ECE 437L Midterm Report

Everett Berry

Sheik Dawood Beer Mohideen

Adam Hendrickson

March 06, 2015

Executive Overview

In this document we provide a brief, technical comparison between single cycle and pipelined processor implementations of the MIPS instruction set. We consider several criteria to measure the performance of the two processors, including the average instructions per clock cycle (throughput) and the FPGA resources (size) needed for the processor.

In order to make an adequate comparison, it is important to note the characteristics of each implementation. The single cycle design is conceptually simpler than the pipelined design and involves a single instruction executing at any given time during one long clock cycle. This architecture is also the greatest downside of the single cycle processor and the pipelined processor seeks to rectify this by executing five instructions simultaneously, thus greatly improving throughput and general performance.

The test program used to collect data on the two designs was a MIPS implementation of the merge sort algorithm. Because it involves a large number of branches and jumps, as well as frequent loads and stores, mergesort.asm covers a broad array of hazards and approximates the processor running for an arbitrary program. Merge sort's complexity and long run time also mean that we should be able to recover performance numbers for the average case and compare the two designs fairly.

After running merge sort through both programs and collecting data from log files and test script outputs, we concluded that the pipeline design does in fact perform significantly better than the single cycle design. Both the maximum frequency and the MIPS (millions of instructions per second) were twice as large as those same numbers for the single cycle design.

