CSE 240A Project 1 Branch Predictor Contest

Hao Zhuang, A53029734
Dept. Computer Science & Engineering, UC San Diego
Email: hazhuang@eng.ucsd.edu

(a) Name of your branch predictor:
My predictor name is PWL, which is PieceWise Linear branch predictor based on [3].

(b) One sentence description of your predictor:
The predictor uses the modulus branch address and previous addresses (in global history address) to fetch and sum up trained weight values in order to make more accurate prediction.

(c) A table with all your results, including arithmetic means over all traces:
We compare Alpha [4], two-level local prediction (2-level)[5], global/local perceptron (GLP) [2], global perceptron (GP) [1][2] branch predictors and our piecewise linear (PWL) [3]

Case	Alpha	2-level	GLP	GP	My PWL
DIST-FP-1	4.517	4.147	2.023	2.454	3.041
DIST-FP-2	2.213	2.311	1.140	1.120	1.087
DIST-INT-1	6.068	14.080	6.270	7.451	6.451
DIST-INT-2	12.207	13.619	11.171	10.327	7.958
DIST-MM-1	11.737	10.813	7.700	7.680	7.380
DIST-MM-2	8.374	14.404	9.927	10.047	8.998
DIST-SERV-1	7.906	10.060	8.764	7.789	6.905
DIST-SERV-2	8.494	10.153	9.207	8.320	7.286
AVG	7.690	9.948	7.025	6.899	6.138

(d) A complete description of the predictor. Include citations of any predictor you are based on:

The predictor is based on the model of piecewise linear, which unifies the perceptron-based and path-based neural branch predictor. In my predictor, there is one global history register (GHR), one global address (GA) array and one 3-dimensional weight array (*W*).

GHR has *h* bits to hold the recent branch history of *taken* and *non-taken*. *GA* has *h* elements to hold recent branch address. The size of each element in *GA* is *m*. The 3-dimensional weight array has the size of $n*m*(h+1)$. The extra one in $h+1$ is for the bias. Each element is a training weight with 8 bits so that value stays in [-128, 127].

The processes of prediction and training follow corresponding pseudo codes in Figure 2 and Figure 3 in [3].

For the tuning of predictor, first I choose the array size from [3], h = 19, n = 1, m = 215. But I notice the Figure 7, when n ==m, it may have better results. Therefore, I sweep the n and m (n, m should be power of 2, e.g. 4, 8, 16, 32, etc.), and then find out (m=16, n = 16) is a good choice for our test cases.

At the same time, the global history register's length is h = 15. The optimal threshold is based on the paper, which is 2.14*(h+1) + 20.58.

Overall, in my program, I use *m = 16*, *n = 16*, and *h = 15*. The total size of tables I use is under the hardware budget, which is 8*16*16*(15+1)+4*15+15 = 32,843 < 33,024 = (32K+256)
(The value 8 is the bit wide of weight. The value of 4 is for GA array's element, which comes from modulo *m* is 16 or 2^4, so that it takes 4 bits to represent branch address and store in GA).

References

[1] Jiménez, Daniel A., and Calvin Lin. "Neural methods for dynamic branch prediction." ACM TOCS 2002.
[2] Jiménez, Daniel A., and Calvin Lin. "Dynamic branch prediction with perceptrons." HPCA 2001.
[3] Jimenez, Daniel A. "Piecewise linear branch prediction." ISCA 2005.
[4] Kessler, Richard E. "The alpha 21264 microprocessor." Micro 1999.
[5] Yeh, Tse-Yu, and Yale N. Patt. "Alternative implementations of two-level adaptive branch prediction." ISCA 1992.