Github: https://github.com/hollowknightsasd/Chenyu-Branch-predictor

The choice of the custom is Perceptron predictor with gshare.

The reason to choose this predictor is because it's easy to be implemented with relativity high accurate.

The Perceptron is based on the paper which is "Dynamic Branch Predictor" written by Calvin Lin and Daniel A. Jim´enez. The principle of the predictor is based on the one of the simplest perceptrons. s

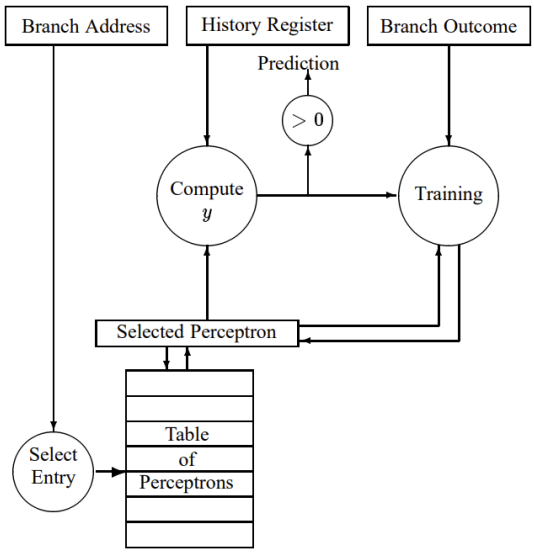
$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$

The x_i is the global history register which will record the past n bits of the branch result, the wi is the weights vector. In perceptron, -1 means NOTTAKEN, 1 means TAKEN. These two arrays do dot product to produce the output y to compare with the threshold θ, the best threshold is $\theta = 1.93 \times h + 14$, which is provided by the paper. h means the number of bits we used in the History table.

After compute the output y, if y is less than 0, it means the branch is not taken. If y is bigger than 0, it means that the branch is predicted to be taken.

During the training stage, compare the absolute value of y with threshold, if the absolute value of y is less than threshold, then it need training. Also, if the predication result is different with the actual result, it also needs training. In the training stage, compared the GHR value with the outcome. If they have the same value, the corresponding weight increased by 1, if not, the corresponding weight decreased by 1. History Register shift left and record the latest branch states.

The structure is shown below but with an addition XOR logic to do computation between Branch Address and GHR:



By testing, the result of each predictor is shown below. Except for mm_1 and mm_2, Tournament have a lower misprediction rate.

Misprediction rate			
	Gshare	Tournament	Custom(perceptron)
FP_1	1.172	0.984	0.829
FP_2	9.656	2.898	0.961
Int_1	19.331	10.338	8.300
Int_2	0.762	0.379	0.309
mm_1	11.578	1.692	1.828
mm_2	13.121	7.685	10.213