# Assignment 2

## Design Direction and Target Predictors

**Anshuman Barnwal · 200259**

Homework Assignment of

**CS422**

Monday 9th October, 2023

# BACKGROUND

In this design assignment, you will implement and evaluate a number of direction predictors for conditional branches and target predictors for indirect control transfer using a portion of the PIN tool that you developed in the first homework. This assignment is not meant to observe the performance advantage that can come from a good direction predictor or target predictor. Instead, you will only evaluate the predictor accuracy and not use any of the predictions.

a) Do not update any tables speculatively.

b) Initialize all tables to zero.

c) Do not shift out the least significant two bits of the PC before indexing because x86 instructions are not necessarily word-aligned.

d) All index functions are simple modulo hashes.

**Benchmarks:**

1. 400.perlbench
2. 401.bzip2
3. 403.gcc
4. 429.mcf
5. 450.soplex
6. 456.hmmer
7. 471.omnetpp
8. 483.xalancbmk

Results for all predictions need to be reported for the above benchmarks belonging to `SPEC 2006` Benchmark Suite.

# TASKS

**Part 1**

Implement the following conditional branch predictors and count the number of mispredictions.

1. Static forward not-taken and backward taken (FNBT) [5 points]
2. Bimodal: `512x2-bit` PHT
3. SAg: `1024x9-bit` BHT, `512x2-bit` PHT
4. GAg: `9-bit` global history, `512x3-bit` PHT
5. gshare: `9-bit` global history, `512x3-bit` PHT
6. Hybrid of SAg and GAg: Same configuration of SAg and GAg as above. The tournament meta-predictor is a table of `512x2-bit` saturating counters indexed by `9-bit` global history.
7. Hybrid of SAg, GAg, and gshare: Same configuration of SAg, GAg, and gshare as above, but with 2 types:
   (a) Majority Vote Meta-predictor
   (b) Tournament Meta-predictor

## Solution.

Below are the tables for each benchmark corresponding to all direction predictors v/s total, forward and backward branches mis-prediction rates.
Some observations:

- `400.perlbench` [1] *FNBT* performs nearly randomly, which implies the heuristic of loops dominant conditional flow is invalid in this benchmark. *SAg* shows huge improvement over *GAg* (and *gshare*) implying global correlations between branches is non-dominant, and local history of a particular branch decides the next condition with less than 4% miss-predictions. Ensemble predictors have their results similar to *SAg*, implying meta-predictors preferring *SAg* over others.

- `401.bzip2` [3] Again the heuristic assumed by *FNBT* fails. Each of *Bimodal*, *SAg* and *GAg* performance are similar, leading to all ensembles having similar predictions as well. The performance of *SAg* and *GAg* are similar (*SAg* outperforms by 3%) due to large number of data-dependence for branch operations, which couldn't be captured just by patterns in history of branches.

- `403.gcc` [5] *FNBT* performs better than random choice, so loop heuristic works more in favour for this benchmark. Other results are similar to `400.perlbench`.

- `429.mcf` [7] Performance of predictors is similar to `401.bzip2`, except that *SAg* leads to 3% more mis-predictions than *GAg*, implying higher inter-branch conditional dependence than specific branch's history dependence.
- `450.soplex` [9] *FNBT* is a good heuristic in this case, and all other predictors perform similar to each other with high accuracies. This is mainly due to possibly simple history-pattern-dependence conditional branches.
- `456.hmmer` [11] *FNBT* gives lower accuracy than random choice! From the data, 76% of forward branches are taken unlike the heuristic, due to possibly more `else` statements being executed. Other predictors performance are on-par with each other, betraying that there is higher global correlation between branches than usual.
- `471.omnetpp` [13] *FNBT* and *Bimodal* both show similar performance, while all other predictors show near 0 mis-predictions. This implies high-correlations between branches as well as history-pattern dependence.
- `483.xalanc` [15] *FNBT* and *GAg* (and *gshare*) have similar results, while the rest of direction predictors give similar results. This implies high inter-branch correlations, with working FNBT heuristic.

