Midterm Report

ECE 437

Team: Mitchel Bouma, Alex Dunker

TA: Chuan Tan

Due Date: December 9, 2016

Overview

In the past seven weeks, we successfully designed and implemented a MIPS single cycle and pipelined processor. There are specific benefits to each design. The single cycle processor is much easier to understand and create. Because everything is completed in one clock cycle, there is no need to worry about timing issues or hazards. In addition, there is less combinational logic, so the power consumption is lower. The pipelined version is more in-depth and therefore harder to understand. It requires more power for combinational logic in order to prevent hazards since multiple instructions are being processed at the same time. The benefit to this design is the fact that you can split the process into multiple stages and therefore have multiple instructions in the pipeline at once. This way, rather than waiting for one instruction to finish completely, we can be using every resource in the processor constantly. The maximum attainable clock frequency is much higher for a pipelined processor.

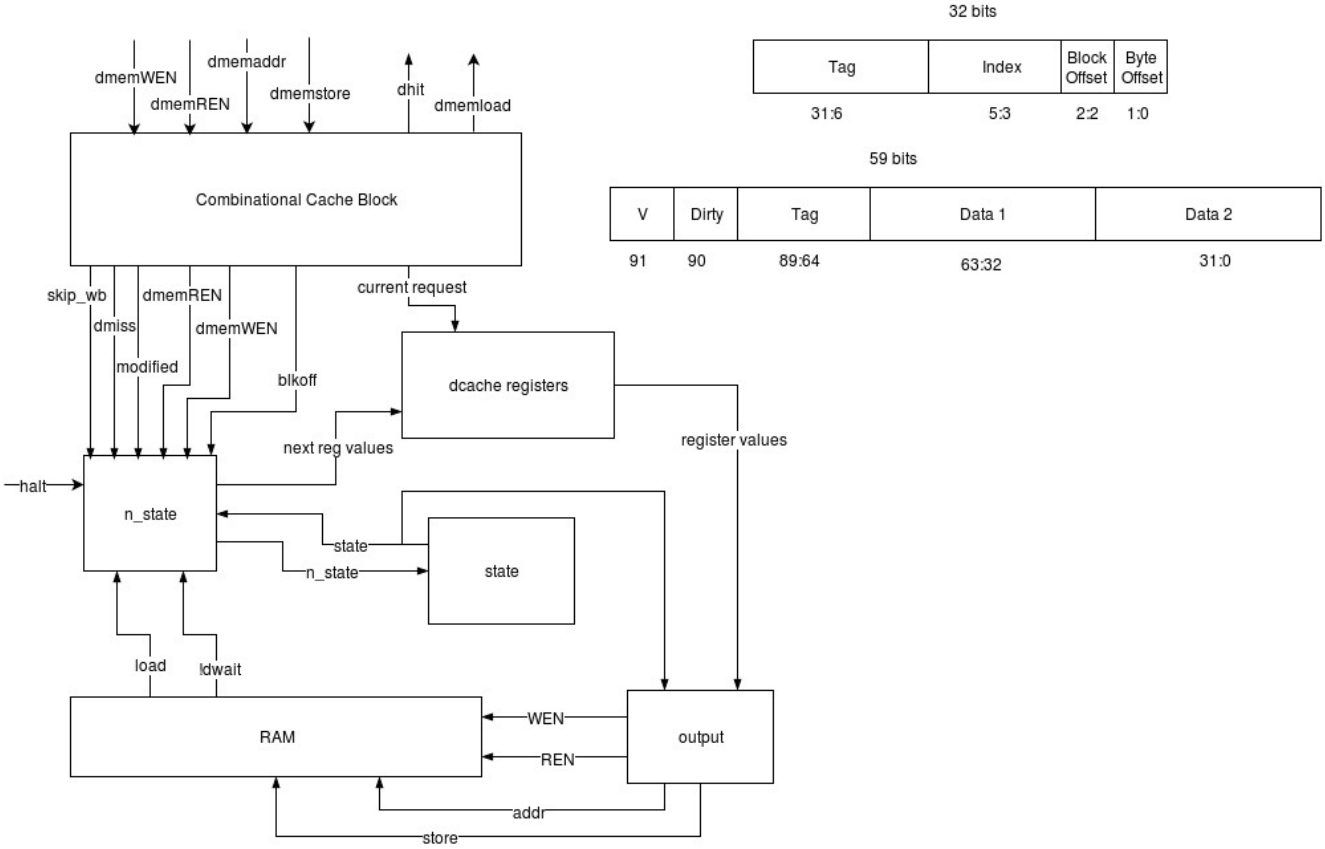
The test bench used to get metrics was mergesort.asm and dual.mergesort.asm. Mergesort has many branches, r-type instructions, and jumps, so it therefore tests the main functionality of the processor well. It is also very complex and has a relatively long execution time, so the metrics will be distinguishable between the designs. The main metrics used when comparing the designs are max frequency, mips (millions of instructions per second), and run time.

