Marks Distribution: 100/100

|  |  |
| --- | --- |
| Image result for utd  Computer Architecture  Project 1 Report | Abstract  Implementation of Local Branch Predictor, Bi-mode branch predictor and Tournament branch predictor in gem5 simulator. Every predictor is experimented on 5 distinct SPEC benchmarks and results are determined.  Anmol Gautam , Soumyadeep Choudhury  AXG190014 SXC180056 |

Table of Contents

Introduction Page 2-3

* Gem5
* Branch Predictors
* Local Branch Predictor
* Bi-Mode Branch Predictor
* Tournament Branch Predictor

Installing gem5 Page 4

Challenge and Issues Page 5-6

Modifications and Changes Page 7-9

Config.ini Page 10

Generated Stats File Page 10

Results Page 11-14

Conclusion Page 14

Introduction

**Gem5**

Gem5 simulator is a platform for simulating computer system architecture. It incorporates system level architecture and processor microarchitecture.

**Key Features**:

* Gem5 gives four interpretation-based CPUs which shares a common high-level ISA architecture.
* A completely coordinated GPU mode.
* Integrated NoMali GPU model removes the need for software rendering.
* Implemented event-driven memory system captures the impact of current and emerging memories.
* A model that plays back elastic traces.
* Homogenous and Heterogeneous multicore model.
* Full system capacity for Alpha, ARM, SPARC and x86
* Various frameworks can be launched inside a solitary reenactment process.
* Gem5's items are masterminded in OS-obvious power and clock areas, empowering a scope of tests in power-and vitality proficiency.
* Gem5 can be effectively ran as a thread inside system event kernel, which synchronizes events and timeline.

**Branch Predictor**:

Branch expectation is a way to deal with PC design that endeavors to relieve the expenses of branches. Branch predication accelerates the handling of branch instructions with CPUs utilizing pipelining. The strategy includes possibly executing certain guidelines if certain predicates are valid. The branch forecast is normally actualized in hardware utilizing a branch indicator.

At the point when a conditional operation, for example, an if… else or loop (like for or while loop) needs to be processed, the branch indicator "estimates" what condition is well on the way to be met. The branch predictor guesses these operations that might be needed in future and execute it beforehand. When program runs and the predicted output does not matches the actual output, CPU needs execute correct branch of operation which adds delay. But if the prediction is correct, extra clock cycle overhead for executing those instructions is reduced and a significant increase in speed is obtained.

**Local Branch Predictor**:

A local branch indicator has a different history cradle for each contingent hop guidance. It might utilize a two-level versatile indicator. The history cradle is independent for each conditional jump guidance, while the pattern history table (PHT) might be discrete too or it might be shared between every contingent jump.

**Bi-Mode Predictor**:

The bi-mode predictor, which divides the prediction tables into two halves and, by dynamically determining the current “mode” of the program, selects the appropriate half of the table for prediction. This approach is shown to preserve the merits of global history-based prediction while reducing destructive aliasing and, as a result, improving prediction accuracy. Moreover, it is simple enough that it does not impact a processor’s cycle time.

**Tournament Branch Predictor**:

A tournament branch predictor consolidates global and local branch predictors. It uses a selector to select the prediction outcome from one of them. It then updates both predictors as well as selector for each branch. Global predictor is indexed by history register and local predictor is indexed by LSB’s of branch PC.

Installing Gem5

We opted to use our local machine for this project. Main reason being that we thought, that on a local machine each individual benchmark would take less time that on server. We inferred that to be correct based on the run time of bzip2 and sjeng provided in the project description and the time we got on our local machine while testing our setup for the first time.

Our local machine specifications are as follow 🡪 Core i5 (7th generation), overclockable up to 3.22 Ghz, 20 GB of DRAM.

