Computer Architecture Lab Assignment 2: Branch Prediction and Predicated Execution

Nanjing University, Spring, 2017

I. Introduction

In this assignment, you will study two important methods for control hazard handling: Branch Prediction and Predicated Execution. You will use the gem5 simulator to study these two techniques. We recommend that you run the gem5 simulator under a Linux system. We will use the gem5 to simulate a system using the X86 ISA.

II. Branch Prediction vs. Predicated Execution in X86

A. Branch and Conditional Move

Consider the following piece of C code:

```
if (x<y)
      x = y;
```

There are at least two ways to generate pseudo assembly code for this:

- Using branches:

```
    compare x, y
    jump if not less to L
    move x, y
L:
```

- Using conditional move:

```
    compare x, y
    conditionally move x to y
```

We will study the performance of these two approaches in this lab assignment to get a better understanding of control hazards.

B. Running on Your PC

In X86 ISA, there is a conditional move instruction called `cmov`. Please refer to the post on cmov from Linus Torvalds, the creator and maintainer of the Linux operating system. Linus has provided a short piece of C code for measuring the performance of branches and conditional moves.

In the first step, you should:

- Copy the code as cmov.c
- Compile two versions of the program using gcc, *e.g.*
  - for the branch version: `gcc -Wall -O2 cmov.c -o ncmov`
  - for the predicated execution version: `gcc -DCMOV -Wall -O2 cmov.c -o cmov`
- Run each version and record the running time on your PC, *e.g.*
  - `time ./cmov`

Is your result similar to that provided by Linus? Why?

Hint 1: You should first understand the C code which uses the embedded ASM.

We provide a link explaining this in Chinese.

Hint 2: You can use the "-S" option to ask gcc to generate assembly codes and check if everything is OK, *e.g.*, `gcc -Wall -O2 -S cmov.c -o ncmov.s`

Hint 3: You need to understand the basics of X86 assembly and the meaning of instructions like `testl` and `cmovne`.

C. Running on gem5

We will take a closer look at these two approaches using gem5. In gem5, we have different branch prediction algorithm in the OoO CPU model. Therefore, to study the branch prediction performance, we should use the "DerivO3CPU" model.

We recommend that you modify the "se.py" to enable branch prediction options. The original "se.py" does not provide branch predictor selections. You may need to add a new option, *e.g.*, `--bpred`, to allow branch predictor selection in the configuration:

- Add `parser.add_option("--bpred", type="str", default="TournamentBP")` in the "se.py"
- Change the predictor for each CPU:

```
brach_predictor = locals()[options.bpred]
system.cpu[i].branchPred = brach_predictor()
```

We also provide a patch file for your reference. You can apply the patch to the original "se.py" that can be found under `gem5/configs/exmaple/se.py`. You can apply the patch by the command:

```
patch < bpred.patch
```

You can invoke the configuration using:

```
build/X86/gem5.opt configs/example/se.py --cmd=your/test/prog
  --cpu-type=DerivO3CPU --caches --l2cache --bpred=LocalBP
```

Note: Please first change the number of iterations to 1,000,000 before running on gem5. Otherwise, it will take a lot of time for simulation.

There are some predefined branch predictor in gem5, including:

- TournamentBP: the default branch predictor in O3CPU
- LocalBP: local 2-bit saturated counter predictor

Please run the two versions of your program, using the TournamentBP and LocalBP. Please record the running time of each setting. Look at `system.cpu.commit.branches` for the number of branches actually committed. Look at `system.cpu.iew.branchMispredicts` for miss predicted branches. You may also look at `system.cpu.iew.predictedTakenIncorrect` and `system.cpu.iew.predictedNotTakenIncorrect` for more information.

Hint 4: Each iteration, there will be one branch for the loop and one/or zero branch for the move. The loop branch has a different behavior than the move branch.

Hint 5: There are a few tens of thousands branches for other part of the program, *e.g.*, printf and loading.

Hint 6: You may try to change the BIT setting in the C code. Note that `testl` takes the BIT as a bit mask for the testing. If BIT = 1, you will get branch pattern of TNTNTNTN. . .. If BIT = 2, you will get branch pattern of TTNNTTNNTTNN. . .. If BIT = 3, you will get branch pattern of TNNNTNNNTNNN. . .. Try different patterns as you wish.

D. Extra-Credit Problem

This part is optional. If you are interested in branch prediction, you may try to implement your own branch predictor and test it using gem5. Please follow the part II of gem5 tutorial provided by University of Wisconsin-Madison.

You can look at the implementation of LocalBP in `gem5/src/cpu/pred/2bit_local.cc`, the implementation of TournamentBP in `gem5/src/cpu/pred/tournament.cc`, related header files, and other related classes. You may also look at the "BranchPredictor.py" to see how parameters are set for these branch predictors. Try to add a new branch predictor, either the two level global branch history or two level local branch history would work quite OK for our case. Run the program using your own predictor and gives the prediction results.

III. Submission Requirements

You should submit both your code and your report in a zip archive. Your codes include the test program, configuration scripts, execution scripts, and your raw "stats.txt" file.

You should also answer the following questions based on your observations:

1) What is the running time of the two versions on your PC, why?
2) What is the running time of the two versions on gem5 with TournamentBP, why?
3) What is the running time of the two versions on gem5 with LocalBP, why?

You can write your report in Chinese.

Please submit both your code and your assignment report in a zip archive to the course web site before the deadline. Any late submission will be rejected.

IV. Acknowledgement

This lab assignment is adapted form homework of "CS 752: Advanced Computer Architecture I", University of Wisconsin-Madison.