Performance of several branch predictor types and different RAS configurations

Advanced Computer Architecture

Simulation project

First semester, 2009/2010

Done by: Dua'a AL-Najdawi

Date: 20-1-2010

Design options

My research was about the performance of several branch predictor types and different Return address stack (RAS) configurations, so I used a **sim-outorder** in simple scalar according to my attend in the project.

It supported these predictor types:

- **nottaken** : always predict not taken
- **taken** : always predict taken
- **perfect** : perfect predictor
- **bimod** : bimodal predictor (BTB w/ 2 bit counters)
- **2lev** : 2-level adaptive predictor
- **Comb** : combination between the bimodal and 2-level adaptive predictor

I interested in **bimodal predictor**, 2-**level predictor** and the **combination** between of them, also in my project I made different configurations for the **RAS size**. Their default configurations and their commands to change their options are seen below:

```
-bpred                  bimod # branch predictor type
{nottaken|taken|perfect|bimod|2lev|comb}
-bpred:bimod     2048 # bimodal predictor config (<table size>)
-bpred:2lev      1 1024 8 0 # 2-level predictor config (<l1size> <l2size> <hist_size>
<xor>)
-bpred:comb      1024 # combining predictor config (<meta_table_size>)
-bpred:ras                  8 # return address stack size (0 for no return stack)
-bpred:btb       512 4 # BTB config (<num_sets> <associativity>)
```

Firstly I run a sim-outorder with **20 million instruction fastforward** and **50 million instruction max**, by using this command:

```
-max:inst 50000000 -fastfwd 20000000
```

And I saved each output of each experiment in txt file for each workload separately, using the below command .Also I accompanied it in CD:

```
-redir:sim sim_output_file
```

Here are 8 experiments, separated it to two parts:

PART 1: Performance of several branch predictors

Experiment 1:

Using the **bimodal branch predictor** with a table size of 256 and its output saved in output2 txt file in the CD, using the following command:

```
-bpred bimod -bpred:bimod 256
```

Experiment 2:

Using the **2-level predictor type** with l1=1, l2=256 and its output saved in output3 txt file in the CD:

```
-bpred 2lev -bpred:2lev 1 256 4 0
```

Experiment 3:

Change the predictor type to **combining predictor** with the same size, and its output saved in output4 txt file in the CD:

```
-bpred comb -bpred:comb 256 -bpred:bimod 256 -bpred:2lev 1 256 4 0
```

PART 2: Different RAS configurations

Experiment 1:

Change the return address stack (RAS) size to 2 in bimodal predictor, and its output saved in output8 in the CD:

```
-bpred bimod -bpred:bimod 256 -bpred:ras 2 -bpred:btb 64 2
```

Experiment 2:

Change the return address stack (RAS) size to 4 in bimodal predictor, and its output saved in output5 in the CD:

```
-bpred bimod -bpred:bimod 256 -bpred:ras 4 -bpred:btb 64 2
```

Experiment 3:

Change the return address stack (RAS) size to 8 in bimodal predictor, and its output saved in output2 txt file in the CD:

```
-bpred bimod –bpred:bimod 256 –bpred:ras 8 –bpred:btb 64 2
```

Experiment 4:

Change the return address stack (RAS) size to 16 in bimodal predictor, and its output saved in output6 txt file in the CD:

```
-bpred bimod –bpred:bimod 256 –bpred:ras 16 –bpred:btb 64 2
```

Experiment 5:

Change the return address stack (RAS) size to 32in bimodal predictor, and its output saved in output7 txt file in the CD:

```
-bpred bimod –bpred:bimod 256 –bpred:ras 32 –bpred:btb 64 2
```

Simulator

SimpleScalar v 3.0, an execution driven simulator that implements a very detailed out-of-order issue superscalar processor with a two-level memory system and speculative execution support. It also has the characteristics to change the branch predictor options, as we saw in the Experiments.

