|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| sample1.tr | 2,027,618 | 1,543,674 | 1,534,907 |
| sample2.tr | 1,743,347 | 1,649,907 | 1,675,140 |
| sample3.tr | 2,099,329 | 1,754,481 | 1,731,672 |
| sample4.tr | 5,870,284 | 4,948,051 | 4,941,708 |
| sample\_large1.tr | 170,165,961 | 133,438,089 | 132,497,533 |
| sample\_large2.tr | 195,591,461 | 158,044,673 | 156,438,882 |

32

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| sample1.tr | 2,027,618 | 1,551,878 | 1,543,286 |
| sample2.tr | 1,743,347 | 1,662,096 | 1,685,482 |
| sample3.tr | 2,099,329 | 1,761,166 | 1,742,413 |
| sample4.tr | 5,870,284 | 4,975,129 | 4,971,170 |
| sample\_large1.tr | 170,165,961 | 137,270,687 | 136,899,185 |
| sample\_large2.tr | 195,591,461 | 158,580,889 | 157,302,835 |

128

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| sample1.tr |  |  |  |
| sample2.tr |  |  |  |
| sample3.tr |  |  |  |
| sample4.tr |  |  |  |
| sample\_large1.tr |  |  |  |
| sample\_large2.tr |  |  |  |

In all cases for the 64 unit table, the 1 bit prediction method significantly outperforms the default with no prediction (I can put numbers here, if need be). In all cases except for sample\_2, the two bit predictor shows a marginal improvement over the one bit predictor with the 64 prediction table. In the case of sample\_2, the two bit predictor performs marginally worse. This is somewhat counterintuitive, but not altogether implausible. For example, if a particular branch followed the pattern T-T-N-N-T-T-N-N, a 1 bit predictor would have a 50 percent success rate, whereas a 2 bit predictor would have a 0 percent success rate. Contrast this with T-N-T-N-T-N-T-N, where the 2 bit predictor would have a 50 percent success rate, and a 1 bit predictor would have a 0 percent success rate. It would follow that a program containing more sequences like the first than like the second would run more efficiently with a 1 bit predictor, but this is not the common case.

Changing the size of the prediction table made no difference for the default method, which, again, is what we’d expect. For the one bit and two bit prediction methods, making the table smaller increased the number of cycles needed to execute the trace. The likely explanation for this is that some elements are being written over on the smaller table, so we are losing information that would otherwise help us make more accurate predictions. These less accurate predictions result in more buffer flushing, which results in longer execution times. The other relationships described above between the 0, 1, and 2 bit prediction methods still hold true for the 32 element table, as well..