I replaced the original gshare predictor with a gskewed predictor. Instead of using just one table to make a prediction, I used three separate tables. The final prediction was determined by whether two tables predicted "taken" or vice versa. Since these are separate tables, the indices for the tables are created using different hashing functions. In my code, I detail the various hashing functions I tested. Ultimately, the hashing function I chose involved a combination of shifting, XORs, and ORs between b.address and history to produce the indices. I also decided to keep the history and TABLE_BITS at 15 bits to keep the overall storage well below a few megabytes, while still achieving better performance than the gshare predictor. The total storage is calculated as 3 * 2^15, which equals 98,304 bytes, or slightly less than 0.1 megabytes. Although increasing the history and TABLE_BITS to 20 bits noticeably improved performance, it caused the overall storage to exceed several megabytes. I could have also tried increasing the number of tables, but this would have required even more storage.