As we know the simple scalar has many simulators different with each other, here I used sim-outorder which is called Detailed Performance simulator. It generates timing statistics for a detailed out-of-order issue processor core with two-level cache memory hierarchy and main memory, and here I interested in IPC and branch predictor hit rate which is supported in sim-outorder.

Workload

There are 26 SPEC2000 Benchmarks 12 Integer and 14 Floating Point. I pick 3 of them two floating point: **equake** , **ammp** and one integer: **gcc**.

Its binaries are for the PISA instruction sets, I included it with their inputs in the CD.

Results:

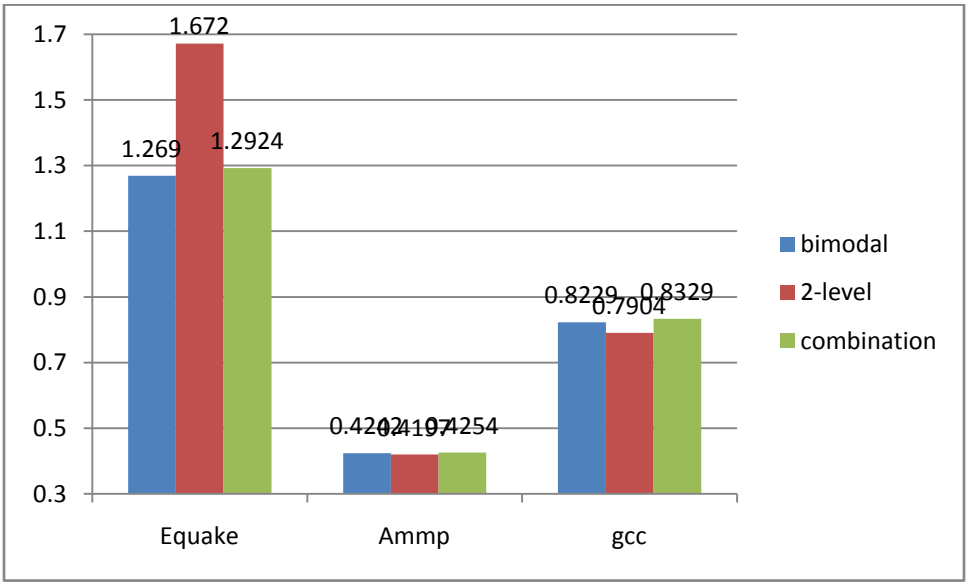
PART 1: Performance of several branch predictors

I used different types of the branch prediction: bimodal prediction, 2-level prediction, combination of the bimodal prediction and 2-level prediction. All have the same size which is 256 (2-level prediction l1=1, l2=256).

The comparison between the different types of branch prediction was in IPC and the branch direction prediction rate and their results are shown in Table 1 and Table 2 respectively also are implemented in graph 1 and graph 2.

Branch prediction type	Equake	Ammp	gcc
bimodal	1.2690	.4242	.8229
2-level	1.672	.4197	.7904
combination	1.2924	.4254	.8329

