

Lab 2: Pipelined Muldiv Unit

Michael Tu

Due: 2/22/2023

1 Abstract

The purpose of this lab was to implement a pipelined, bypass enabled processor that can execute the PARC2 subset of x86-instructions. This provides enough functionality to run basic C code which does not require systemcalls. In particular, the processor also includes a pipelined multiplication/division unit to increase performance.

2 Design Considerations:

Section 1: For section 1, the main design change I needed was to add additional wires to capture additional branch conditions besides `bne`.

Section 2: For section 2, I modified my control unit to send bypass signals (`r[x]_byp_[X]h1`) to my datapath. The only cases in which one would need bypass values is when reading from the register file. Therefore, I added a mux input that combined the register files and bypass values. Finally, I added stalling for load-use dependencies, where a instruction must stall if it depends on a load.

Section 3: For section 3, I added two additional stages into the pipeline. Additionally, I needed to pass the `val` and `rdy` through the control units. The first additional stage “Execute 2” does not perform any computation. Instead, it is only used to stall until the pipelined MulDiv unit completes. The second additional stage, “Execute 3” is used to receive the output of the pipelined muldiv unit and pass it to the writeback stage.

3 Testing

In order to test, I tried repeatedly loading and reading from the same memory location.

4 Evaluation

The benchmark results are below

pv2Stall Results			
Benchmark	Cycles	Instructions	IPC
vvadd	478	455	0.951883
cmplx-mult	16718	1864	0.111497
masked-filter	16174	4499	0.278162
bin-search	3382	1279	0.378179


pv2Byp Results			
Benchmark	Cycles	Instructions	IPC
vvadd	473	455	0.961945
cmplx-mult	15312	1864	0.121735
masked-filter	13832	4499	0.325260
bin-search	1749	1279	0.731275

pv2long Results			
Benchmark	Cycles	Instructions	IPC
vvadd	473	455	0.961945
cmplx-mult	2662	1864	0.700225
masked-filter	6058	4499	0.325260
bin-search	1749	1279	0.731275


5 Discussion

From the benchmarks provided, we can see that bypassing provides a significant performance improvement over stalling, up to an almost $2\times$ improvement. However, we see that even greater performance improvements can come by preventing the pipeline from stalling for long multiplications. This leads to a nearly order of magnitude improvement, especially for the complex multiplication test which involves many multiplications.

6 Figures

 dpath2.png

This is the datapath for part 2, the important difference is the addition of two muxes which allow for bypass values into the op0 and op1 muxes.

 dpath3.png

This is the datapath for part 2, the important difference is the addition of two muxes which allow for bypass values into the op0 and op1 muxes.