Lab03

Created by	Noviorlu Chan
Created time	@November 2, 2023 3:21 PM

Name: Jiping Li, Yang Chen

Total #of cycles EIO

Benchmark	gcc.eio	go.eio	compress.gio
Tomasulo Cycles	1681443	1695064	1851550

Tomasulo Stage Algorithmic Explanation

Fetch to Dispatch + Fetch

Retrieve an instruction from the instruction trace (skipping it if the instruction is a Trap or is empty) and push this instruction into the Instruction Fetch Queue. Then, set the instruction’s tom_dispatch_cycle to the current cycle.

Dispatch to Issue

Retrieve a new instruction from the Instruction Fetch Queue (IFQ)(skipping it if the instruction is a Branch or is empty.) Check for an available reservation station(if none is free, initiate a stall). Once a reservation station is free, populate its entry with the new instruction, and then update the entry's tags with the corresponding tags from the Map Table based on the input registers. Adjust the Map Table in response to the new instruction's output register (at this stage, potential Read After Write (RAW) hazards are identified). Lastly, set the instruction's tom_issue_cycle to the current cycle.

Issue to Execute

Check if the Functional Units are available (if not, skip this step). Then, select the oldest instruction that is ready (all input registers must be ready, resolve RAW hazards) from the corresponding Reservation Station and load it into the Functional Unit entry. Set the instruction’s tom_execute_cycle to the current cycle.

Execute to CDB

Verify whether any instructions in the Functional Units have been completed (if not, skip this step). In cases where multiple instructions have finished, broadcast them through the Common Data Bus (CDB) according to their index (skip this for 'store' instructions). After broadcasting, clear the completed instruction's entries in both the Reservation Station and Functional Unit. Lastly, set the broadcast instruction's tom_cdb_cycle to the current cycle.

CDB to Retire

Propagate the CDB result into the map table and reservation stations. If any entry requires the broadcasted instruction's result, clear that entry. Finally, clean the CDB.

Special Handling

As mentioned above, we purposely disallow some instructions into the next stage, based on the type of instruction.

Helper Function

int update_Fetch_index(instruction_trace_t trace_ptr)**

Since get_instr provided by instr.c is problematic because it can not update trace = trace→ next correctly, we wrote our own function to update trace pointer and fetch index.

instruction_t* instr_queue_pop()/void instr_queue_push(instruction_t* insertEntry)

To make instr_queue a proper queue by shifting elements when pop.

void cleanInstr(instruction_t* instrFinished)

Used in Issue to Execute. Clean the finished instruction's RS and FU entries

int returnRSfreeIndex (instruction_t * rs[], int size)

return a free entry index for RS, if RS full, return -1

Testing Correctness

For correctness testing, we have selected gcc.eio to examine and print out the stages of all instructions, analyzing the corner cases managed by the Tomasulo Algorithm to verify correct cycle updates. Starting from the Dispatch stage and continuing through to the Writeback stage, we have identified all corner cases that could lead to IFQ structural

hazards, RS structural hazards, FU structural hazards, CDB structural hazards, as well as RAW, WAR, and WAW data hazards. It is crucial to ensure that each case is handled properly by our algorithm.

Toughest Bug

report.pdf : a brief three-page pdf report (single-spaced with 12-point font size). Make sure your report can be viewed on the ug machines through xpdf or acroread.
Report the “total numbers of cycles with tomasulo” for the first one million instructions of each EIO trace.

1. **Segmentation fault:** During our development of the Tomlasulo algorithm, we encountered at least one Segmentation fault for each stage. This is related with our code development style, we forgot to check when accessing instr pointer’s is null or not before accessing it. This do cause us to use the gdb with bt command a lot.
2. **Trace Table Update:** as we are trying to update the fetch_index we encountered the fetch table access issue, we have to update the `trace` manually which cause us lots of trouble that we discussed the solution in the Special Handling section.

Work Statement

Work has been evenly divided between us, as we employ the peer coding method where one person writes the code and another person handles debugging.