Lab Assignment 2: Dynamic Branch Prediction

Angel Serah: 1000491773 Harshita Huria: 1000980398

1. Our microbenchmark consisted of 5 iterations of a for loop within a for loop of 1000 iterations. The inner loop had a pattern of NNNNT. We verified that our two-level predictor predicted NNNNN giving a misprediction of 1 instruction in 5, just for the inner loop.
We calculated a total of 61,000 instructions based on the assembly dump. The outer loop executes 1000 times, so it has a misprediction of 1/1000 (when it exits out of the loop), not including the inner loop. With the inner loop executing 5 times for each outer loop with a misprediction of 1, we will have 1*1000+1 = 1001 mispredictions for the 1000 iterations.
This makes the MPKI = (1001/61000)*1000 = 16.409
The value that we got as MPKI from the two-level predictor was 9.629. There is a mismatch as we didn't take into account the rest of the instructions outside of the for loops, which would increase the total instruction count, thus decreasing the MPKI.
Thus the microbenchmark helped us verify the two-level predictor.

2.Table1

1: 2-bit saturation Prediction 2: 2-level Predictor 3: Open-ended predictor

	Benchmarks	B1 (astar)	B2 (bwaves)	B3(bzip2)	B4(gcc)	B5(gromacs)	B6(hmmer)	B7(mcf)	B8(soplex)
1	# of mispredicted branches	3695830	1182969	1224967	3161868	1363248	2035080	3657986	1065988
	MPKI	24.639	7.886	8.166	21.079	9.088	13.567	24.387	7.107
2	# of mispredicted branches	1785464	1071909	1297677	2223671	1122586	2230774	2024172	1022869
	MPKI	11.903	7.146	8.651	14.824	7.484	14.872	13.494	6.819
3	# of mispredicted branches	1033453	866279	1175953	752918	990132	1732977	1510555	748439
	MPKI	6.890	5.775	7.840	5.019	6.601	11.553	10.070	4.99

3.We decided to implement a version of a perceptron branch predictor. We referenced the algorithm from the report: https://www.cs.utexas.edu/~lin/papers/hpca01.pdf. In the algorithm, we defined 1 as taken and -1 as not-taken. We created three tables, one with the perceptrons and their history of weights, one with the global branch history (a sequence of 1s & -1s), and perceptron bias table. We indexed to the perceptron table using PC% (size of perceptron table). We calculated the dot product of weights of perceptron with the branch

history table. If that value is non-negative, we predict a branch to be taken. Else we return a prediction of not taken. If the predicted value is different from the actual value, we update the weights of the perceptrons by incrementing it with the product of the value at an indexed global branch history table with the result of resolveDir (1 = taken; -1= not taken).
We calculated the storage requirements by adding up the total number of bits used by the perceptron weight array, perceptron bias array and global history array. We ensured that the perceptron weight array was 16bits/weight*90perceptrons*85weights = 122400bits, perceptron bias was 16bits/weight*90perceptrons = 1440bits and global history array was 8bits*85historical prediction values = 680bits. This brought to a total number of bits =124520 bits = 15,565Bytes which was less than the storage limit.

4. We modified only two parameters for all the .cfg files given below:

Config parameter/Config file	2level-bpred-1 (1 BHT)	2level-bpred-2 (8 PHTs)	open-ended-bpred-1(Perceptron Weight Array)	open-ended-bpred-2(Perceptron Bias Array)	Open-ended-bpred-3 (Global History Array)
Size	64	128	15300	180	85
Block Size	1	2	170	2	1

	Area	Access Latency	Leakage Power
Two-level	383.60 μm^2	0.235609ns	0.0645814mW
Open-ended	0.02497 mm^2	0.411038ns	5.3326 mW

5. Work division:
- Angel: Worked on all deliverables (0.5 of total workload)
- Harshita: Worked on all deliverables (0.5 of total workload)