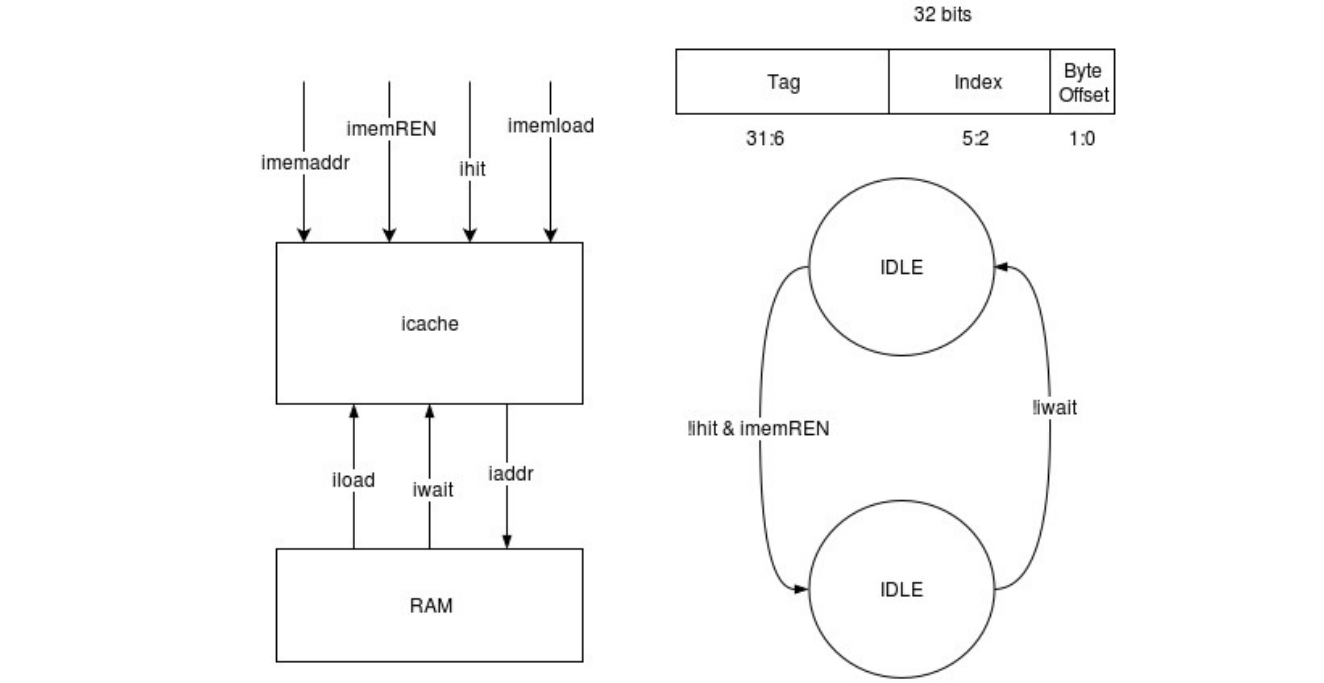
Surprisingly, in our case, the single cycle version seemed to perform better. Although the maximum frequency was lower, it was able to perform many more mips, and it completed in a degree of magnitude less time. We believe this may be due to the fact that there were hazards which require flushing, and a branch predictor may greatly improve the pipeline's performance.