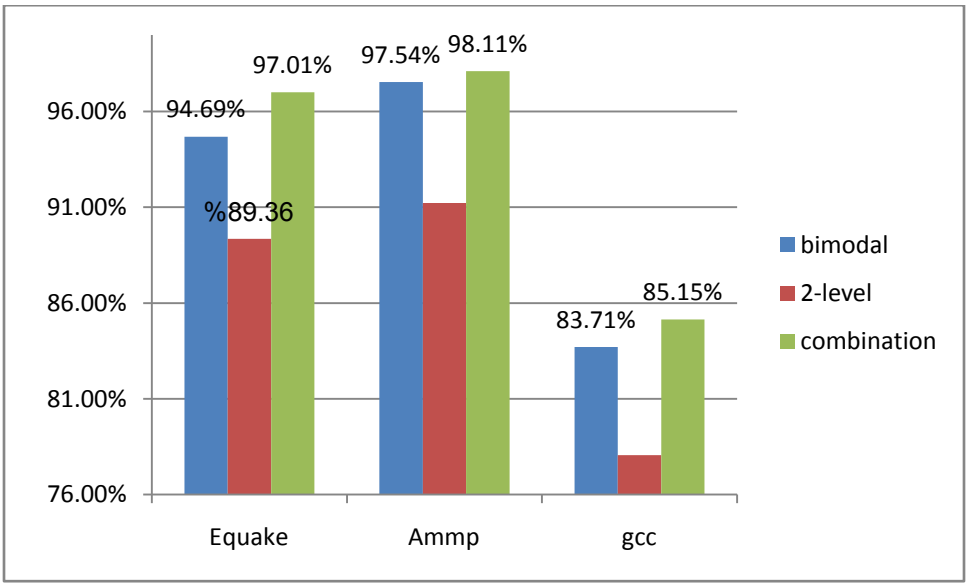
Table 1: IPC for different types of branch prediction



Graph 1: IPC for different types of branch prediction

	Equake	Ammp	gcc
bimodal	94,69%	97,54%	83,71%
2-level	89,36%	91,23%	78,05%
combination	97,01%	98,11%	85,15%

Table 2: prediction hit rate to different types of branch prediction



Graph 2: prediction hit rate to different types of branch prediction

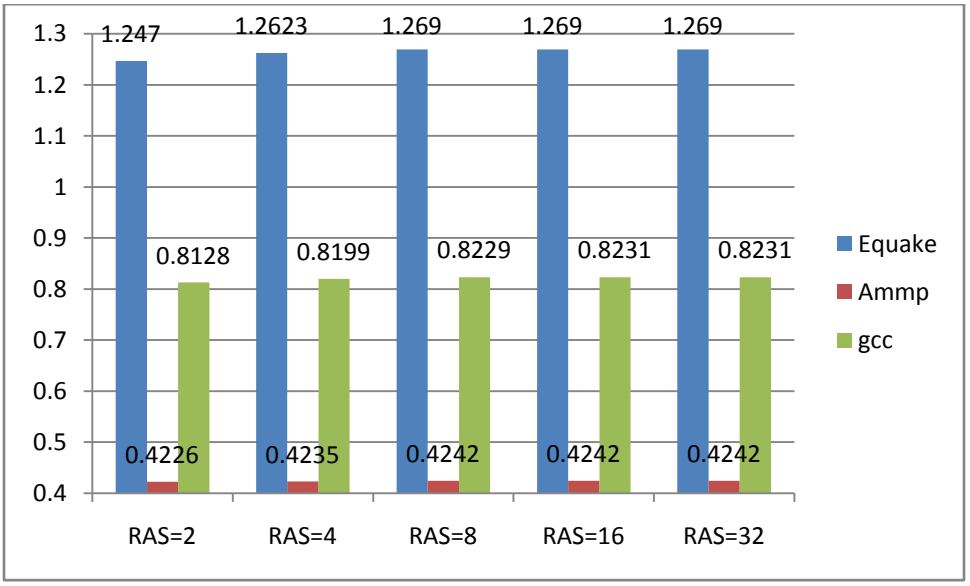
PART 2: Different RAS configurations

The Return Address Stack (RAS) Changes to different sizes: 2,4,8,16,32 (Of course the size of RAS will be the power of 2). And the type of the branch prediction is the **bimodal branch prediction** and its table size is 256.

The comparison between its different sizes in **IPC** and the **branch address prediction rate** are shown in Table 3 and Table 4 respectively also is implemented in graph 3 and graph 4.

