**EE/CE/CS 6304: Computer Architecture**

**Project # 2**



Project Description

Different choices predictors (i.e. taken, not taken, bimodal, 2 level, combinational etc.) for the branch affect the performance of a microprocessor. In this project, it was asked to use sim outorder, which models all the execution aspects of the Alpha microprocessor 21264 for three individual benchmarks. Thus, the simulator will provide the calculated CPI (sim\_CPI variable) to use for comparisons. The branch prediction design parameters it which have been modified are:

* **Branch Predictor Type**: Bimodal, 2level and combinational.
* **Branch Predictor Configuration**: The default configuration uses 256 as Table size.
* **Two Level Prediction Configuration**: N is # of entries in the first level which is usually 1, M is # of entries in second level which is usually 1024, W is width of the shift register(s) which is usually 8, X is XOR history and address for 2nd level index (yes-1/no-0).
* **Combining Predictor Configuration**: It combines Bimodal and 2 Level Predictor. Usually uses 1024 as table size.
* **Return Address Stack**: Value for RAS is usually selected from 4, 8 and 16 (0 for no return stack).
* **Branch Table Buffer**: number sets and associativity, usually it is 64 for number of sets with 2-way associativity.

Since simulation of the out-of-order model is much slower, we have used the –fastfwd and -max:inst switches to limit the number of instructions executed. Hence we are skipping at least 1M instructions and executing at least 10M instructions. Branch predictors with minimum miss rate are generally considered as better branch predictors, they also come with more cost. Thus, design is chosen considering tradeoffs.

**Part 1: Compare performance of several branch predictor types and different**

**RAS configurations**

In this part the performance of three (bimodal, combinational and two level) branch predictor types of three individual benchmarks is compared with baseline configuration and analyzed.

Baseline configuration will be the Alpha 21264 EV6 configuration:

- **Branch Predictor Type:** Bimodal.

- **Branch Predictor Configuration Table size**: 256.

- **Return Address Stack**: 8.

- **Branch Table Buffer**: number of sets = 64, associativity = 2.

Given the baseline configuration decision is made between bimodal, 2 level and combinational predictors and return address stack values which are 4, 8 and 16.

## **Assumptions:**

1. To simplify the analysis, for comparing branch predictor performance only CPI, address hit rates (bpred\_addr\_rate) and direction hit rates (bpred\_dir\_rate) are compared with baseline configuration. Other hit rates like JR rate and RAS rate are not considered as these values are covered by address hit rate.
2. To compare the performance of Return Address Stack, the return address stack hit rates (ras\_rate) and CPI(sim\_CPI) are used.

**Performance Comparison of branch predictor types:**

**Benchmark: GCC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **CPI** | **Dir\_hit\_rate** | **Address\_hit\_rate** |
| Baseline-bimod | 0.9741 | 0.8507 | 0.6409 |
| 2 level | 0.9998 | 0.8021 | 0.5935 |
| combining | 0.954 | 0.8777 | 0.6546 |

bimodal predictor has marginally better performance than 2-level predictor considering CPI value. In address hit rate and CPI, combining predictor has marginally better performance than bimodal and 2 level predictor has least performance. In direction hit rate also combining predictor have marginally better performance than bimodal predictors and 2 level predictor has least performance.

The combining predictor has best performance out of all three predictors. So having static predictor is marginally better in GCC benchmark. Having combinational predictors which decides which one to use from bimodal and 2-level predictor gives better result.

**Benchmark: Anagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **CPI** | **Dir\_hit\_rate** | **Address\_hit\_rate** |
| Baseline-Bimod | 0.4212 | 0.9644 | 0.9744 |
| 2 level | 0.4171 | 0.9695 | 0.9799 |
| Combining | 0.4102 | 0.9844 | 0.9844 |

2-level predictors have marginally better performance than bimodal predictor in terms of CPI. The combining predictor has best performance out of all three predictors. So having static predictor is marginally better in GCC benchmark. Having combining predictor which decides which one to use from bimodal and 2-level gives the better result.

      In address hit rate, combining predictors have better performance than 2 level predictor and bimodal has least performance. In direction hit rate, combining has better performance than bimodal and 2 level predictor.

**Benchmark: Go**

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **CPI** | **Dir\_hit\_rate** | **Address\_hit\_rate** |
| Baseline-bimod | 0.7212 | 0.8174 | 0.7527 |
| 2 Level | 0.7525 | 0.7656 | 0.6981 |
| Combining | 0.7169 | 0.8208 | 0.7539 |

The bimodal predictor has better performance than 2 level predictor. In address hit rate, combining predictor has better performance than bimodal and 2 level predictor. In direction hit rate also combining predictor has better performance than bimodal predictors and 2 level predictor. Amongst all of three, 2-level predictor has least performance.

