**Jordan Carr, Kyle Hartenstein, Aric Hudson**

*CS1541  
Project 1*

*2/19/18*

|  |  |  |  |
| --- | --- | --- | --- |
| #Cycles for 64 Entry Table | | | |
| File name | No prediction (0) | 1 bit predictor (1) | 2 bit predictor(2) |
| sample1.tr | 1939824 | 1929230 | 1925984 |
| sample2.tr | 1927240 | 1858945 | 1880021 |
| sample3.tr | 2196389 | 1954864 | 1951767 |
| sample4.tr | 6063444 | 5743032 | 5757765 |
| sample\_large1.tr | 174440671 | 168002361 | 167358315 |
| sample\_large2.tr | 199433328 | 178459105 | 178211675 |

|  |  |  |  |
| --- | --- | --- | --- |
| #Cycles for 32 Entry Table | | | |
| File name | No prediction (0) | 1 bit predictor (1) | 2 bit predictor(2) |
| sample1.tr | x | x | x |
| sample2.tr | 1927240 | 1861703 | 1882407 |
| sample3.tr | 2196389 | 1956634 | 1953459 |
| sample4.tr | 6063444 | 5747560 | 5761843 |
| sample\_large1.tr | 174440671 | 170224777 | 170303826 |
| sample\_large2.tr | 199433328 | 178599432 | 178346342 |

|  |  |  |  |
| --- | --- | --- | --- |
| #Cycles for 128 Entry Table | | | |
| File name | No prediction (0) | 1 bit predictor (1) | 2 bit predictor(2) |
| sample1.tr | x | x | x |
| sample2.tr | 1927240 | 1853004 | 1875217 |
| sample3.tr | 2196389 | 1954847 | 1951776 |
| sample4.tr | 6063444 | 5737074 | 5752970 |
| sample\_large1.tr | 174440671 | 167987373 | 167337101 |
| sample\_large2.tr | 199433328 | 178457238 | 178207039 |

x= no change from the first table

**Notes on Testing**

Tests were performed on Thoth.cs.pitt.edu.

We interpreted instruction “squashing” as the insertion of “NOP” instructions into the pipeline.

**Results of Prediction Methods**

The tests performed yielded results that were mostly to be expected, with a few interesting variations therein.

When varying between prediction methods, we found that branch prediction, across the board, performed better than no prediction at all. This leads us to the conclusion that the predictions were accurate enough to reduce the number of cycles needed to execute the program, so any branches that were not taken (thus resulting in unnecessary steps from the prediction) were outweighed by the number of branches that *were* taken, and predicted correctly. This is true for the sample files provided.

Comparing the 1-bit and 2-bit prediction methods was not as straightforward. Some samples performed better on the 2-bit prediction method. Others performed better with a 1-bit prediction method. This might be the case due to the 2-bit predictor’s characteristic of requiring 2 mispredictions before changing what it should be predicted as. For these files, it may have made the program more inefficient to run overall since it may have been squashing 6 instructions for 2 wrong predictions as opposed to just 3 for the 1 bit predictor.

**Results of Table Size**

The tests showed with almost uniform consistency that increasing the size of the hash table improved cycle time with prediction. This leads us to conclude that, as a general rule, increasing the table size allows for more accurate prediction due to a fewer number of misses. There were two main exceptions to this rule: sample1.tr showed no change as a result of table size, indicating that there are likely very few branches in this file. The other exception is sample3.tr, which saw a very small increase in the number of cycles from a 64 bit table to a 128 bit table in the 2-bit predictor only. It is difficult to say for certain why this is occurring; there is only a nine-cycle difference, so for most intents and purposes these files run at essentially the same speed. It is possible that there is a random collision or table miss in the 128 bit table that does not occur in the other table sizes. Regardless, increasing table size does seem to improve performance as a general rule.