ECEC-412/621

Project 1: Implementing gShare branch prediction and evaluating its performance using ML workloads.

Instructor: Anup Das
TAs: Shihao Song, Adarsha Balaji

1 Introduction

This project is intended to be a comprehensive introduction to branch predictors. There are two parts to this project:

- In part one, you will evaluate the performance of two provided branch predictors with three AI workloads from SPEC CPU 2017.
- In part two, you will design a gshare branch predictor and evaluate its performance against part one.

2 Development Environment

- Operating System: Linux.
- Code-base: `https://github.com/Shihao-Song/Computer-Architecture-Teaching`. Please *git clone* the repository to your Linux machine:

```
$ git clone https://github.com/Shihao-Song/Computer-Architecture-Teaching
```

3 Branch Predictor Framework Overview

1. You will be working under directory *Computer-Architecture-Teaching/C621/Branch_Predictor*. To navigate to the directory:

```
$ cd Computer-Architecture-Teaching/C621/Branch_Predictor/
```

2. To compile and run the simulator:

```
$ make
$ ./Main sample.cpu_trace
```

3. You should be able to see the following output:

```
Number of correct predictions: 8467
Number of incorrect predictions: 1533
Predictor Correctness: 84.669998%
```

4. *sample.cpu_trace* is part of the *leela* AI workload from SPEC CPU 2017. To take a look at the format of the trace file:

```
$ head -10 sample.cpu_trace
94706322334810 B 1
94706322334854 B 0
94706322334863 B 0
94706322334868 B 1
94706322407179 B 0
94706322407214 B 1
94706322443146 B 0
94706322443146 B 0
94706322406110 B 1
94706322406227 B 0
```

Each entry is composed of three components:

- PC: the program counter of the instruction, e.g., 94706322334810.
- Instruction type: B, means the instruction is a branch instruction.
- The **correct** branch direction: 0 - the branch is not taken; 1 - the branch is taken. The simulator relies on this information to determine the correctness of the prediction.

4 Sample Branch Predictors

Two types of branch predictor have been provided: a *two-bit local* predictor and a *tournament* predictor. Please read through *Supplement One and Two* for more details.

4.1 Configure a Two-bit Local Predictor

The following steps illustrate how to configure a 2-bit local predictor for your simulator.

1. Open *Branch_Predictor.h*:

```
$ vim Branch_Predictor.h
```

2. Make sure *TOURNAMENT* stays *commented*:

```
// Predictor type
define TWO_BIT_LOCAL
// #define TOURNAMENT
```

3. You can change the configurations such as the *size of the local predictor* and the *counter precision* in *Branch_Predictor.c*

```
const unsigned localPredictorSize = 2048;
const unsigned localCounterBits = 2;
```

4. Re-compile and run the simulator:

```
$ make clean
$ make
$ ./Main sample.cpu_trace
```

4.2 Configure a Tournament Predictor

The following steps illustrate how to configure a tournament predictor for your simulator.

1. Open *Branch_Predictor.h*:

```
$ vim Branch_Predictor.h
```

2. Make sure *TOURNAMENT* is *un-commented* and *TWO_BIT_LOCAL* is *commented*:

   ```
   // Predictor type
   // #define TWO_BIT_LOCAL
   #define TOURNAMENT
   ```

3. You can change the configurations in *Branch_Predictor.c*

   ```
   const unsigned localPredictorSize = 2048;
   const unsigned localCounterBits = 2;
   const unsigned localHistoryTableSize = 2048;
   const unsigned globalPredictorSize = 8192;
   const unsigned globalCounterBits = 2;
   const unsigned choicePredictorSize = 8192; // Keep this the same as globalPredictorSize.
   const unsigned choiceCounterBits = 2;
   ```

4. Re-compile and run the simulator:

   ```
   $ make clean
   $ make
   $ ./Main sample.cpu_trace
   ```

5 Branch Predictor Evaluation

1. AI workloads (Branch Only): `https://www.dropbox.com/sh/4lhpo0xyhlbkexv/AADd6e-MeZmD5ezg1Syp9IrJa?dl=0`.

2. Evaluate the performance of the *two-bit local* predictor with different configurations shown in Table 1.

localPredictorSize	localCounterBits
2048	1
2048	2
4096	2
8192	2
16384	2
32768	2
65536	2

Table 1: Two-bit Local Configuration

3. Which combination gives you the best performance? Please keep this combination for later experiment.

4. Evaluate the performance of the *tournament* predictor with different configurations shown in Table 2.

localHistoryTableSize	globalPredictorSize	choicePredictorSize
2048	8192	8192
4096	8192	8192
4096	16384	16384

Table 2: Tournament Configuration

5. Besides the configurations shown in 2, you are encouraged to try larger table sizes. Which combination gives you the best performance?

6 A gShare Branch Predictor

Please read through *Supplement One* and design a gShare branch predictor.

1. Evaluate your gShare predictor and find out the configuration that gives the best performance.
2. Compare your gShare predictor against the *two-bit local* predictor and the *tournament* predictor. Does your gShare predictor out-perform them?

7 Submission

1. Summarize your experiment in Section 5 and 6. Compile your report in PDF format.
2. All the source codes.
3. Zip above and submit through Bblearn.