ECE4504
Project Phase 1: Pipeline Mechanisms

Emil Bengtsson

October 2018

Implementation

The first task was to implement forwarding in the baseline EduMIPS64 simulator. But forwarding is already implemented in the simulator and can be activated by checking the checkbox under the "Main settings" tab in the settings. Since the forwarding functionality is already implemented there is no reason to implement it again.

The second task was to prevent simultaneous reading and writing of the register files, this was not already available in the baseline EduMIPS64 simulator and had to be implemented. I implemented it by checking whether any register had been written to in the writeback stage of the pipeline, and if it had the program blocks the corresponding register file from being accessed. If an instruction tries to access a register file that is not accessible a structural stall is introduced.

Experimental Setup

For evaluation I will use the three benchmarks provided to me as part of the assignment. All three benchmarks do some loading from memory and a lot of basic arithmetic with operations that depend on previous operations which will generate RAW ("read after write") stalls. This is good for testing the forwarding since it will presumably make the RAW stalls less frequent which will show up in the statistics of the simulator. It is also good for seeing what happens when simultaneous reading and writing of the register files is not allowed, since instructions that store data (many of the instructions carrying out basic arithmetic) should create more stalls in the pipeline (Because they prevent other instructions from reading data).

I will evaluate the implementations by running the benchmark programs, first with the baseline simulator, and then with the modified simulator. I will then compare the statistics of each run.

I will refer to the three benchmark programs I use by their names: minimiconcorrenti, raddrizzamentomediavoti, and ricopiavaloriinternirange.

Results

Results of the benchmarks running on the baseline simulator:

Benchmark	minimiconcorrenti	raddrizzamentomediavoti	ricopiavaloriinternirange
Cycles	360	424	243
Instructions	200	211	122
CPI	1.8	2.009	1.991
RAW Stalls	104	186	92
WAW Stalls	0	0	0
WAR Stalls	0	0	0
Structural Stalls (Register not available)	0	0	0
Structural Stalls (Divider not available)	0	0	0
Structural Stalls (Memory not available)	0	0	0
Branch Taken Stalls	0	0	0
Branch Misprediction Stalls	0	0	0

Results of the benchmarks running on the simulator with forwarding enabled:

Benchmark	minimiconcorrenti	raddrizzamentomediavoti	ricopiavaloriinternirange
Cycles	267	243	151
Instructions	200	211	122
CPI	1.335	1.151	1.237
RAW Stalls	11	5	0
WAW Stalls	0	0	0
WAR Stalls	0	0	0
Structural Stalls (Register not available)	0	0	0
Structural Stalls (Divider not available)	0	0	0
Structural Stalls (Memory not available)	0	0	0
Branch Taken Stalls	0	0	0
Branch Misprediction Stalls	0	0	0

Results of the benchmarks running on the simulator with forwarding disabled and simultaneous reading and writing of the register files disabled:

Benchmark	minimiconcorrenti	raddrizzamentomediavoti	ricopiavaloriinternirange
Cycles	452	544	302
Instructions	200	211	122
CPI	2.26	2.578	2.475
RAW Stalls	85	160	80
WAW Stalls	0	0	0
WAR Stalls	0	0	0
Structural Stalls (Register not available)	111	146	71
Structural Stalls (Divider not available)	0	0	0
Structural Stalls (Memory not available)	0	0	0
Branch Taken Stalls	0	0	0
Branch Misprediction Stalls	0	0	0

Comment on the results

The results from running the benchmarks on the different configurations of the program show that enabling forwarding can reduce the amount of RAW stalls drastically. This leads to quite a big reduction in the CPI of the program and also makes the program run faster. The speedup from forwarding can vary depending on the program though, and the speedup of programs other than the ones used as benchmarks might not be as big. This is because the benchmark programs have a lot of dependencies in the code which cause stalls when forwarding is not enabled.

The results also show that disabling simultaneous reading and writing of the register files significantly increases the number of stalls for the benchmarks, because it introduces a lot of structural stalls. In the results the amount of RAW stalls is reduced in comparison with the baseline simulator without forwarding enabled, this is because whenever there is both a structural stall and a RAW stall the program counts it as a structural stall. Because of the increase in structural stalls, the CPI of the benchmark programs are increased by quite a lot.

One interesting thing to note is that for all the instances of the simulator and all the benchmark programs, the number of instructions and stalls together don't match up to the number of cycles. I think this is because the simulator always predicts that branches will not be taken, and then when they are taken there is a stall. The simulator does not, however, keep track of these stalls or print them in the "statistics" window.