The combining predictor has best performance in all three predictors. So having static predictor is marginally better in GCC benchmark. Having combinational predictors which decides which one to use from bimodal and 2-level gives the better result.

**Performance Comparison for Different RAS Configurations:**

***GCC Benchmark:***

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **Hit\_Rate** |
| Baseline-RAS 8 | 0.9741 | 0.9687 |
| RAS 4 | 0.9765 | 0.9222 |
| RAS 16 | 0.9739 | 0.972 |

The CPI for RAS 16 and 8 are less than that of RAS 4 configuration. The performance in terms of RAS Hit rate is better for RAS 16 than that of RAS 4 and RAS 8 configuration.

***Anagram Benchmark:***

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **RAS\_Hit\_Rate** |
| Baseline - RAS 8 | 0.4212 | 0.9949 |
| RAS - 4 | 0.4378 | 0.955 |
| RAS 16 | 0.4212 | 0.9995 |

The CPI for RAS 16 is better than that of RAS 8 and RAS 4 configuration. The performance in terms of RAS Hit rate is also better for RAS 16 than RAS 8 and RAS 4 configuration.

***Go Benchmark:***

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **RAS\_Hit\_Rate** |
| Baseline - RAS 8 | 0.7212 | 0.9759 |
| RAS - 4 | 0.7215 | 0.9683 |
| RAS 16 | 0.7212 | 0.9759 |

The CPI for RAS 16 and RAS8 is less than that of RAS 4 configuration. The performance is better for RAS 16 and 8 than that of RAS 4 configuration.

**OPTIMUM BRANCH PREDICTOR FOR ALL THREE BENCH MARKS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| Bimod - RAS 8 | 0.9741 | 0.4212 | 0.7212 |
| 2-level | 0.998 | 0.4171 | 0.7525 |
| Combining | 0.954 | 0.4102 | 0.7169 |
| Bimod - RAS 4 | 0.9765 | 0.4378 | 0.7215 |
| Bimod - RAS 16 | 0.9739 | 0.4231 | 0.7212 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| Bimod-RAS 8 | 0.9687 | 0.9995 | 0.8975 |
| 2 - level | 0.8021 | 0.9842 | 0.7656 |
| Combining | 0.8777 | 0.9844 | 0.8208 |
| Bimod-RAS 4 | 0.9222 | 0.9878 | 0.9683 |
| Bimod-RAS 16 | 0.972 | 0.9989 | 0.9759 |

For simplification only direction hit rate is considered here as it contains address hit rate. So the graph above shows comparison of address hit rates of different configuration for all three benchmarks.**Conclusion:**

***GCC Benchmark:***

1) Combining Predictor has best performance of all three branch predictor types in terms of CPI and branch predictor hit rate.

2) Both RAS 16 and RAS 8 have better performance than RAS 4 configuration.

***Anagram Benchmark:***

1) Combining Predictor has best performance of all three branch predictor types in terms of CPI and branch predictor hit rate.

2) RAS 16 has better performance than RAS8 and RAS 4 configuration.

***Go Benchmark:***

1) Combining Predictor has best performance of all three branch predictor types in terms of CPI and branch predictor hit rate.

2) Both RAS 16 and RAS 8 have better performance than RAS 4 configuration.

***Overall Benchmark:***

1) Combinational Predictor has best performance of all three branch predictor types in terms of CPI and branch predictor hit rate.

2) RAS 16 has better performance than RAS8 and RAS 4 configuration.

**Points inferred from above Analysis:**

* ***Branch Predictor Types:***

1) Combining predictor has the best performance compared to bimodal and 2 level predictor in all the benchmarks. Some branches do not follow a particular pattern, so learning history would not help. The bimodal is better in this case. Other branches learning history, makes the prediction better. The combination of these models helps us to predict overall direction better.

2) 2 Level predictor has better performance than bimodal predictor in 1/3 benchmarks as learning the history of the branches makes the prediction better.

3) Bimodal predictor has better performance than bimodal predictor in 2/3 benchmarks as learning the history of the branches makes the prediction better.

* ***RAS Configuration:***

1. Increasing RAS size, we get better performance in terms of CPI and hit\_rates.

2) There is limit on increasing the RAS size, increasing beyond some point will keep the hit rate constant.