Processor Design

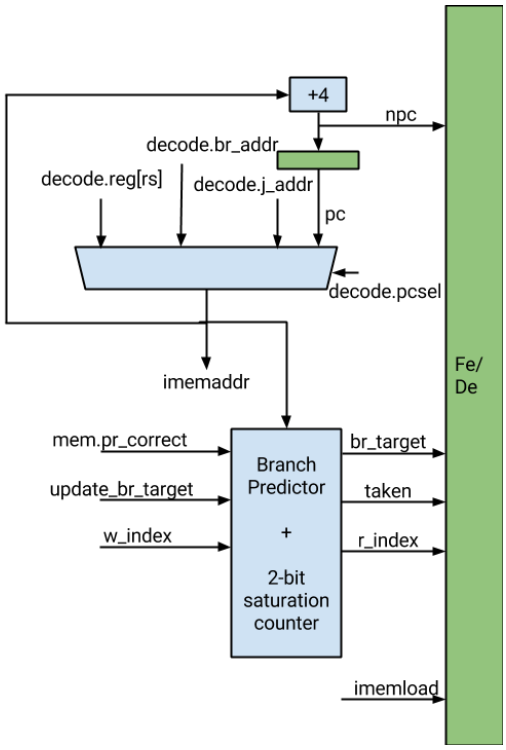

Figure 1: Fetch Stage

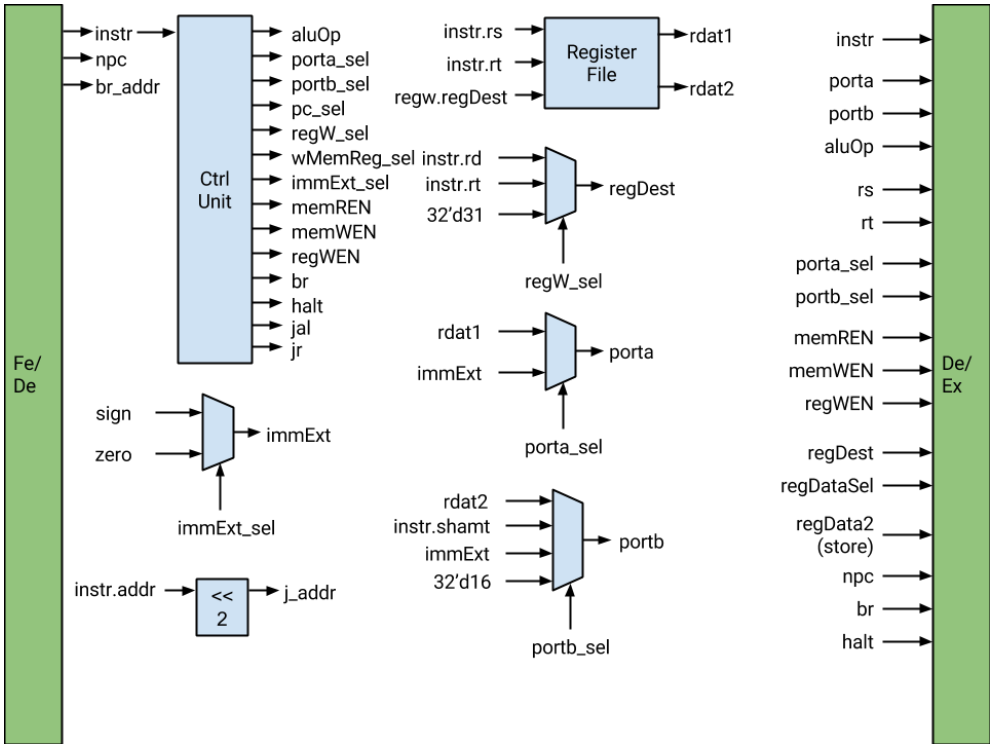

Figure 2: Decode Stage

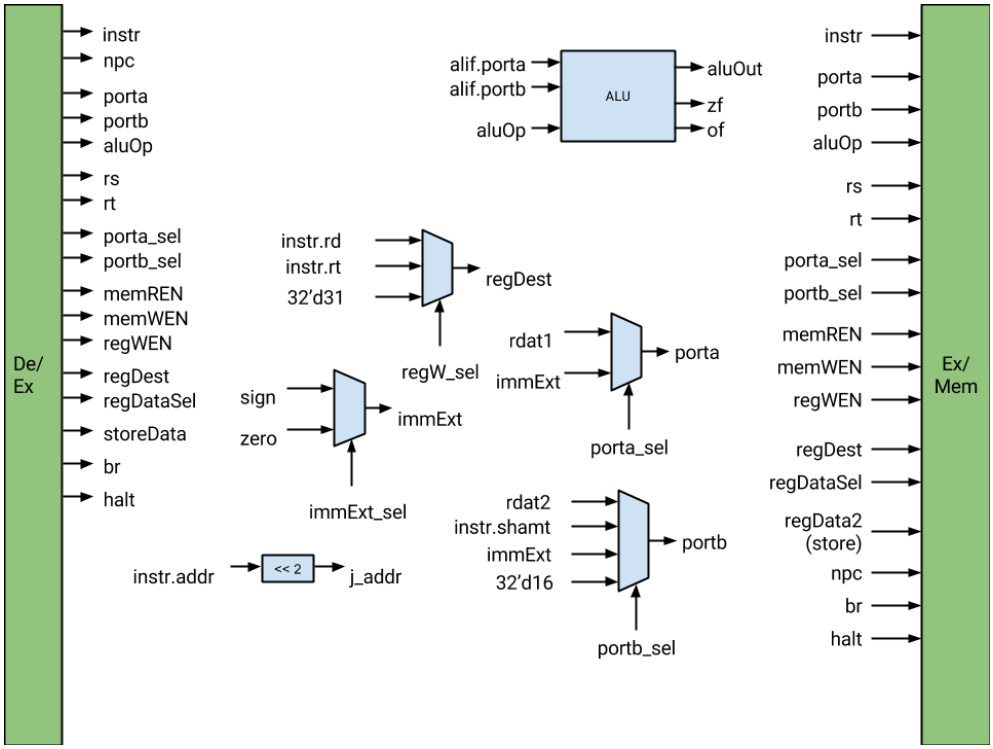