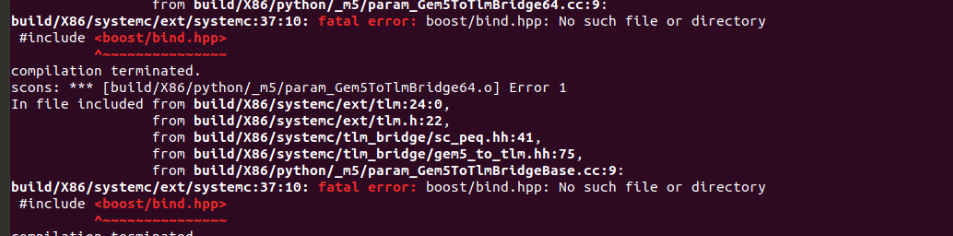
Challenges and Issues

**Challenges:**

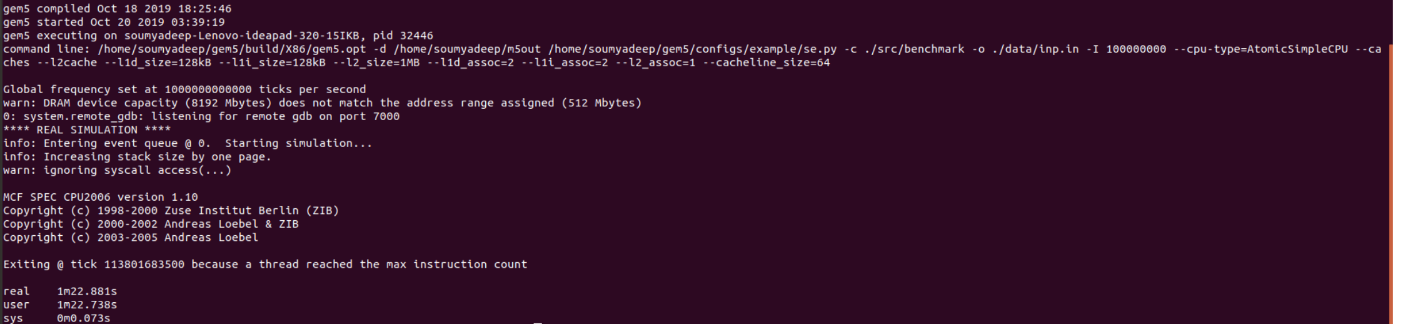
* To build on gem5, we had to install dependencies for which we referred to the official documentation of gem5 where required softwares and libraries are mentioned.
* Simply installing dependencies as mentioned in manual pages, on our local system, did not work
  + This happened because in our system we are using deep learning cuda network with python 3.6 libraries for Machine Learning.
  + We are using cuda10.1, Python 3.6, cudnn 2.1., cross compiler is upgraded to latest version and kernel version had been upgraded to > 4.3.3. Official manual page of gem5 suggests using kernel version between 3.1.3 to 4.1.1.
  + Ubuntu version is 18.04.5 LTS. Libraries suggested in gem5 manual page are not directly compatible for this Ubuntu version.
* To solve the above mentioned issues, we created two instances of the kernel and installed the following files :
  + Softwares : SCons1, zlib1, m41, protobuf1, pydot1
  + Dependencies : libfdt1, dnet1, iostream3, libelf1, PLY1, x11ksyms1, fputils1
* Cross compiler issue was solved by aliasing UEFI mode in boot system so that simulator can run on CPU kernel instead of GPU (default for cuda network).

**Issues:**

* Faced an issue of <boost/bind.hpp> while building gem5. This was because of kernel version issue. This was solved by reducing kernel version.



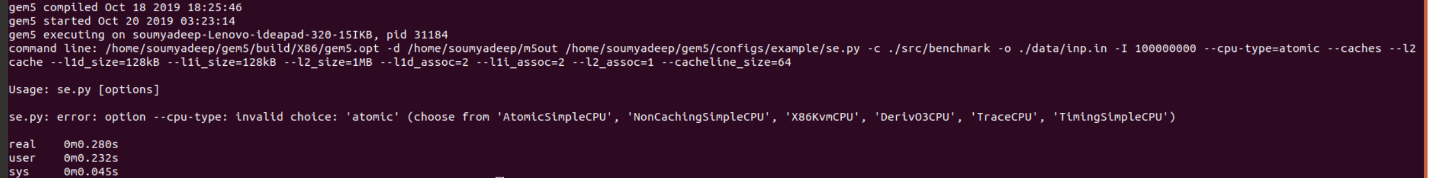
* Issue faced of parsing Input arguments in benchmark files. These took string values for which appropriate changes were made in shell file.
* Last thread was implicitly exiting every time. The reason for this was that the number of channels and time steps (lbm) were not synchronized.
* Maximum number of threads that can run parallelly in gem5 was prespecified at a less value. Because of that gem5 was not able to execute all instructions if the total number of instructions were given a high value. Solved by increasing the maximum thread count.



