Yi Liu (999736889)
Daiqing Li (1000795357)

ECE552 Lab 3: Dynamic Scheduling with Tomasulo

Total number of cycles with Tomasulo for the EIO traces:

Trace	Tomasulo cycles
gcc	1814118
go	1852943
compress	1979819

Stage descriptions:

In Tomasulo algorithm's main loop, the functions that simulate each stage are called in reverse order relative the sequence that an instruction would go through. This is because the program is written for simulation purposes and does not represent actual hardware implementation, in which everything would be done in parallel. This reverse calling order of the functions mimics hardware implementation such that an instruction would move forward of a maximum of one stage in a single cycle. If done otherwise, then an instruction can move multiple stages in a cycle as these functions are called sequentially in program order in a single iteration of the main loop. An alternative would be to declare extra data structures in the program to save the program's simulation state at each stage (more closely mimic actual hardware), in which case the functions can be called in any order.

fetch_To_dispatch:

Keeping fetching instructions from the trace until a non TRAP instruction is found, given that there the instruction fetch queue(IFQ) is not full (no structural hazard). If successfully fetched a new instruction and inserted into the IFQ, assign tom_dispatch_cycle to the current cycle.

dispatch_To_issue:

First check if IFQ is empty. If it is not, check special case for branch ops. In the case of branch instructions, it is not issued and immediately free it from IFQ. Then we search for free reservation stations, if we find one, we modify the cycle it enters issue stage to be the current cycle, free it from IFQ, check its dependency on map_table and write its target register to map_table. Note that we the order for checking dependency and write target to map_table matters.

Yi Liu (999736889)
Daiqing Li (1000795357)

issue_To_execute:
In this stage, if an instruction does not have any dependencies and at least one of the functional unit of that instruction type is available, it is issued to a functional unit for execution. In order to preserve the priority of aged instruction when competing for functional unit, a queue is constructed to store ready-to-execute instructions in index order. All of the occupied reservation stations are scanned and checked for data dependencies. If a instruction is ready to execute (no RAW hazards), it is added to the ready queue. After that, the ready queue is sorted based on instruction index (order in the sequence) and the instructions are issued to any available functional units in this ascending sorted order.

execute_To_CDB:
In this stage, the following must be checked for:
(1) if a instruction currently in the functional unit has executed enough cycles. The latency of integer functional units is 4 cycles while the latency of floating point functional units is 9 cycles. In every cycle, the instructions that are currently occupying the functional units are checked to determine if they have completed execution.
(2) If multiple instructions complete execution during a single cycle then the oldest instruction has the priority of writing back to the CDB. This causes a structural hazard for CDB. Consequently it also must be ensured that reservation stations and functional units do not get deallocated until an instruction is ready to enter the CDB
(3) Since store instructions do not write to CDB thus they will simply be removed from functional unit and its reservation station deallocated.

CDB_To_retire: In this stage, the following must be done and checked for:
(1) On writeback through CBD, those reservation stations for instructions that stall due to RAW hazards must be notified to inform that the RAW hazard to the one of the source register has been resolved and they may be able to execute if no other hazards exist.
(2) After the writeback is broadcasted (by checking the source registers of reservation station entries), the map table must be checked for matches. A match would indicate that only this instruction has renamed the register and as this instruction retires, the resultant value will be written to cache, making the corresponding map table entry invalid . Otherwise if there is no matching entry, it means a subsequent instruction has overwritten that register and the mapping entry should not be removed.

Yi Liu (999736889)
Daiqing Li (1000795357)

Verification of correctness:

The following methods are employed to verify the correctness of the simulator code:

1. Running the SimpleScalar simulator with a small sample of the benchmark EIO trace using the -max:inst option of the sim-safe program while inserting print instruction macro function calls at appropriate lines to verify the instruction flow at various stages.
2. Inserting breakpoints in the functions to verify the program state and data structures used by the simulator
3. Manually perform Tomasulo algorithm simulation of the instruction sequence in the benchmark on paper. After that examine and compare with the outputs of print_all_instr() function which prints the outputs of the detailed cycle-breakdown of all the trace instructions.

Two toughest bugs:

1. The first one is the order of checking instruction dependencies and write target register to map_table in dispatch_To_issue stage. If the map_table entry for the target register is set first prior to the source registers, those instructions that both read from and write to the same register (such as addi r1, r1, 1) will have a self dependency loop that cannot be resolved.
2. The second one is when broadcasting data using CDB, all occupied reservation stations needs to be broadcasted regardless of the type (integer or floating point) of the CDB instruction. Some instructions depend on other type of instruction such as mtc1 which move from integer to floating point register file.

Work Completed:

Daiqing Li:

- Wrote the tomasulo.c simulator program code
- Edited and proofread the report

Yi Liu:

- Check/verify the logic and made corrections on the tomasulo.c program code
- Wrote the report