Lab 2 Report

Microbenchmark Explanation

The two-level predictor has 6 per address history bits. This means that for branches that take the direction corresponding to 6-bit long branch history patterns or shorter, we can always predict the behavior 100% after training. This will be significantly different from the 2-bit saturating counter. Our microbenchmark includes a conditional branch on the loop counter each of mod 2 and mod 3. The 2 bit-saturation will mispredict 50% for the mod 2 branch and 25% for mod 3 branch respectively since it initialized weakly not taken. Then we add a conditional branch that will be taken every 6 iterations of the loop. Here the 2 bit-saturation will be incorrect 25% while the 2 level would predict it correctly since it has 6 history bits. The result of our benchmark is with MPKI ~40 for 2 bit sat while it is ~0 for the two level. This demonstrates the advantage of the two-level predictor.

Mispredicted Branches and (MPKI) Table

Benchmark	2 Bit Sat Missed	MPKI	2 level Missed	MPKI	Open-end (perceptron)	MPKI
Astar	3695830	24.639	1785464	11.903	844237	5.628
Bwaves	1182969	7.886	1071909	7.146	856193	5.708
Bzip2	1224967	8.166	1297677	8.615	1175082	7.834
Gcc	3161868	21.079	2223671	14.824	682181	4.548
Gromacs	1233248	19.088	1122586	7.484	907486	6.05
Hmmer	2035080	13.567	2230774	14.872	1735895	11.573
Mcf	3657986	24.387	2024172	13.494	1481939	9.88
Soplex	1065988	7.107	1022869	6.819	745470	4.97
Average	2157242	15.740	1597390	10.645	1053560	7.024

Perceptron Branch Predictor

We read several papers of branch prediction about TAGE Predictor and Perceptron Predictor. We implemented a model resembling the predictor described in “Dynamic Branch Prediction with Perceptron” by Jimenez and Lin from University of Texas at Austin. The paper described the concept of perceptrons, linear separability, implementing perceptrons and their experimental results. The showed that given the same hardware budget, perceptrons did better than Gshare, bimodal and hybrid of Gshare and bimodal predictors. The perceptron method requires, a history array of length *h* and *n* sets of weights each with dimension *h*. Referencing the paper, we selected history length h to be 62 and 1024 perceptrons. We used integers for each weight for the software implementation but since the weights are small we can use half bytes in the hardware implementation. This means that in total we require a memory budget of 60*128*16 perceptron table weights bits + 60 global history bits + 32 output bits + 64 bias bits or 123Kbits in total. We implemented the three components as follows:

1. Initialization
    - Initialize all weights in perceptron weight table and global history array to 0.

2. Predicting Branches
    - Select entry in perceptron table that is indexed by 10 least significant bits of PC
    - Compute y by multiplying weights of selected entry by history bits. In the history, TAKEN is represented by the value '+1', NOT_TAKEN has value '-1'

$$y = w_0 + \sum_{i=1}^{h} x_i w_i$$

    - Predict TAKEN if y is greater than 0, otherwise predict NOT TAKEN
3. Updating Weights (Updating Prediction Table)
    - Select entry in perceptron weight table using the same indexing scheme from PC
    - Update weights of selected entry if predicted target is not the actual result or if the confidence is lower than the threshold value 'theta': for every weight add the product of the target and the global history bit
    - Add new bit to global history vector

Area, Access Latency and Leakage Power

	2 Level Predictor (Cache)	**Perceptron (RAM)**
Area	0.00102589 mm^2	0.0229728mm^2
Access Latency (Access Time)	0.168319 ns	0.275772 ns
Leakage power	0.404632 mW	5.3099 mW

We used the RAM configuration for the Perception because all data accesses are byte aligned and the paper suggested RAM for hardware area cost estimations. For 2 level, we modified the size in bytes to 512, and we changed tag size to 6 bits. For open-ended, we modified total size in bytes to 15375, we modified to block size to be 2 bytes (1 short), bus width to 16 bits.

Work completed by each partner

Rahul read the TAGE papers and implemented a simplified TAGE predictor, Judy read the perceptron prediction paper and implemented the perceptron predictor. Everything else was done together.