## **Part 2: Modify Simple scalar to accommodate address misses**

*Considering parameters given below,*

* dir\_hits = number of correct predictions
* misses = number of incorrect predictions on directions only
* addr\_hits = number of right address prediction = shows no of cases when the predictor predicts the result of the branch instruction correctly

*Address miss rate formula used:*

**addr misses = misses + dir\_hits – addr\_hits**

The source code is modified in the function,

void bpred\_reg\_stats(struct bpred\_t \*pred,struct stat\_sdb\_t \*sdb)

***Modified Source Code:***

sprintf(buf, **"%s.addr\_misses",** name);

sprintf(buf1**, "%s.misses + %s.dir\_hits - %s.addr\_hits"**, name, name, name);

stat\_reg\_formula(sdb, buf, **"total number of direction-predicted misses + address-predicted misses"**,buf1, **"%12.0f"**);

***For compilation of Source code:***

1) *make clean* - used to clean the available object and executable file

2) *make* - used to build the modified code

**Part 3: Compare performance of the branch target buffer (BTB) with**

**different configurations**

**BTB performance analysis for different configuration:**  
  
       The number of address hits (addr\_hits) and number of address misses (addr\_misses) are considered for evaluating BTB performance for varying number of sets and associativity.

***Performance of Branch Target Buffer with Varying Set:***

*GCC Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Bimod : 64 sets | 1005521 | 563340 |
| Bimod : 32 sets | 937745 | 631116 |
| Bimod : 128 sets | 1100970 | 467891 |

From the above graph and data it can be concluded that when number of sets increases, the address hits increases and address misses decreases. BTB with 128 set has the best performance.

*Anagram Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Bimod : 64 sets | 1317577 | 609150 |
| Bimod : 32 sets | 1256380 | 789105 |
| Bimod : 128 sets | 1418870 | 598764 |

From the above graph and data it can be concluded that when BTB sets increases, the address hits increases and address misses decreases. BTB with set size 128 has the best performance.

*GO Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Bimod : 64 sets | 1051854 | 345634 |
| Bimod : 32 sets | 1010312 | 387176 |
| Bimod : 128 sets | 1076639 | 320849 |

From the above graph and data it can be concluded that when BTB sets increases, the address hits increases and address misses decreases. BTB with number of sets 128 has the best performance.

***Overall Performance with different number of sets:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| Bimod : 64 sets | 1005521 | 1317577 | 1051854 |
| Bimod : 32 sets | 937745 | 1256380 | 1010312 |
| Bimod : 128 sets | 1100970 | 1418870 | 1076639 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| Bimod : 64 sets | 563340 | 609150 | 345634 |
| Bimod : 32 sets | 631116 | 789105 | 387176 |
| Bimod : 128 sets | 467891 | 598764 | 320849 |

**Conclusion:**

1. Increasing the number of sets gives better performance as it increases addr\_hits.
2. Increasing the number of sets gives better performance as it decreases addr\_misses.

***Performance of Branch Target Buffer with Varying Associativity:***

*GCC Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Associativity 2 | 1005521 | 563340 |
| Associativity 4 | 1018386 | 550475 |
| Associativity 1 | 995879 | 572982 |

From the above graph and data it is clear that when BTB associativity increases, the address hits increases and address misses decreases. BTB with 4-way associativity set has the best performance.

*Anagram Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Associativity 2 | 1317380 | 460915 |
| Associativity 4 | 1327556 | 432915 |
| Associativity 1 | 1185369 | 478920 |

From the above graph and data it is clear that when BTB associativity increases, the address hits increases and address misses decreases. BTB with 4-way associativity set has the best performance.

*GO Benchmark:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Address\_Hits** | **Address\_Miss** |
| Associativity 2 | 1051854 | 345634 |
| Associativity 4 | 1054300 | 343188 |
| Associativity 1 | 1031232 | 366256 |

From the above graph and data it is clear that when BTB associativity increases, the address hits increases and address misses decreases. BTB with 4-way associativity set has the best performance.

***Overall Performance with different set associativity:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| Associativity 2 | 1005521 | 1317380 | 1051854 |
| Associativity 1 | 1018386 | 1327556 | 1054300 |
| Associativity 4 | 995879 | 1185369 | 1031232 |

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration | GCC | Anagram | Go |
| Associativity 2 | 563340 | 460915 | 345634 |
| Associativity 1 | 550475 | 432915 | 343188 |
| Associativity 4 | 572982 | 478920 | 366256 |

**Conclusion:**

From the above graph, it is clear that as the associativity increases the performance gets better i.e the address hits increases. 4 -way associativity shows increment in addr\_hits and decrement in addr\_miss amongst all.

**BTB performance to the Overall performance:**

1. For comparing the effect of BTB performance to the overall performance, the CPI and address hit rate are considered.
2. The direction hit rate is not considered here, since the direction hit rate would be constant, the configuration for branch history table is fixed.

