Ringo Cheung - 1000463955
Yassir Solomah - 1000458284

Lab 2 - Branch Prediction

1 Microbenchmark

The following microbenchmark was used to test the validity of the implemented two-level PAp branch predictor.

```
int main() {
        int a, i;
        for (i=0; i<100000; i++) {
// Jump Not Taken occurs (100000-1)/x
                if((i%x) == 0) a = 10;
                a = 15;
        }
        return 0;
}
```

Value of x	Number of Mispredictions
6	1317
7	1318
8	13 816
9	12 427

Table 1: Value of $\boldsymbol{x}$ and number of mispredictions

The values shown here align with what was expected, the Pap predictor is able to accurately track x = 6 and x = 7 because only 6 bits of history are needed in both of these cases.

For x = 8, the number of occurrences is of the branch is 12499 ((100000-1)/8). This is validated by observing the difference between the number of mispredictions between 7 and 8, which is 12498 (=13816-1318).

Finally, for x = 9, the number of occurrences of the branch is 11111 (=(100000-1)/9) that are incorrectly predicted. Taking the difference between x = 7 and x = 9, we arrive at 11109.

2 Open-ended branch predictor implementation

The open-ended branch predictor used is a perceptron model with a Gshare like global history hashing to index into the perceptron table - *GPerceptron*. The perceptron model used is described in [1]. *GPerceptron* achieved an average $MKPI$ of 6.16 using 129,060 bits which is less than the 2^{17} limit imposed.

The weights and history bits are used in the prediction process as follows:

$$y = w_o + \sum_{i=1}^{n} w_i x_i \tag{1}$$

The inputs x_i are the history bits from the global history register: x_i is 1 if the history bit is 1 for taken and -1 if the history bit is 0 for not taken. w_o is a bias that is always set to 1.

Perceptrons are trained using the following algorithm:

```
if sign(y) ≠ t or |y| ≤ θ:
        for i = 0 to n do
                w_i = w_i + t x_i
        end for
end if
```

where y is the prediction made by the prediction stage above, t is -1 if the branch outcome was not taken and 1 if it was taken, w_i are the weights in the perceptron and x_i are the history bits from the global history register. The update stage has two steps:

1. Get the perceptron corresponding to the current PC and global history register using the process described in the Dimensions section. Apply the training algorithm described above to the perceptron
2. Update the global history register with the branch outcome

2.1 Dimensions and Indexing

The dimensions of the implementation of GPerceptron used are a 512 row perceptron table with 36 signed integer vectors at each row of 7 bits each which uses 129024 bits. An additional 36 bits are used for the global history register. The perceptron table is indexed using a combination of PC and the global history register which represents an innovation over the perceptron branch predictor described in [1]. The indexing is done by performing an XOR on bits [32:2] of PC with the most recent 32 bits of the global history register and then using the lower 9 bits of the result to use as an index into the perceptron table. This XOR operation with the global history register appears to reduce aliasing when compared with just using the PC to index.

3 Results

Benchmark	2bitsat (%)		2level (%)		GPerceptron (%)	
	Mispredictions	MKPI	Mispredictions	MKPI	Mispredictions	MKPI
astar	3 695 830	24.639	1 785 464	11.903	573 215	3.821
bwaves	1 182 969	7.886	1 071 909	7.146	626 184	4.175
bzip2	1 224 967	8.166	1 297 677	8.651	1 086 864	7.246
gcc	3 161 868	21.079	2 223 671	14.824	480 417	3.203
gromacs	1 363 248	9.088	1 122 586	7.484	751 612	5.011
hmmer	2 035 080	13.567	2 230 774	14.872	1 708 054	11.387
mcf	3 657 986	24.387	2 024 172	13.494	1 444 272	9.628
soplex	1 065 988	7.107	1 022 869	6.819	717 000	4.780

Table 2: Number of mispredictions and number of mispredictions per 1k instructions for all 8 benchmarks and the three different branch predictors

4 CACTI

Predictor	Area (μm^2)	Access Latency (ns)	Leakage power (mW)
2level	1 052.77	0.163585	0.195006
GPerceptron	30 306.1	0.265857	5.96382

Table 3: CACTI values for area, access latency and leakage power for the 2level and open-ended predictors

The modified value in '2level-bpred.cfg' was the size which is 512 bytes since the cache is 4096 bit. The modified values in 'open-ended-bpred.cfg' were the size which is 16128 bytes $(512 * 36 * 7/8)$, the block size which is 32 bytes since each entry contains $36 * 7/8 = 31.5$, and the bus width which is 256 since it should be 8 times the block size.

5 Statement of Work

Equal work was completed by both partners.

References

[1] D. A. Jiménez and C. Lin. *Dynamic branch prediction with perceptrons.* In *HPCA-7*, 2001.