Processor Design

Processor Design

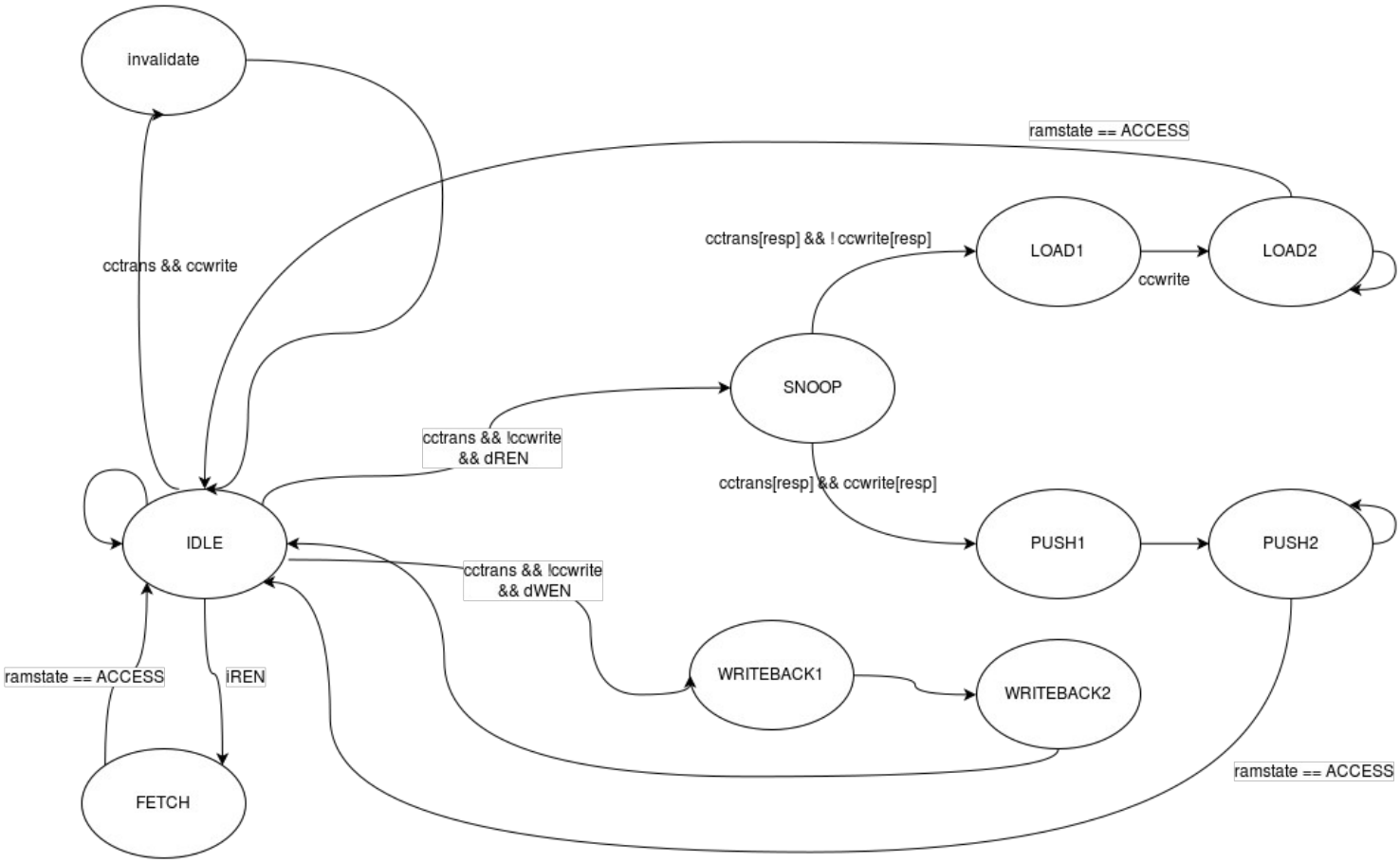
Cache Design





Coherence Control

Memory Control State Diagram



Results

Test Case: Mergesort.asm

	Single Cycle	Pipeline	
Max Clock Frequency (logs)	39.26 MHz	58.67 MHz	(system.log)
Max Clock Frequency (tb)	50.0 MHz	76.92 MHz	1/period in tb
Avg Instructions/Cycle	0.39	0.079	(instructions/cycles)
Critical Path	24.94 ns	16.96 ns	(system.log)
Latency	25.47 ns	85.22 ns	(period or period*5)
Mips (mergesort.asm)	19.5	3.95	((I/C * C/S) / 10^6)
FPGA Resources	1315	1629	(system.log)
Run Time	275,860 ns	1,366,440 ns	(make system.sim)
Cycles	13792	68321	(make system.sim)
Instructions	5399	5399	(sim command)

Conclusion/Contributions

It is quite surprising that our single cycle processor seems to be performing more efficiently than the pipelined version. As stated above, this could simply be due to the fact that branch prediction is not yet implemented, and there could be many flushes happening which basically waste clock cycles. It is important to note, however, that the max frequency of the pipelined processor is higher than that of the single cycle which proves that splitting up instructions into multiple processes is beneficial. Finally, the registers is higher in pipeline, but that is expected since we had to create extra registers for each stage of the process.

Mitch can be credited with figuring out the base comparisons to be made in order to check for hazards. We also used his single cycle processor as a starting point since it performed fairly well in comparison to the rest of the class. In addition, he created the block diagrams and the test bench in order to prove that the stall signal gets asserted when a hazard is found that can't be solved by forwarding.

Alex was instrumental in finding the edge cases in terms of hazards. We didn't quite account for all the possibilities the first time through, but Alex was able to go back and implement a few more comparisons/control signals. In addition, he was responsible for splitting the individual stages and creating the flow between the fetch, decode, execute, and memory registers. This included the individual register files as well as many modifications to the datapath.