ECE552 - Lab 3 Report Pushkar Bettadpur | 999774971
Dynamic Scheduling with Tomasulo Tirthak Patel | 999678969

Total Number of Cycles with Tomasulo (First 1 Million Instructions)

	gcc.eio	go.eio	compress.eio
sim_num_tom_cycles	1729520	1722445	1821637

Description of Code for Each Stage

1. *fetch_To_dispatch*
   - Check if the Instruction Fetch Queue is full or if the required number of instructions have already been fetched. In either case, return.
   - If not, get the instruction from the Instruction Trace and increment *fetch_index.*
   - Check if the Instruction is a TRAP Instruction, if so, do not place it on the Fetch Queue.
   - If not, place the Instruction on the Fetch Queue, set *tom_dispatch_cycle* for the instruction to the current cycle and increment *instr_queue_size*.

2. *dispatch_To_issue*
   - Check if the Fetch Queue is empty; if so, return, if not, grab the first instruction (*head*).
   - Check if the *head* uses an Integer FU, if so, it requires an Integer RS.
   - Loop Through the Integer RS' and if one is available, place *head* onto that RS.
   - For each Input Register of *head*, compare its corresponding value on the Map Table and place the value found in the respective *Q* index of *head*.
   - Additionally, for each output register, replace the corresponding Map Table entry with *head*.
   - Remove the Instruction off the fetch queue and decrement *instr_queue_size.*
   - Similarly, perform the same steps for a FP RS.
   - Lastly, if the instruction uses neither the FP, nor the INT FU's, take it off the Fetch Queue and return.

3. *issue_To_execute*
   - Loop through all the functional units and check if any is available.
   - For each available FU, we loop through the RS and find the oldest Instruction with no dependencies (*Q's* are all NULL) and place it on the FU and set its *tom_execute_cycle*

4. *execute_To_CDB*

- Start by looping through all the INT functional units. If the unit is empty, continue.
- Otherwise, check is the instruction has finished by comparing its *tom_execute_cycle* to the *current_cycle*.
- If it is finished, check if the instruction writes to the CDB by checking the WRITES_CDB macro. If it does write, check if the instruction is the oldest finished instruction in the functional units by comparing its index against the indices of other instructions..
- Repeat the above three steps for FP functional units and get the oldest overall finished instruction.
- Put this instruction on the *commonDataBus*, set its *tom_cdb_cycle* to the *current_cycle* and set the corresponding function unit to NULL.

5. *CDB_To_retire*
   - Check if the *commonDataBus* is NULL. If it is occupied, move on to the next step.
   - Loop through all the INT reservation stations and check if any instruction is dependent on the instruction on the *commonDataBus*. If it is, set the corresponding Q value to NULL.
   - Repeat the above step for all the FP reservation stations.
   - Look for the *commonDataBus* instruction in the map table and once found, set the map table entry to NULL.
   - Lastly, set the commonDataBus to NULL.

Testing the Correctness of the Code

We tested the correctness of our code by comparing our design against an empirically measured set of values. To do so, we limited our maximum instructions for a benchmark to a small value such as 10 and stepped through (using gdb) and evaluated the performance of our code, cycle by cycle, against the theoretically expected values for Tomasulo

Two Toughest Bugs

1. When we first compiled the code, we saw that the simulation was running for infinite amount of time. We used the GDB debugger to go through the code line by line and check values of all the major variables, We found out that the head of the IFQ was not getting picked up for execution during the *dispatch_To_issue* stage. This was because we were checking if the instruction was unconditional branch, conditional branch, used INT FU or used FP FU. However, the first instruction had the op code flag set to OP_NA. Since, we were not checking this option, the IFQ was not emptying, the instructions were not moving forward and thus, the while loop ran forever. Therefore, we changed the conditions to checking if the instruction uses INT FU, else if uses FP FU or else neither of those.

2. A relatively tedious bug we encountered was in the termination of our while loop. In our *is_simulation_done* function, we were initially comparing the value of *sim_num_insn* with the size of the trace. This meant that our code was terminating much earlier than we expected it to. We eventually realized our mistake and corrected the function to compare *fetch_index* with *sim_num_insn* and terminated the loop when the former value exceeded the latter. However, the values still seemed incorrect and upon further analysis, we realized that even though *fetch_index* may exceed *sum_num_insn*, there may still be some instructions in other stages of the pipeline. Therefore, we had to add an additional check to see if all the other stages were clear and only then, do we terminate the loop.

Work Distribution

Tirthak and Pushkar worked together on all the components of the lab.