* Panic condition fd<0 occurred which led the failure of opening the .src/benchmark file because DRAM memory capacity usage exceeded/did not match the address range that was assigned.



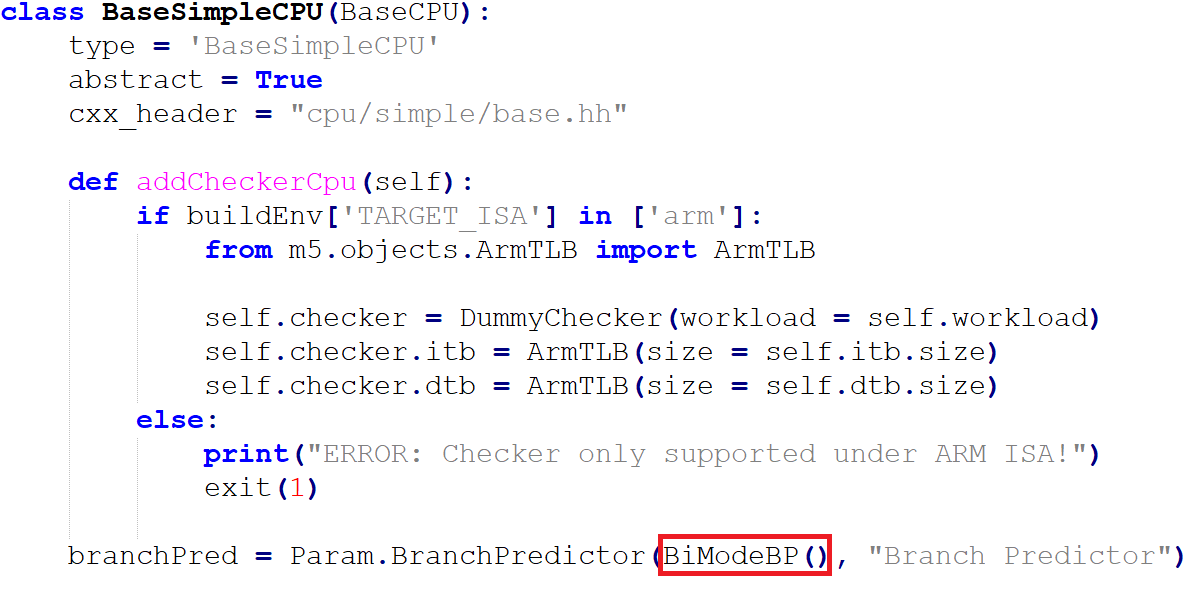
* CPU type in shell was given as Atomic but name convention of executable created by python libraries in gem5/build/86 is different.



Modifications and Changes

To make branch predictors work, following are the changes made along with their specific files.

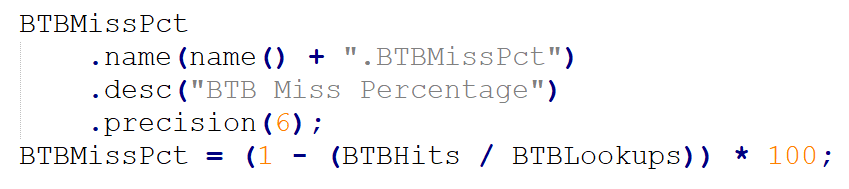
* We change the value specified as NULL in BaseSimpleCPU.py with the correct predictor function call.



OR



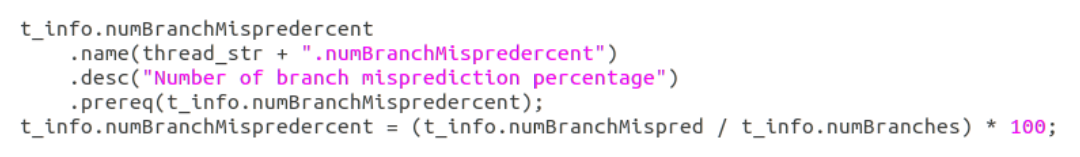
* In bpred\_unit.cc, Branch Target Buffer (BTB) miss percentage is added.



* In bpred\_unit.hh header file for bpred\_unit.cc variables used for BTB miss percentage and branch miss percentage is added in stat register.