	Equake	Ammp	gcc
RAS=2	1.2470	.4226	.8128
RAS=4	1.2623	.4235	.8199
RAS=8	1.2690	.4242	.8229
RAS=16	1.2690	.4242	.8231
RAS=32	1.2690	.4242	.8231

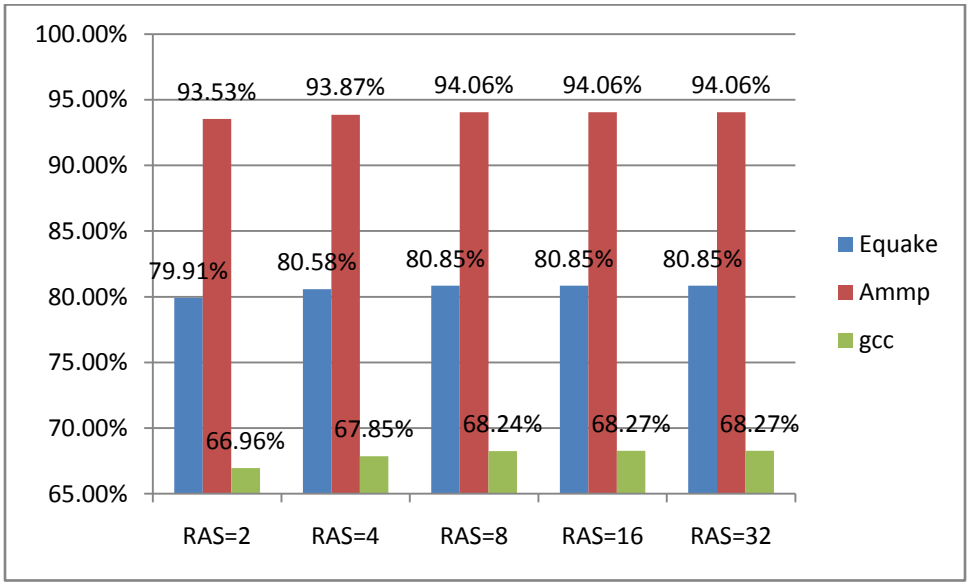
Table 3: IPC for different sizes of RAS



graph 3: IPC for different sizes of RAS

	Equake	Ammp	gcc
RAS=2	79,91%	93,53%	66,96%
RAS=4	80,58%	93,87%	67,85%
RAS=8	80,85%	94,06%	68,24%
RAS=16	80,85%	94,06%	68,27%
RAS=32	80,85%	94,06%	68,27%

Table 4: prediction hit rate for different sizes of RAS



Graph 4: prediction hit rate for different sizes of RAS

6. Conclusion

We saw the performance of the bimodal predictor is better than the performance of 2-level adapter predictor. When we combine between them the performance all will be better, prediction hit rate and IPC increase.

When we increase the size of RAS (2, 4, 8) the IPC and the prediction rate increase so all the performance will enhance. This result up to RAS=8, but after that (RAS=16 or =32) almost the results will be the same so the best size of RAS is 8 which is used now in the superscalar processor.

7. References

1-Speculative Return Address Stack Management Revisited, HANS VANDIERENDONCK and ANDR´E SEZNEC,**Transactions on Architecture and Code Optimization (TACO)**, November 2008

2-**Comparison of branch prediction schemes for superscalar processors ,ICEEC 2004 , Youssif, A.A. Ismail, N.A. Torkey, F.A.**

3-http://bwrc.eecs.berkeley.edu/Classes/CS252/Projects/Reports/terry_chen.pdf

4-http://harryscode.blogspot.com/2008/10/installing-simplescalar.html

5-http://students.cse.tamu.edu/msahn/csce614/hw1.pdf

6-http://www.simplescalar.com/docs/simple_tutorial_v2.pdf

7-Arguments:http://students.cse.tamu.edu/msahn/csce614/spec2000args.tgz

8-Binaries:http://www.eecs.umich.edu/mirv/benchmarks/gcc2000.v3.tar.gz.

CD Contents:

- Report
- Simple Scalar V *3* simulator
- Simple Scalar tutorial and installation guide
- Workloads including:
  - A-Arguments
  - B-Binaries
- Tutorial of simple Scalar installations and run it with different parameters
- Output of my simulation