1.0 Overview

In this report I go over the simulation result for prediction accuracy of four different predictors against a hierarchical predictor for four different benchmarks (go, gcc, fpppp and vpr). I then discuss the results and provide a conclusion for future steps. Only the same parts as mentioned in the tutorial section of the assignment instruction were modified.

2.0 Custom Predictor Architecture

2.1 Hierarchical Predictor

Given the limitation of 256K for the total memory size of the predictor, I concluded that there is enough memory available to construct a hierarchical predictor. With this assumption in mind, I first divided the memory into 128K for a cheap 2BC predictor and 128K for a more accurate tournament predictor. The tournament includes a tag field for each row that specifies whether the specific branch is included in the tournament predictor. If a branch is not in the tournament predictor then the prediction is carried out using the larger simple 2-bit saturated counter predictor.

2.2 Tournament Predictor

In the case that the simple predictor makes a wrong prediction the branch is added to the tournament predictor for future predictions. The tournament predictor has two components, a pshare predictor and a gshare predictor (both of which have a gselect history). The tournament predictor selects the prediction from these two predictors based on the state of a pseudo-predictor counter. It then updates the counters and the history for both predictors and updates its state according to the following algorithm:

- Increment pseudo-predictor state if pshare was correct but gshare was incorrect
- Decrement pseudo-predictor state if gshare was correct but pshare was incorrect
- Do not change the pseudo-predictor state otherwise

The predictor architecture is shown in figure 1

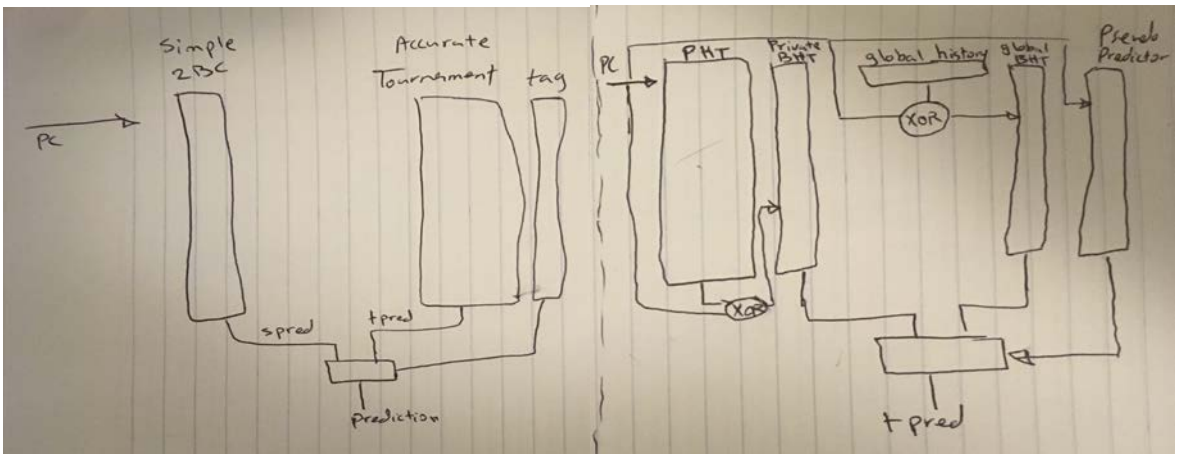

Figure 1 – a) The hierarchical predictor b) The inner tournament predictor

To decide on the distribution of memory size between different components of the tournament predictor I first assumed that 8K of rows would be appropriate, as it is not too large or too small relative to the simple predictor that has 128/2 = 64K rows. Therefore, the following formula was used:

Tag Size + PHT Size + Private BHT Size + Global BHT Size + Pseudo BHT Size = 16 bits (i)

By tweaking the variables in (i) along with the Global History Size in simulation, I arrived at the following results for a satisfactory performance.

3.0 Simulation Results

The simulation results is shown in figure 2.

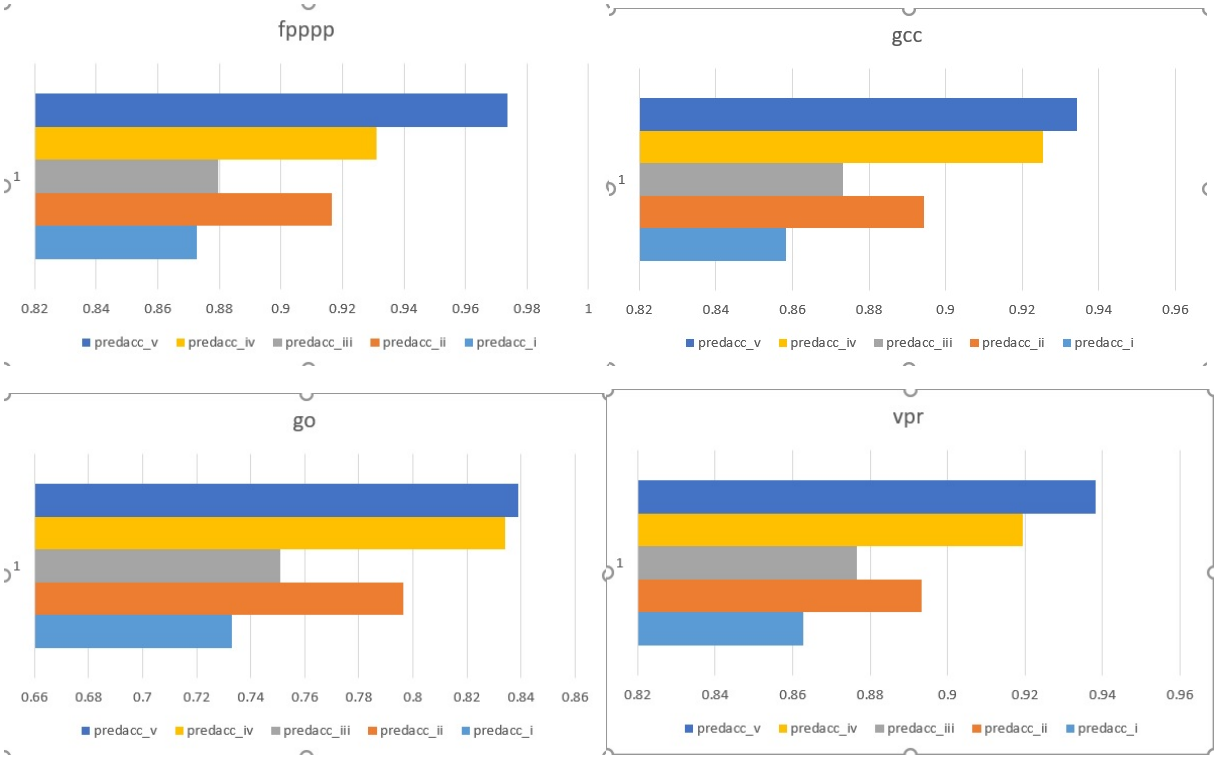

Figure 2 – Simulation result for 5 different predictor architectures against four benchmarks

4.0 Conclusion

As stated in section 2. Some of the design decisions for this predictor were based on assumptions. Therefore as the next step, I would test these assumptions in simulation to see if they hold. As an example, I would use 4K for the number of rows for the tournament predictor for a more accurate second level predictor. I would also try decoupling the number of rows in the gshare predictor from the number of rows in the pshare predictor as the gshare has a cheaper memory cost per row. This can be done by extending the hierarchical predictor to three levels instead of two levels with the gshare sitting between simple and pshare predictors. However, this has the drawback of having to use an additional table for the gshare tags. The parameters will also be optimized in a systematic way as opposed manually by tweaking.