**Varying Number of Sets in BTB:**

*GCC BENCHMARK:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **addr\_rate** |
| Bimod : 64 sets | 0.9741 | 0.6409 |
| Bimod : 32 sets | 0.9899 | 0.5977 |
| Bimod : 128 sets | 0.9495 | 0.7018 |

From the above graph, it is clear that as the number of sets increases the performance gets better i.e the address hit rate increases and CPI decreases. BTB with 128 sets has better performance.

*ANAGRAM BENCHMARK:*

|  |  |  |
| --- | --- | --- |
| **configuration** | **CPI** | **addr\_rate** |
| Bimod : 64 sets | 0.4165 | 0.9744 |
| Bimod : 32 sets | 0.4171 | 0.9799 |
| Bimod : 128 sets | 0.4103 | 0.9844 |

From the above graph , it is clear that as the number of sets increases the performance gets better i.e the address hit rate increases and CPI decreases. BTB with 128 sets has better performance.

*GO BENCHMARK:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **addr\_rate** |
| Bimod : 64 sets | 0.7212 | 0.7527 |
| Bimod : 32 sets | 0.7268 | 0.7229 |
| Bimod : 128 sets | 0.7164 | 0.7704 |

From the above graph, it is clear that as the number of sets increases the performance gets better i.e the address hit rate increases and CPI decreases. BTB with 128 sets has better performance.

***OVERALL BENCHMARK:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| 64 sets, 2 way associativity | 0.9741 | 0.4165 | 0.7212 |
| 32 sets, 2 way associativity | 0.9899 | 0.4171 | 0.7268 |
| 128 sets, 2 way associativity | 0.9495 | 0.4103 | 0.7164 |
| 32 sets, 4 way associativity | 0.9737 | 0.4165 | 0.7209 |
| 128 sets, 1 way associativity | 0.9748 | 0.4312 | 0.723 |

From the above graph, it is clear that as the number of sets increases the performance gets better i.e the address hit rate increases and CPI decreases. BTB with 128 sets has better performance.

***Varying Associativity in BTB:***

*GCC BENCHMARK:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **addr\_rate** |
| Associativity 2 | 0.9736 | 0.6409 |
| Associativity 1 | 0.9737 | 0.6348 |
| Associativity 4 | 0.9734 | 0.6491 |

From the above graph, it is clear that as the associativity increases the performance gets better i.e the address hit rate increases and CPI decreases. 4 -way associativity buffer has better performance.

*ANAGRAM BENCHMARK:*

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **addr\_rate** |
| Associativity 2 | 0.4312 | 0.9659 |
| Associativity 1 | 0.4356 | 0.955 |
| Associativity 4 | 0.4305 | 0.9844 |

From the above graph, it is clear that as the associativity increases the performance gets better i.e the address hit rate increases and CPI decreases. 4 -way associativity buffer has better performance.

GO BENCHMARK:

|  |  |  |
| --- | --- | --- |
| **Configuration** | **CPI** | **addr\_rate** |
| Associativity 2 | 0.7212 | 0.7527 |
| Associativity 1 | 0.723 | 0.7544 |
| Associativity 4 | 0.7209 | 0.7679 |

From the above graph, it is clear that as the associativity increases the performance gets better i.e the address hit rate increases and CPI decreases. 4 -way associativity buffer has better performance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **GCC** | **Anagram** | **Go** |
| 64 sets, 2 way associativity | 0.6409 | 0.9659 | 0.7527 |
| 32 sets, 2 way associativity | 0.5977 | 0.9799 | 0.7229 |
| 128 sets, 2 way associativity | 0.7018 | 0.9844 | 0.7704 |
| 32 sets, 4 way associativity | 0.6491 | 0.955 | 0.7544 |
| 128 sets, 1 way associativity | 0.6348 | 0.9844 | 0.7379 |

From the above graph, it is clear that as the associativity increases the performance gets better i.e the address hit rate increases and CPI decreases. 4 -way associativity buffer has better performance.

**CONCLUSION:**

***Gcc Benchmark:***

1) BTB with more sets (128) has better performance than 64 and 32 sets.

2) BTB with higher associativity (4-way) has better performance than 2-way and 1-way

associativity

***Anagram Benchmark:***

1) BTB with more sets (128) has better performance than 64 and 32 sets.

2) BTB with higher associativity (4-way) has better performance than 2-way and 1-way

associativity

***Go Benchmark:***

1) BTB with more sets (128) has better performance than 64 and 32 sets.

2) BTB with higher associativity (4-way) has better performance than 2-way and 1-way

associativity