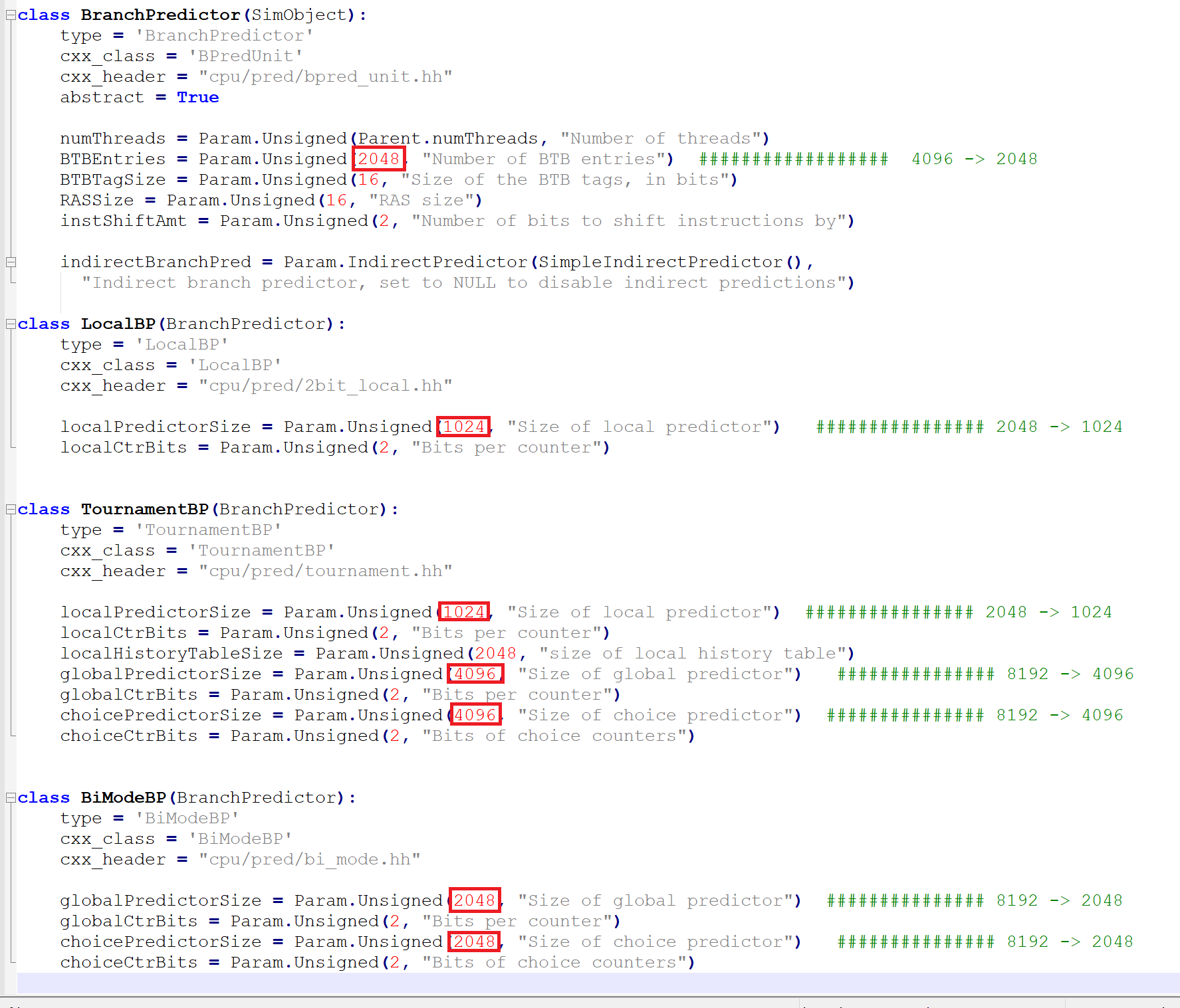
* In base.cc 🡪Branch miss prediction percent is added



