Lab3 Report

Total number of cycles with Tomasulo

Assume that the total number of cycles is the cycle after the CDB resource is freed.

Microbenchmark	Total number of cycles
gcc.eio	1681444
go.eio	1695065
compress.eio	1851551

Algorithm Overview

This section will provide a high-level overview of the Tomasulo's algorithm and any special cases that need to be considered. Tomasulo's algorithm is implemented by five different functions: fetch_to_dispatch, dispatch_to_issue, issue_to_execute, execute_to_CDB, CDB_to_retire.

In fetch_to_dispatch stage, the processor fetch and dispatch the instruction at the same cycle. The processor fills the instruction fetch queue with a new instruction. However, there are a few exceptions. If the fetch index is greater than the total number of simulation instruction (meaning there are no more to fetch), the fetch stage does nothing. If a trap instruction is encountered, the fetch index is incremented until the next non-trap instruction is encountered.

In the dispatch_to_issue stage, the processor moves an instruction from the instruction fetch queue to the reservation station. There are three different cases that need to be handled. If the instruction is a branch instruction, the processor does nothing; If the instruction is a floating-point computation, the processor removes the instruction from the instruction fetch queue and allocates the instruction in reservFP if there is an empty slot. The processor also updates the source registers' tag in the current instruction and disallowing it to proceed to issue stage if there are any dependencies. The destination registers' tags are updated in the map table meaning the destination registers will be written by this instruction; If the instruction is a integer computation, similar steps are produced, the only difference is that the processor looks for an entry in reservINT.

In the issue_to_execute stage, the processor iterates through the reservation stations and checks whether there are any instructions ready to be executed (meaning RAW hazards are resolved). The processor then checks if there are any functional unit available. If the number functional units are less than the number of ready instructions, older instructions take priority. The steps need to be done for both FP and INT computations.

In the execute_to_CDB stage, the processor iterates through the floating-point and integer functional units and checks if there are any instructions ready to be written back. If there are more than one instruction ready, the oldest instruction take priority. The oldest ready instruction free the reservation station and the functional unit. One thing to note is that store instructions do not require

CDB, so it will free its reservation station and functional unit immediately after it finishes execution.

In the CDB_to_retire stage, the instruction broadcasts the tags through CDB. It updates map table entries and reservation entries if there is a match. It will release the CDB source at the end of the cycle.

The simulation is done if the fetch index is greater than or equal to the total number of instructions and the instruction fetch queue, reservation stations, functional units and CDB are empty.

Helper Functions

There are two helper functions written for instruction fetch queue manipulation: ifq_insert and ifq_delete. Ifq_insert inserts a new instruction into the instruction fetch queue and increment the tail pointer if the queue is not empty. Ifq_delete removes an instruction from the queue and increment the head pointer if the head is not equal to the tail.

Testing

The first thing we verified is whether the delay for each stage is correct. So, we started with one instruction and see if the start cycle for different stages match with what we expect. We paid close attention to the delays of CDB and functional units.

The next thing we checked is whether the dependent instructions are executed correctly. When we moved onto 10 instructions, we observed that there were dependencies within instructions. So, we checked whether the dependent instructions wait for the previous instructions to broadcast their tags before it enters execution.

Next, we wanted to verify if the instruction fetch queue is implemented correctly, so we tested the algorithm for around 50 instructions. At this point, we observed that the instruction fetch queue started to stall new instructions. And a new instruction is only fetched when the instruction fetch queue has an empty spot.

After that, we looked for cases where there is more than one instruction ready to writeback and check if the older instruction has the priority. We also checked the same thing for functional units, meaning if there are more ready instructions than the number of available functional units, the older instructions take priority.

One thing we noticed is that the first two microbenchmarks do not have any floating-point computations. So, we repeated the steps for the third microbenchmark. We also checked the

following corner cases: store instructions do not write back, branch instructions do not occupy any reservation station or functional units, TRAP instructions are not fetched by the processor.

Two toughest bugs

The first toughest bug we had is that we were using functional unit index to index into the reservation station. It was hard to debug because the number of reservation stations is greater than the number of functional units. And it does not cause a segmentation fault and we used letter i and j to indicate the indices and they looked similar. So, it took us a while to figure out the issue.

The second toughest bug we had is we did not free up the resources for store instructions after it finishes execution. We only freed up resources for the instructions that require CDB. When we tested it, the simulation did not finish because the resources used by store instructions are never released. So, we used the debugger and let it run for a while and figured that the resources are all occupied by store instructions. That was how we figured out the root cause of it.

Work distribution

Tianyi Xu: implemented Tomasulo's algorithm, wrote report

Ternal Wang: Debugging