CS4110: Computer System Design Lab
Assignment #5
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Submission deadline: 25th Oct, 2015

Superscalar processor simulator: Extend the scalar pipeline processor simulator, designed as part of Assignment #4, to support ***n***-wide (configurable parameter) superscalar processor that can fetch ***n*** instructions per cycle, decode ***n*** instructions per cycle, issue ***n*** instructions per cycle, and commit ***n*** instructions per cycle. Add all the necessary components to support multi-instruction issue, out-of-order execution, in-order commit, operand forwarding, load bypassing, load forwarding, structural hazards handling, and false dependency elimination. Assume that each functional unit has a set (configurable parameter) of reservation stations. Implement the register-renaming concept as discussed in the Tomasulo algorithm. Note that source operands can be available either from a register file or from reservation stations. The simulator needs to take care of precise exception handling.
In order to support superscalar processing, consider the following changes to the scalar pipeline processor:

- The **Register Read** (RD) stage can perform both register read as well as instruction dispatch to reservation stations.
- **Execute** (EX) and **Memory access** (MEM) stages are combined together as a multi-cycle **Execute** stage (refer to the superscalar organization, given in slide 3 of Lectures 23-24).
- The **Write-Back** (WB) stage is split into **Complete** stage and **Retire** stage to deal with ALU/LD and ST instructions, respectively (the roles of these two stages are discussed in Slide 7 of Lectures 23-24).

Assume that the simulator simulates the same set of instructions as was consider for the scalar pipeline processor design.
Assume that when both *load* and *store* requests come to the data cache at the same time, *load* requests will be given priority. Also assume that the architectural register file consists of 8 registers, whose values can be initialized at run-time.
Design the simulator in such a way that user can specify the number of entries in the reservation station, re-order buffer, store buffer, and the latency for each ALU operation. Given a program code (will be supplied at the evaluation time), the simulator has to compute the total number of cycles required to complete the execution of the code, the final result of the computation, and the contents of the architectural register file.