Project Progress report: Pipeline Mechanics

Chris Morin and Dirk Dubois

Table of Contents

Abstract. To progress with the evaluation of the MIPS64 Pipeline mechanics, the eduMIPS64 environment was examined. EduMIPS64 uses a simple forwarding mechanism which while elegant in its simplicity, doesn't allow for customisation. EduMIPS64 is being modified to allow more control over which forwarding paths can be taken. Also, a set of test bench assembly files were created that generate read after write dependencies that can be resolved with forwarding.

1. Work Completed

1.1. The MIPS64 EDU environment

The MIPS64 EDU environment is a platform of choice to examine the various functional differences of the MIPS64 pipeline. This java base simulator provides a set of tools to examine the pipeline, where instructions are stalled, the contents of the registers, and the overall structure of memory. Using all of these tools we are able to evaluate the performance of our pipeline as we make changes to its structure.

1.2. Implementing forwarding

The eduMIPS64 simulator code was examined and the way it implements forwarding was studied. It was found that eduMIPS64 uses a simply but inflexible way of managing forwarding: it simply executes the write-back stage of an instruction as soon as the data is available. The simulator uses semaphores associated to the registers to manage if a register has been needs to be written to. The write-back stage of these forwarded instructions also decrements these semaphores to signal their availability.

A plan to implement more customisable forwarding was devised. Instead of there being one flag signaling that forwarding has been enabled, there will be two: one signaling that forwarding is enabled at the EX stage and the other signaling it's available at the MEM stage. This would provide us with a total of four forwarding options.

- no forwarding

- forwarding at the EX stage only
- forwarding at the MEM stage only
- forwarding at both the EX and the MEM stage

1.3. Creating a Test Bench

To determine if the different forwarding schemes applied to the standards MIPS64 pipeline offer any sort of improvement, a set of standard test cases would be required.

In order to test if the pipeline will stall on a read after write (RAW), the test bench should be able to generate this type of error. After implementing forwarding, these test cases can be run again, and the number of stall cycles compared for each type of pipeline.

Two RAW data hazard test benches were written. The first creates a true data dependency between two registers that can be solved by forwarding to the EX stage. The second creates a dependency between two registers needed for a load and store operation, which can be solved by forwarding from the EX stage to the MEM stage and from the MEM stage to the EX stage. For the second test case, since memory is being used, there will still be at best one RAW stall.

The following code snippet shows a portion of data hazard test bench 1:

```
;Do Useful work
;Without forwarding 2 RAW stalls expected due to R1 dependency
;Adding forwarding from EX to back to EX will solve this stall

DADD R1, R2, R3 ;Put something in R1
DADD R4, R1, R5 ;Dependency on R1

DADDI R3, R3, 0 ;Execute filler instruction
SYSCALL 0 ;Exit program
```

2. Work Remaining

We still have to implement the additional forwarding functionality. We' also have to write more test cases and possibly find some existing MIPS64 assembly code to test. We'll create an automation script which collects the data from the various simulator runs and sees how the performance changes with the instruction type frequencies and the forwarding options.

3. Issues Encountered

At this point we haven't encountered many issues as we haven't actually done much simulating. The eduMIPS64 simulator code base was quite extensive and finding how to modify the forwarding paths without spending days searching through source code was a challenge and required us to generate the JavaDoc from the source files and work through a CPU "step" in our heads.