Advanced Computer Architecture Mini Project

Title: Implementation and Evaluation of 2-Level Adaptive Branch Prediction in SimpleScalar 3.0

Author: Ashish Kumar
Roll No: 25CS06013
Institute: IIT Bhubaneswar

Project Overview

This mini project explores the working and modification of the **2-Level Adaptive Branch Predictor** in the **SimpleScalar 3.0** architecture simulator. The goal was to enhance understanding of how modern processors predict conditional branches, improve prediction accuracy, and analyze how different predictor configurations affect performance metrics.

Objective / Aim

- To implement and modify a **2-Level Adaptive Branch Predictor**.
- To analyze **bpred_2lev.addr_hits** and **bpred_2lev.dir_hits**.
- To compare prediction performance with and without modifications.
- To understand how local and global history affect prediction accuracy.

Software Used

- **SimpleScalar 3.0** – Architectural simulator for Alpha ISA.
- **GCC** – For compiling simulator source code.
- **SPEC-like Benchmarks** – compress95, anagram, cc1, go.

Implementation Steps

Step 1: Environment Setup

- Cloned and configured `simplesim-3.0` source files.

- Verified baseline build using:

  ```
  make sim-bpred
  ```

Step 2: Core Files Modified

File	Purpose
`bpred.c`	Core logic of the branch predictor (prediction and update mechanisms).
`bpred.h`	Predictor parameters, structure definitions, and function declarations.
`sim-bpred.c`	Simulation driver and result statistics collector.
`misc.c` / `misc.h`	Utility functions for debugging and error handling.

Step 3: Adding Confidence Mechanism in bpred_dir_create()

```
dp->config.two.confidence = calloc(dp->config.two.l2size, sizeof(unsigned char));
dp->config.two.last_outcome = calloc(dp->config.two.l2size, sizeof(unsigned char));
```

This was used to introduce confidence-based prediction tracking.

Step 4: Adaptive Update Logic

To modify how the history interacts with branch outcomes:

```
if (bpred->class == BPred2Level) {
    int history = (bpred->config.twolevel.l1table[history_index]) & history_mask;
    int prediction = bpred->config.twolevel.l2table[history];
    if (taken)
        prediction = SATURATE_UP(prediction);
    else
        prediction = SATURATE_DOWN(prediction);
}
```

This section controls how prediction counters are updated after each branch.

Step 5: Experimental Variation

To introduce XOR-based history indexing (to increase correlation):

```
history = (history ^ (addr >> 2)) & history_mask;
```

Commands Used

Compilation:

```
make sim-bpred
```

Execution (Example):

```
./sim-bpred -bpred 2lev -bpred:2lev 1 1024 8 0 -bpred:btb 512 4 -bpred:ras 8 tests-alpha/bi
```

If `2levconf` predictor was removed or renamed, run using standard 2lev.

Evaluation and Results

Benchmark	Modification	bpred_2lev.addr_hits	bpred_2lev.dir_hits
Compress.95	No	0.7700	0.8357
Compress.95	Yes	1.1133	0.6315
anagram	No	0.5788	0.7386
anagram	Yes	0.6687	0.4448
cc1	No	0.8326	0.8577
cc1	Yes	1.1764	0.6328
go	No	0.9112	0.9177
go	Yes	1.4012	0.7005

Observations and Analysis

- Introducing XOR-based history improved **address correlation**, increasing `bpred_2lev.addr_hits`.
- However, it reduced **directional accuracy (`dir_hits`)**, indicating prediction overfitting to specific address patterns.
- Modified predictors may perform better for workloads with repeating address patterns, but worse for random branches.
- This experiment demonstrates the **trade-off between local vs global history prediction**.

Usage Details

Command	Description
`make sim-bpred`	Compiles the simulator after code modification.
`./sim-bpred -bpred 2lev`	Runs simulation using the 2-level predictor.
`bpred.c`	Modify prediction and update rules.
`bpred.h`	Define new predictor structures.
`misc.c`	Handles debug and error messages (updated function parameters).
`sim-bpred.c`	Registers stats and prints prediction data.

Conclusion

- Implemented a **2-Level Adaptive Branch Predictor** and experimented with confidence-based updates.
- Observed that history correlation can improve prediction hits but also reduce stability.
- Learned practical aspects of branch predictor design and evaluation using **SimpleScalar**.

Future Enhancements

- Implement **Hybrid Predictors** combining local and global tables.
- Add dynamic predictor selection mechanisms.
- Include visualization for predictor behavior and hit/miss tracking.

Acknowledgement

This mini project was completed as part of **Advanced Computer Architecture** coursework under the guidance of faculty at **IIT Bhubaneswar**.