Table 1: 400.perlbench #branches

| Total | Forward | Backward |
|---|---|---|
| 129991027 | 103900130 | 26090897 |

Table 2: 400.perlbench Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 41.2919 | 37.562 | 56.1451 |
| Bimodal | 9.53699 | 9.95926 | 7.85545 |
| SAg | 3.65683 | 3.75314 | 3.27332 |
| GAg | 12.1096 | 12.8332 | 9.22766 |
| gshare | 10.1866 | 10.3049 | 9.7154 |
| T_SAgGAg | 3.13967 | 3.23221 | 2.77112 |
| M_SAgGAgGshare | 4.96942 | 5.13221 | 4.32115 |
| T_SAgGAgGshare | 2.79987 | 2.92081 | 2.31825 |

Table 3: 401.bzip2 #branches

| Total | Forward | Backward |
|---|---|---|
| 129923149 | 63177353 | 66745796 |

Table 4: 401.bzip2 Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 46.892 | 30.9892 | 61.9445 |
| Bimodal | 9.99839 | 10.7903 | 9.24878 |
| SAg | 10.1409 | 11.4242 | 8.92617 |
| GAg | 13.2202 | 15.722 | 10.8522 |
| gshare | 11.3094 | 12.1152 | 10.5467 |
| T_SAgGAg | 9.85796 | 11.1365 | 8.64775 |
| M_SAgGAgGshare | 9.39338 | 10.3249 | 8.51166 |
| T_SAgGAgGshare | 9.25029 | 10.2208 | 8.33171 |

Table 5: 403.gcc #branches

| Total | Forward | Backward |
|---|---|---|
| 145808191 | 114547462 | 31260729 |

Table 6: 403.gcc Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 36.5864 | 31.8035 | 54.1121 |
| Bimodal | 12.9408 | 14.7196 | 6.42318 |
| SAg | 5.09598 | 5.36649 | 4.10474 |
| GAg | 16.7859 | 17.5679 | 13.9204 |
| gshare | 9.73661 | 9.13438 | 11.9433 |
| T_SAgGAg | 4.28374 | 4.57834 | 3.20424 |
| M_SAgGAgGshare | 5.46378 | 5.46159 | 5.47183 |
| T_SAgGAgGshare | 2.98034 | 2.95794 | 3.06245 |

Table 7: 429.mcf #branches

| Total | Forward | Backward |
|---|---|---|
| 178242897 | 89125970 | 89116927 |

Table 8: 429.mcf Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 31.9495 | 35.7086 | 28.19 |
| Bimodal | 18.0195 | 16.2872 | 19.7519 |
| SAg | 13.0497 | 15.0971 | 11.002 |
| GAg | 10.0859 | 10.209 | 9.96275 |
| gshare | 10.2192 | 10.3475 | 10.0909 |
| T_SAgGAg | 9.45551 | 9.80084 | 9.11014 |
| M_SAgGAgGshare | 8.92204 | 9.07397 | 8.77008 |
| T_SAgGAgGshare | 8.61069 | 8.78472 | 8.43664 |

Table 9: 450.soplex #branches

| Total | Forward | Backward |
|---|---|---|
| 103254426 | 33692475 | 69561951 |

Table 10: 450.soplex Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 16.9548 | 19.9305 | 15.5134 |
| Bimodal | 4.82514 | 0.912735 | 6.72012 |
| SAg | 4.01566 | 0.658447 | 5.64174 |
| GAg | 3.84076 | 0.920947 | 5.25497 |
| gshare | 3.97118 | 1.3007 | 5.26464 |
| T_SAgGAg | 3.61228 | 0.673969 | 5.03545 |
| M_SAgGAgGshare | 3.69067 | 0.782534 | 5.09923 |
| T_SAgGAgGshare | 3.58002 | 0.660489 | 4.9941 |

Table 11: 456.hmmer #branches

| Total | Forward | Backward |
|---|---|---|
| 144361279 | 120193490 | 24167789 |

Table 12: 456.hmmer Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 63.914 | 76.6308 | 0.669726 |
| Bimodal | 8.5551 | 10.2042 | 0.353682 |
| SAg | 9.12531 | 10.8636 | 0.48023 |
| GAg | 11.6455 | 13.6352 | 1.7501 |
| gshare | 10.2738 | 11.7955 | 2.70597 |
| T_SAgGAg | 8.89633 | 10.5382 | 0.730973 |
| M_SAgGAgGshare | 8.45636 | 10.0604 | 0.478976 |
| T_SAgGAgGshare | 8.38702 | 9.97782 | 0.475492 |

Table 13: 471.omnetpp #branches

| Total | Forward | Backward |
|---|---|---|
| 144361279 | 120193490 | 24167789 |

