Understanding Branch Prediction

Issued : 9th February 2015

Due : 16th February 2015 at 4.00pm

This assignment represents the first practical component of the Computer Architecture module of Inf3. This practical contributes 5% of the overall mark for the module. It consists of a programming exercise culminating in a brief written report. Assessment of this practical will be based on the correctness and the clarity of the solution (see more details below), and on the completeness and clarity of the written report. This practical is to be solved individually to assess your competence on the subject. Please bear in mind the School of Informatics guidelines on plagiarism. You must submit your solutions before the due date shown above. Follow the instructions provided in Section 2 for submission details.

In this practical, you are required to build a simulator to explore Branch Prediction techniques. You are strongly advised to commence work on the simulator as soon as possible. The simulator you produce may also be somewhat useful for the second coursework, which models caches, so you should keep your source code in a safe place where you can find it later.

You are required to document your experiments and inferences from these experiments, along with the structure of your simulator and any additional information in a hand written report and submit it before the deadline.

1. Experiments

A branch predictor tries to guess which way a branch will go before this is known for sure. The purpose of the branch predictor is to improve the flow in the instruction pipeline. In this exercise you are asked to investigate the efficiency of various branch predictors. To do this, you will have to write a simulator which reads a sequence of instruction addresses of branch instructions and whether the branch is taken or not, and simulate the following branch predictors:

1. Static Branch Prediction: Always Taken, Always Not Taken, Profile Guided Predictor.
2. Dynamic Branch Prediction: Two-Level Correlating Predictor (take a look at the GAg design in the paper "*Alternative Implementations of Two-Level Adaptive Branch Prediction*" by Yeh & Patt, ISCA '92).

For each of these predictors, you are required to determine the misprediction rate. For a profile guided predictor, the trace file should initially be profiled as follows: branch addresses taken 50% or more of the time recorded as predicted taken, and the rest predicted as not taken. The trace file should be re-run using these recorded address-specific predictions to obtain the misprediction rates.

Since the Two-Level Correlating Predictor requires additional storage space, maintaining a 2-bit saturating predictor for each of the 2^k different patterns in the Pattern History Table (PHT), you are required to determine the misprediction rate for different lengths of history maintained. Consider K equal to 4, 8, 12 and 16 bits. **For each K, initialize: the Global History Register (GHR) as all not taken (0s); and all 2-bit predictors at weakly not taken (01).** Record the misprediction rates and include any inferences from the these experiments in your report.

You can write a single simulator with command line switches for each predictor (and the length of the history), or a separate simulator for each predictor. You should print the misprediction rate of the branch prediction technique at the end of the simulation. You are required to provide a readme file with the simulator to detail the compilation and execution of the simulator. Remember that your simulator will be compiled/executed on a DICE machine, so please ensure it works on DICE.

In order to carry out your experiments, you will be provided the branch instruction trace of gcc and mcf, which are part of the SPEC2006 benchmark suite. The trace files include around 500k branch instructions. Each line of the trace file provides the address of the branch instruction followed by a binary digit signifying whether the branch has been taken or not. The instruction addresses are 48 bits long. An example is provided below. Additionally, you may also generate some sample traces to verify the code that you have written.

B 47aa6d1d 1
B 47a72326 0
B 46f31ddb 1
B 46f3154b 0

To summarize, in this exercise you are required to perform simulations of various branch predictors. Your simulator is required to read from trace files, and output the misprediction rate. For the Two-Level Correlating Predictor, you have to investigate the variation of the misprediction rate with respect to the length of history maintained. Present your inferences from these experiments in your report.

2. Format of your submission

Your submission should clearly indicate (in both the report and the code) which branch prediction techniques you have simulated completely and which you have only partially completed.

You should submit a copy of your simulator and a soft copy of your report (compressed file titled <cw1-name_matricno.tar.gz>) before 4pm on 16 February 2015, using the command

```
submit car 1 cw1-name_matricno.tar.gz
```

at a command-line prompt on a DICE machine. Make sure that your simulator is accompanied by a readme file detailing the compilation and execution procedure of your simulator.

The report should contain the following details:

1. Written description of the internal structure and workings of your simulator (around 1 page depending upon complexity), explaining clearly which parts are complete and which incomplete.

2. A report detailing the experiments you ran and giving a critical summary of the results (around 1-2 pages including results). You should provide enough information to make the experiments repeatable. Present your data clearly and concisely. Also present any inferences you have obtained from the results of these experiments, with reasons for these inferences.

3. Marking Scheme

- Static Branch Prediction - 45%
  - 15% each for correct miss rates of each static predictor type and correct methods for arriving at these results
- Two-Level Correlating Predictor - 32%
  - 8% for correct miss rates of each history length and correct methods for arriving at these results
- Clarity of code within the simulator(s) - 3%
- Report on the Simulator - 8%
  - 5% for stating internal structure – this should be an overview of the major components structured in a logical manner
  - 3% for readme file and stating which parts are complete and which incomplete
- Report on Experiments - 12%
  - 4% on stating which experiments were run
  - 4% on critical summary of the results and clarity of presentation
  - 4% on inferences drawn from the results and reasons for these inferences

Your mark will then be scaled to be worth 5% of the course.

In the case of incomplete/inaccurate predictors efforts will be made to give partial credit for what work has been completed. Clear comments explaining the important tasks (and where they are missing/failing) will increase your chances of receiving credit.

4. Working in Stages

This section contains some suggestions as to how you might tackle the work in stages. There is no obligation to follow these, but your report must make it clear exactly what your simulator can handle.

Stage 1

First write a simulator to read from trace files and check whether the branches recorded in it are taken or not. This will give you the statistics for some of the static prediction techniques.

Stage 2

For the Two-Level Correlating Predictor, start by developing a PHT of 2-bit predictors for the minimum history length (K=4). Use single branch address hand written traces to verify the predictor. Then use the rest of values for K (8, 12, 16).

Stage 3

Once you have verified the simulator, use the provided trace files to evaluate the various branch predictors you are required to simulate. Add the results and any inferences you derive from it in your report.

5. Reporting Problems

Send email to jcanore@inf.ed.ac.uk for any issues regarding the assignment.