DAT105 Computer Architecture
Study Period I, 2019
Department of Computer Science and Engineering
Chalmers University of Technology, Göteborg, Sweden

Laboratory Assignment 2: Exploring the Impact of Branch Prediction and Instruction-Level Parallelism (ILP) on Processor Performance

Contributors: Mehrzad Nejat, Mohammad Waqar Azhar, Per Stenstrom

Learning outcome: The main objective of this lab assignment is two-fold: (1) to understand the impact of branch prediction on processor performance and (2) to explore several micro-architectural techniques to exploit Instruction-Level Parallelism (ILP). At the end of this assignment, you should be able to

- Explore the design space of popular branch predictors and assessment of the impact on performance of the prediction accuracy
- Evaluate the impact of dynamic scheduling, multiple issue and hardware-based speculation on performance

Background: You must have finished the first assignment before starting this one. Therefore, you should already be familiar with the simulation environment and the basic simulation methodology. You should also have some basic knowledge about branch prediction, dynamic scheduling and out of order (OoO) execution, and multiple-issue superscalar architectures as covered by lecture 2-4 in the course.

In the first assignment, you have studied the impact of basic cache parameters (cache size, associativity, and block size) on overall performance. You tried to get close to the maximum possible speedup and proposed a new optimal configuration for instruction and data caches.

In this assignment, we continue to explore other performance enhancing techniques including branch prediction and different instruction scheduling techniques. Check the related sections of the simulator user guide to see what parameters are available and what each parameter refers to. You may need to have another look at the course material and references to make sure that you understand the concept and functionality of these configurations and parameters. If you have any doubts, discuss it with the teaching assistants.

Evaluation: You must write a report that includes: A brief description of the problem and the method you used, important assumptions (if any), simulation results and observations, your design choices and the reason behind them. Detailed report guidelines will be available on Ping-Pong.

Task 1: Branch Prediction

Start from the last configuration you proposed in the previous lab assignment (OPT2) as a base for this task and call it “base2”. If you check the branch predictor type, you will notice that it is set to “not taken”. This is a static branch predictor. There are also several types of dynamic predictors: Bimodal, two-level, and combined. Think about the following questions:

- Static branch predictors always assume the same prediction for all the branches. Therefore, they have a minimum overhead. But, what about their prediction accuracy compared to dynamic predictors? How big is the difference?
- How much variation do you expect in branch prediction accuracy among different types of dynamic predictors?
- How big is the effect of the accuracy of branch predictors on the overall performance?
- How much does the answer to the previous questions depend on the application?

Similar to the methodology in the previous assignment, start by finding the maximum possible speedup when improving branch prediction. You can use the ‘Perfect’ predictor available in the simulator configuration. First, perform the simulations for base2. Then, create new models by changing the branch predictor type[1] to perfect, bimodal, two-level, and combined.

The design space to be explored is specified by the entries in the configuration file. You will notice more configuration details for each branch predictor, plus return address stack (RAS) and branch target buffer (BTB). In the final step try to alter these values to see how much further improvement you can make. But, you are limited to the following changes:

- Doubling the table size for bimodal
- Doubling of the L1 and/or L2 size of the two-level predictor
- Doubling of the table size of combined

Step 1: Devise a methodology for establishing how to explore the design space of branch predictors with respect to performance. The goal is to find the best performing branch predictor. The exploration should take into account the type of branch predictors (bimodal, two-level, combined) and the size of the tables they use.

Step 2: Carry out the design space exploration.

Step 3: State your conclusions on what is the best design point backed up by numbers (tables, diagrams) and state what are the limitations of the methodology.

[1] Do not change anything else

Task 2: Exploiting Instruction-Level Parallelism

In a simple single-issue in-order pipeline, CPI is usually more than one. The reason is that during the execution of the program there are some additional cycles wasted on waiting for memory accesses, recovering from a branch miss prediction, or handling dependencies between instructions. You have already minimized the number of wasted cycles due to memory accesses and branch prediction inaccuracy.

The goal now is to reduce the number of wasted cycles due to instruction dependencies by improving the processor core and allowing out-of-order execution. Then, you will further utilize ILP by increasing the pipeline width such that multiple instructions can be issued and executed in parallel at the same time. Use the best configuration you proposed in the previous task as a base for this task and name it "base3.txt". Similar with the previous assignment, improve the design in a structured step-wise fashion.

2.1. Increasing the Functional Units (FU)

In the first step, increase the number of FUs. These FUs are Arithmetic-Logical-Units (ALU) and Multiplier/Dividers for integer and floating-point operations. However, if you have checked the results of the previous simulations more carefully, you have noticed that floating point units are never used. This is because the benchmark programs you are using in this lab use only integer arithmetic. So, you should focus on the number of integer units.

Task 2.1:

Goal: Establish the number of ALUs and Multiplier/Dividers needed to maximize performance.

Step 1: Devise a methodology for establishing the impact of the number of functional units on performance

Step 2: Carry out the design space exploration

Step 3: State your conclusion backed up by numbers (tables, diagrams). And discuss the limitations of the methodology.

2.2. Out-of-order execution

In this part, you will simulate a single-issue out-of-order processor (core). Keep decode, issue, and commit widths equal to one and change the issue to out-of-order.

Your task is to study the impact of the size of the Reorder Buffer (RUU in the table) and Load/Store Queue (LSQ in the table) has on the performance for a single-issue out-of-order processor (core).

Parameter	Possible values
IF Queue size	2,4,8
RUU size	16,32,64,128
LSQ	RUU size ÷2
Number of ALUs	1-8
Number of Mults	1-4
Issue wrong path	True/False

Task 2.2:

Step 1: Devise a methodology for establishing the impact of the size of the Reorder buffer on performance

Step 2: Carry out the design space exploration

Step 3: State your conclusion backed up by numbers (tables, diagrams).

2.3. Wide issue

Here, we will go beyond single-issue out-of-order processors and especially consider the impact the issue, decode and commit width have on the performance, where the width is varied between 2 and 8. You should consider how the width may affect the size of the Reorder buffer and the number of functional units.

Task 2.3:

Goal: Establish the width and the number of functional units to maximize performance.

Step 1: Devise a methodology for establishing the impact of the issue/decode/commit width on performance

Step 2: Carry out the design space exploration

Step 3: State your conclusion backed up by numbers (tables, diagrams).

Look back and summarize

At this point, you have improved the performance step-by-step from cache optimization via increasing branch prediction accuracy to exploiting ILP.

- Compare the best CPI you achieved with the CPI of the first base configuration.
- Compare the improvements you made in each task. Which improvements were more effective?
- Reflect qualitatively upon how much you increased the hardware resources to achieve this improvement?