Table 14: 471.omnetpp Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 24.9853 | 25.0424 | 15.065 |
| Bimodal | 24.7716 | 24.8756 | 6.70223 |
| SAg | 0.102834 | 0.0592631 | 7.67444 |
| GAg | 0.184697 | 0.127685 | 10.0921 |
| gshare | 0.184254 | 0.121447 | 11.0987 |
| T_SAgGAg | 0.110858 | 0.0610949 | 8.7585 |
| M_SAgGAgGshare | 0.136062 | 0.0863449 | 8.7757 |
| T_SAgGAgGshare | 0.114107 | 0.063917 | 8.83593 |

Table 15: 483.xalanc #branches

| Total | Forward | Backward |
|---|---|---|
| 176657397 | 130391813 | 46265584 |

Table 16: 483.xalanc Miss-rates

| Predictor | Total | Forward | Backward |
|---|---|---|---|
| FNBT | 7.98427 | 8.9646 | 5.2214 |
| Bimodal | 3.71057 | 4.25039 | 2.18917 |
| SAg | 1.93878 | 2.19565 | 1.21484 |
| GAg | 5.38099 | 5.94853 | 3.7815 |
| gshare | 4.16752 | 4.55032 | 3.08867 |
| T_SAgGAg | 1.47013 | 1.61294 | 1.06766 |
| M_SAgGAgGshare | 2.31175 | 2.47906 | 1.84021 |
| T_SAgGAgGshare | 1.33369 | 1.47087 | 0.947078 |

**Part 2**

The branch target buffer (BTB) can offer nearly 100% accurate target prediction for direct calls and unconditional jumps because the targets of these control transfer instructions do not change. This may not be true for indirect control transfer instructions. Particularly, the target of an indirect control transfer may have high correlation with the control flow path from which the branch is made. We will evaluate the following two target predictors for indirect control transfer instructions in terms of the number of mis-predictions.

a) **Branch target buffer** (BTB) indexed with PC

b) **Branch target buffer** (BTB) indexed with a **hash of PC and ghr**

## Solution.

Table 17: BTB and BTBP performance in %

| Application | #Jumps | BTB | | BTBP | |
|---|---|---|---|---|---|
| | | Accuracy | Miss | Accuracy | Miss |
| 400.perlbench | 28113355 | 64.4209 | 0.729852 | 87.8113 | 3.56937 |
| 401.bzip2 | 791933 | 51.7064 | 0.00745013 | 52.4765 | 0.0270225 |
| 403.gcc | 7047066 | 63.9846 | 0.732844 | 86.7383 | 2.19991 |
| 429.mcf | 12556568 | 99.3865 | 7.16756e-05 | 99.5898 | 0.00111495 |
| 450.soplex | 6333664 | 99.9745 | 0.00271565 | 99.9654 | 0.0104995 |
| 456.hmmer | 201471 | 93.3732 | 0.300291 | 96.9757 | 0.807064 |
| 471.omnetpp | 30294875 | 69.8068 | 1.14926 | 87.1636 | 2.12404 |
| 483.xalancbmk | 33225542 | 66.8726 | 8.80345 | 67.4221 | 20.515 |

A clear result from the above is *BTBP*, aka BTB indexed with hash of PC and ghr has better accuracy, while having more miss-rates per benchmark as well.

This can be accounted by noting that the same PC arrived from a different set of conditional flow will be mapped to a new set, leading to a miss for the same PC, while leading to saving the target of PC with different possible paths instead of over-writing them, which increases the accuracy.

Also low accuracy of ~ 65% for both `400.perlbench` and `483.xalancbmk` arise from different situations, as indicated by entirely different miss-rates. `400.perlbench` low miss-rates implies `pc` gets hit in cache, but target mis-matches, implies global branches flow isn't indicative about the target as well as in the case of `483.xalancbmk`. This is also shown in Q1 by lower performance of GAg in perlbench v/s xalancbmk.

## Sidenotes

1. `TARGE_IA32` is a preprocessor directive that is `true` when building for `ia32`
2. LRU states were captured using positioning of a linked list, that represented a set in BTB cache. The head of the set represents the most recently-used block, while the tail represents least recently-used.

| #Jumps | Number of Jumps |
|---|---|
| FNBT | Forward Not-taken, Backward Taken |
| T_SAgGAg | Hybrid of SAg and GAg using Tournament Meta-predictor |
| M_SAgGAg | Hybrid of SAg and GAg using Majority Vote Meta-predictor |
| T_SAgGAgGshare | Hybrid of SAg, GAg, gshare using Tournament Meta-predictor |

Table 18: Abbreviations

Please find more information regarding the submission in `README.md` in the submission directory.