Parallel Computer Architecture: Simulation Project 1

Fall 2020, Instructor: Kubilay Atasu

Date of Assignment: 27.10.2020

Due Date: 17.11.2020

Goal: Functional and performance simulation of a processor architecture that implements dynamic scheduling using Tomasulo's algorithm with a reorder buffer. Assume that the processor can issue a single instruction per clock cycle. However, the processor implements several functional units that can execute in parallel. Execution of some of these functional units can take several clock cycles. These functional units are not pipelined and remain busy while executing. The architecture implements a naïve branch predictor, which predicts that the branches are always taken and speculatively issues and executes the instructions that start from the branch target address until the branch condition is resolved in the execute stage.

Task: Implement your own simulation model of the architecture in C/C++ or Python.

Step 1) When started, your simulator should automatically parse the following input files:

- **Units.txt:** Lists the functional units implemented by the architecture. A functional unit can execute only a single processor instruction at any point in time, but it can be used to execute multiple different types of processor instructions one after another. These instructions are listed in the file for each functional unit. The number of clock cycles each functional unit takes in the execute stage is also provided.
- **Parameters.txt:** Parameters of the architecture, including the number of registers and the size of the instruction window.
- **Instructions.txt:** Instructions supported by the architecture.
- **Program.txt:** Program to be executed on the architecture.

Step 2 (Functional Simulation): Simulate the execution of the program on the processor architecture by interpreting the instructions starting from program address 0. Print the contents of R0 when the execution of the program is complete. Locate the loop in the program. List all the data dependences you have identified in the program. How many iterations have been performed? Explain what this program is trying to do.

Step 3 (Cycle-Accurate Simulation): Perform a cycle-by-cycle simulation of the processor components and report the contents of each component from cycle 0 until the termination of the program using the format given in Report.txt. Note that the instruction window stores a list of instructions to be issued and the reorder buffer stores the instructions that are already issued but not yet retired. Report also the contents of the registers, reservation stations, and the Common Data Bus. Refer to the example covered in the lecture when in doubt. How many cycles did it take? What is the content of R0 when the program terminates?

Deliverables:

- Your detailed answers to the questions posed in Step 2 and Step 3 above.
- Your automatically-generated cycle-by-cycle simulation report in a separate text file.
- Your simulator source codes ready to be executed on the input files provided and able to produce an output file in the format defined in Report.txt. Please provide your source and input files in a single zipped directory. The instructor reserves the right to

make changes in Program.txt, Parameters.txt, and modify the cycle counts given in Units.txt. Your simulator should work correctly even in the presence of such changes.

Notes:

- The project can be done in groups of two. However, each participant should be able to individually explain the source codes and demonstrate the operation of the simulator when requested to do so by the instructor.
- Plagiarism will not be tolerated. Plagiarists will receive zero points from the projects.