* In exec\_context.hh 🡪 Branch miss prediction percent is declared in this

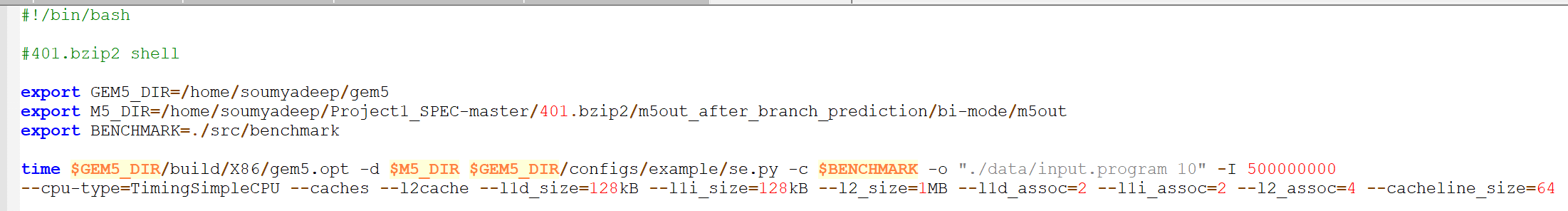


* In BranchPredictor.py changes to size and entry numerical bound parameters were changed as suggested in Part 4 of the project description. (highlighted in yellow in project description)

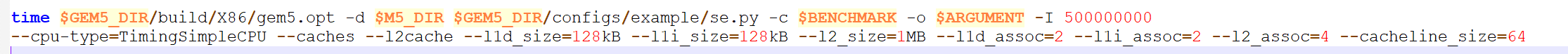


* Changes to shell file of each benchmark is done as parsing of argument parameters is different for each benchmark. Also the output file location is provided and changed before running each predictor to direct output to specific desired locations.

