored branch history. The length of the history bits pattern used in each table is an approximation of a geometric series (L(i) = a^(i-1) * L(1)). The values I used for the different lengths are {0,2,4,8,16}. We updated the branch predictor whenever we encounter a wrong prediction or when the computed sum of each prediction from the tables is under a certain threshold (set equal to the number of tables for best performance yield, in our case at 5).

5 (num pred tables) * 5 (5-bit sat cntr) * 4096 (pred table entries) = 102400 bits =12800Bytes

Table 2: Statistics of the 2-level BHT and PHT structures, and O-GEHL prediction tables

Statistic	2Level-BHT	2Level-PHT*	Opened – O-GEHL
Access Time (ns) / Tag Side (ns)	0.163585	0.166405	0.363629
Cycle Time (ps)	0.117678	-	0.469473
Total dynamic read energy per access (pJ)	0.000420231	0.000192222	0.00496718
Total Leakage Power of a bank (mW)	0.195006	0.0945203	6.50934
Cache height x width (mm^2)	0.001052776	0.000584154	0.029778119

*For the 2Level-PHT, we used the statistics of the tag array only since we are not using the data array.

For the 2Level, we observe a lower dynamic read energy, leakage power and area for PHT compared to the BHT. This makes sense, since the tagged structure of the PHT allows us to use 2 bits instead of the 8 bits approximation required from the BHT with the same cache size.

The RAM module for the O-GEHL predictor uses a much higher storage space, and thus the area is ~30 times greater than that of the 2Level. We see the same increase in the access time, read energy and leakage power parameters.

For the 2Level-PHT, we used the statistics of the tag array only since we are not using the data array.

Statement of Work: Andreea implemented all three branch predictors. Andreea also wrote the CACTI configuration files, the microbenchmark and the report.