ECE 486/586
Computer Architecture
Winter 2014

Project 2
Branch Prediction

Overview

In this project your team will simulate a branch predictor. You are required to implement a tournament predictor based upon the design employed by the Alpha 21264 described in Kessler, R.E., "The Alpha 21264 Microprocessor", IEEE Micro, March/April 1999, pp 24-36.

However, your branch predictor will depart from that described in the Kessler paper in that it will not employ a second level local history table. Instead the local history predictors will be directly indexed by the PC.

Framework

You'll be provided with a C++ software framework including several execution traces. The framework will read a trace and call your predictor with a record containing information about each branch. Your predictor will make a prediction and then be told the actual outcome of the branch. Upon completion of the trace, the framework will print out statistics including your misprediction rate (expressed as mispredicts/1000 branches).

Source code of the framework, three traces (one each floating point, integer, multimedia), and a makefile will be provided to you on the resources page of the course web site. You are responsible for coding two functions:

```
bool get_prediction(const branch_record_c* br, const op_state_c* os)
void update_predictor(const branch_record_c* br, const op_state_c* os, bool taken)
```

The framework will open a trace file and then call your get_prediction() passing it a record with the current branch information. Your function will return true if you're predicting the branch to be taken and false if not-taken. The framework will then call your update_predictor() letting it know the actual result of the branch so that you can update branch history, etc. The branch_record is defined in the tread.cc file in the framework. You can ignore op_state_c.

You do not need to understand details of the framework. Just add your functions to the predictor.cc file.

The branch_record_c looks like this:

```
class branch_record_c {
public:
    branch_record_c();
    ~branch_record_c();
    void init();                // init the branch record
    void debug_print();         // print info in branch record (for debugging)
    uint   instruction_addr;    // the branch's PC (program counter)
    uint   branch_target;       // target of (taken) branch; unconditional are always taken
    uint   instruction_next_addr;// PC of the instruction following the (untaken) branch
    bool   is_indirect;         // true if target is computed; false if it's PC-relative
                                // returns are also considered indirect
    bool   is_conditional;      // true if branch is conditional; false otherwise
    bool   is_call;             // true if branch is a call; false otherwise
    bool   is_return;           // true if branch is a return; false otherwise
};
```

You can download the framework from the course web site (Resources page) and uncompress it. Do not copy or uncompress the trace files – the framework dynamically uncompresses them as needed. Use them in place.

Grading Criteria

The project is worth 100 points as follows:

Produces correct results	50
Code (quality, readability)	25
Presentation/Report	25

Your presentation should not exceed 20 minutes and must include a description of your algorithm, the rationale, a "space" budget that accounts for all storage used, a description of the implementation, a description of your testing procedures, and the statistics on the public benchmarks, ending in a demonstration showing the compilation and execution of your code on the competition benchmarks. A written report summarizing the above information must be turned in at the time of your presentation.

Due Date

You need to schedule a 30-minute slot for sometime Monday, March 17th through Wednesday, March 19th using this Doodle link http://doodle.com/bb4wdw2wsp4fz5fe. Be sure to enter the family (last) names of all the teammates. During your slot you will give a brief presentation of your predictor and submit the code for the predictor for compilation and execution. Demos will be in the FAB Intel PC Lab.