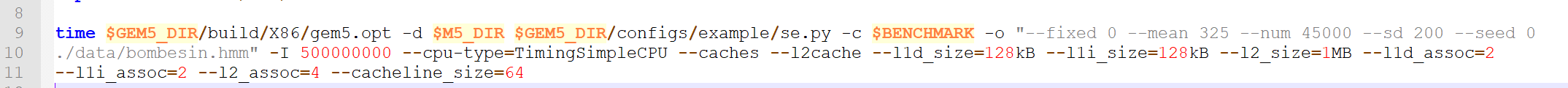
**401.bzip2:**



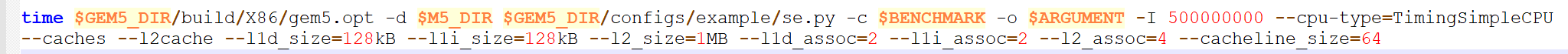
**429.mcf:**



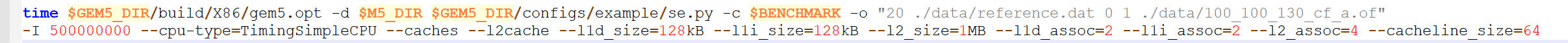
**456.hmmer:**



**458.sjeng:**

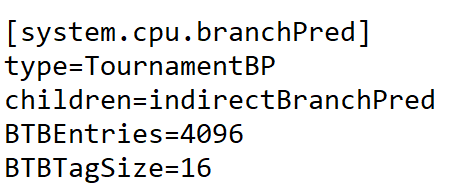


**470.lbm:**



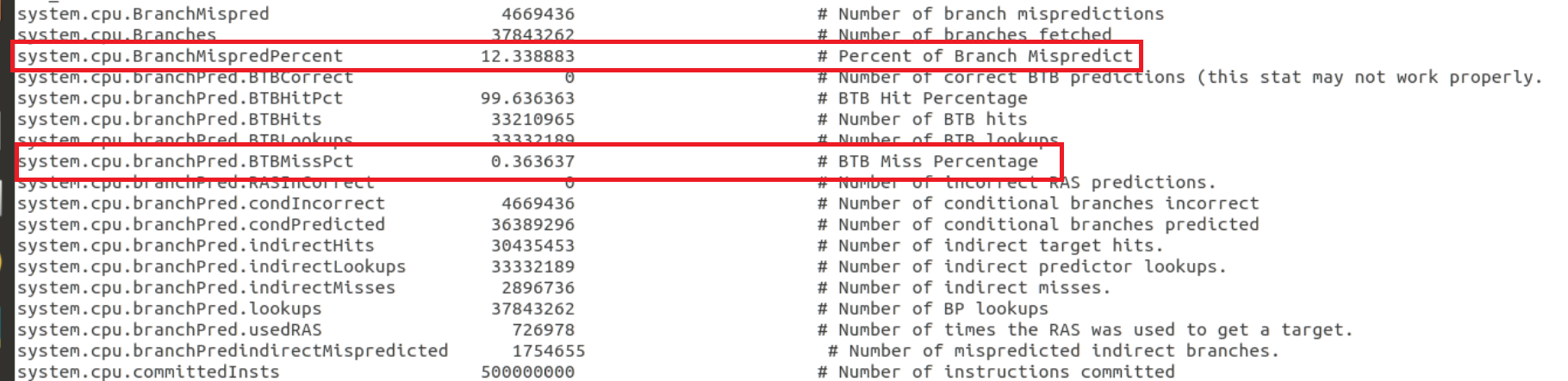
Config.ini

Whenever a branch predictor runs successfully, config.ini file reflects the name of the predictor used. Results of config.ini for Tournament Branch Predictor are given below. The same changes shows on all config.ini files of all benchmarks.



Generated Stats File

Stats are generated for BranchMispredPercent and BTBMissPct. An example of which is given below.

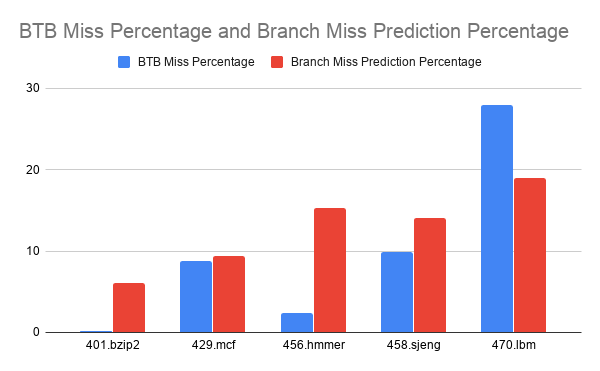


Results

All 3 predictors were tested using 5 benchmarks specified in project description. All the data for all three predictors are mentioned in a tabular column to ease the understanding along with the calculated CPI of the system as per the formula given in the project description.

**Local Branch Predictor:**

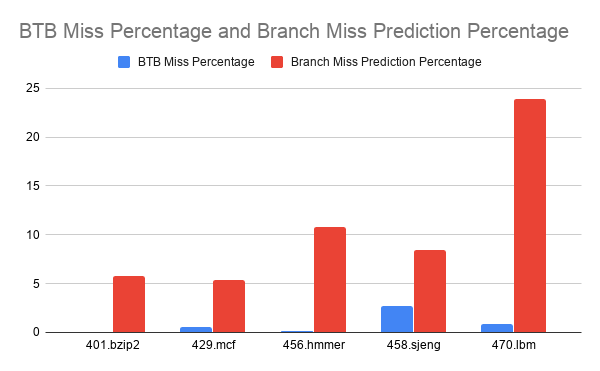
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETERS** | **401.bzip2** | **429.mcf** | **456.hmmer** | **458.sjeng** | **470.lbm** |
| BTB Miss % | 0.183930 | 8.734989 | 2.386576 | 9.875465 | 27.932256 |
| Branch Miss Prediction % | 6.01821 | 9.3814 | 15.286 | 14.0412 | 18.9972 |
| L1 Data Misses | 4712756 | 10327189 | 311 | 942325 | 189 |
| L1 Instruction Misses | 986 | 690 | 445 | 2378 | 332 |
| L2 Cache Misses | 2938275 | 7758241 | 632 | 8175286 | 540 |
| CPI (System) | 1.5644 | 2.4479 | 1.00125232 | 2.32793 | 1.0000968 |



Comparison figure for Local Branch Predictor

**Bi-Mode Branch Predictor:**

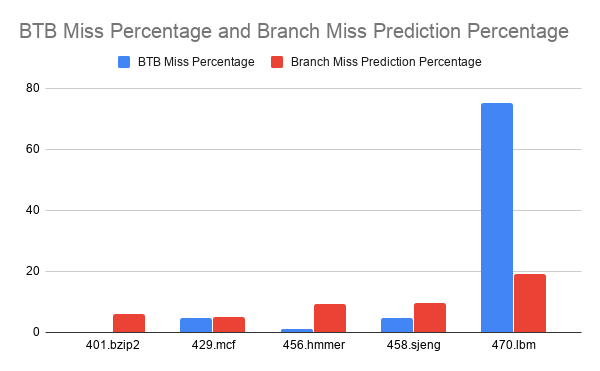
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETERS** | **401.bzip2** | **429.mcf** | **456.hmmer** | **458.sjeng** | **470.lbm** |
| BTB Miss % | 0.039517 | 0.586867 | 0.162733 | 2.744171 | 0.892387 |
| Branch Miss Prediction % | 5.76032 | 5.3886 | 10.79926 | 8.4639 | 23.86433 |
| L1 Data Misses | 4712756 | 10327189 | 311 | 942325 | 189 |
| L1 Instruction Misses | 986 | 690 | 445 | 2378 | 332 |
| L2 Cache Misses | 2938271 | 7758242 | 631 | 8175283 | 540 |
| CPI (System) | 1.5644 | 2.4477 | 1.00125234 | 2.32794 | 1.0000968 |



Comparison figure for Bi-Mode Branch Predictor

**Tournament Branch Predictor:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETERS** | **401.bzip2** | **429.mcf** | **456.hmmer** | **458.sjeng** | **470.lbm** |
| BTB Miss % | 0.136522 | 4.637573 | 1.077412 | 4.773571 | 75.311463 |
| Branch Miss Prediction % | 5.9542 | 4.97278 | 9.42718 | 9.79563 | 19.02369 |
| L1 Data Misses | 4712654 | 10327041 | 287 | 942220 | 189 |
| L1 Instruction Misses | 979 | 673 | 442 | 2369 | 331 |
| L2 Cache Misses | 2931639 | 7751382 | 611 | 8175232 | 538 |
| CPI (System) | 1.05633 | 2.4467 | 1.000772 | 2.3269 | 1.0000967 |



Comparison figure for Tournament Branch Predictor

As suggested in project description size of Local predictor, Global predictor and choice predictor is varied and the BTB miss percentage (BTBM%) and Branch misprediction rate (BMiss%) is mentioned in the table below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TYPE** | **Local** | **Global** | **Choice** | **401.bzip2** | **429.mcf** | **456.hmmer** | **458.sjeng** | **470.lbm** |
| BTBM%  BMiss% | 1024 | 4096 | 4096 | 0.136  5.95 | 4.637  4.972 | 1.077  9.42 | 4.77  9.79 | 75.311  19.02 |
| BTBM%  BMiss% | 1024 | 2048 | 2048 | 0.134  5.97 | 5.897  5.091 | 1.082  9.77 | 4.98  10.23 | 74.06  19.77 |
| BTBM%  BMiss% | 1024 | 2048 | 4096 | 0.140  5.96 | 4.6447  5.045 | 1.267  9.513 | 4.82  10.08 | 77.18  19.44 |
| BTBM%  BMiss% | 1024 | 2048 | 8192 | 0.141  5.94 | 4.708  5.013 | 1.78  9.578 | 4.58  9.87 | 78.25  19.38 |
| BTBM%  BMiss% | 1024 | 4096 | 2048 | 0.13  5.97 | 6.35  5.019 | 1.13  9.4 | 4.81  10.05 | 76.4545  19.4 |
| BTBM%  BMiss% | 2048 | 2048 | 8192 | 0.142  5.93 | 4.6923  4.95 | 1.08  8.36 | 7.31  14.4 | 75.89  20.63 |
| BTBM%  BMiss% | 2048 | 8192 | 8192 | 0.135  5.99 | 4.609  4.804 | 0.367  8.078 | 4.61  9.24 | 69.87  18.94 |
| BTBM%  BMiss% | 2048 | 2048 | 2048 | 0.129  5.96 | 6.68  5.03 | 0.87  8.521 | 5.77  12.82 | 71.45  19.998 |
| BTBM%  BMiss% | 2048 | 2048 | 4096 | 0.139  5.95 | 4.605  4.977 | 1.793  8.42 | 4.74  9.78 | 77.86  19.13 |
| BTBM%  BMiss% | 2048 | 4096 | 8192 | 0.1395  5.92 | 4.87  4.705 | 1.0378  8.15 | 4.51  9.41 | 75.107  19.07 |

Conclusion

With the increase in buffer size we observe that the branch misprediction rate is decreasing more in comparison to the change in BTB miss. This is because if size of the local and global predictors are increased then we can store more combinations of pattern history and it will lead to better results due to spatial and temporal locality.

Also it was observed that tournament predictor works better than the other two predictors in most of the cases.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------