Figure 3: Execute Stage

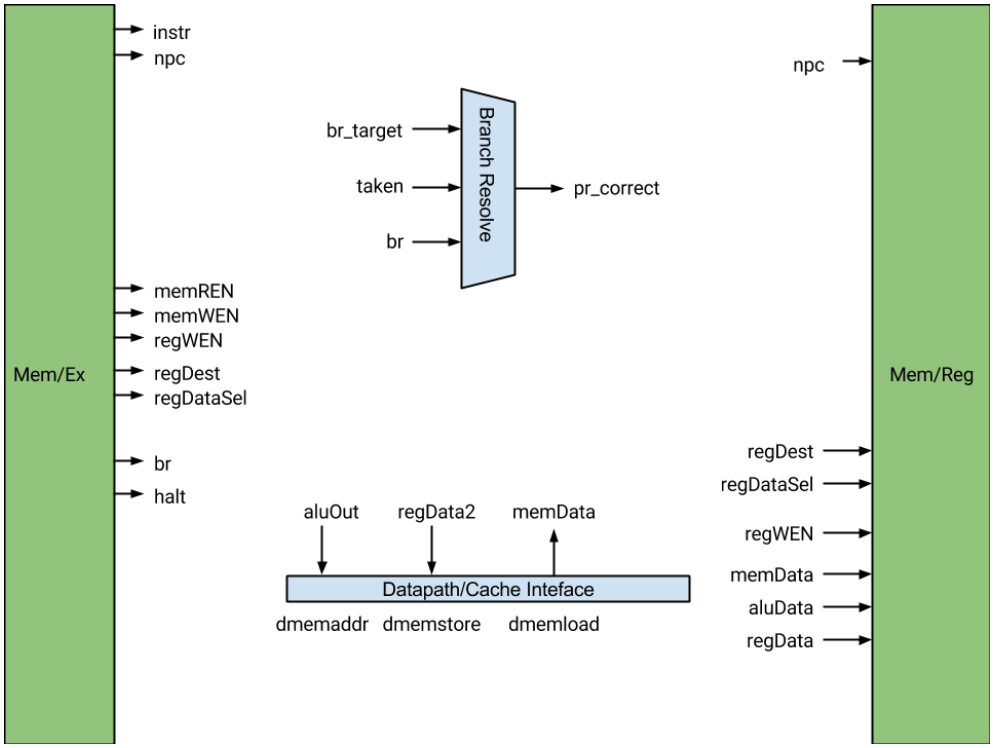

Figure 4: Memory Stage

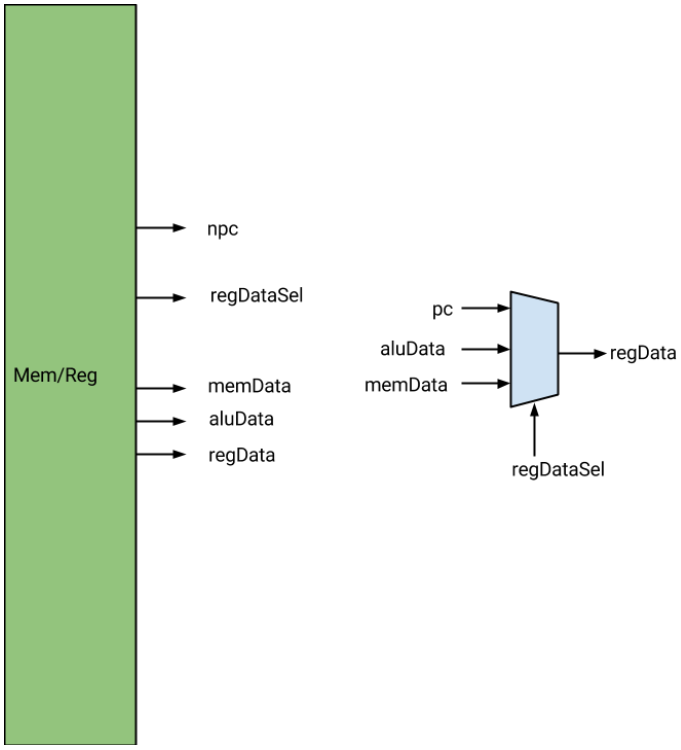

Figure 5: Write Back Stage

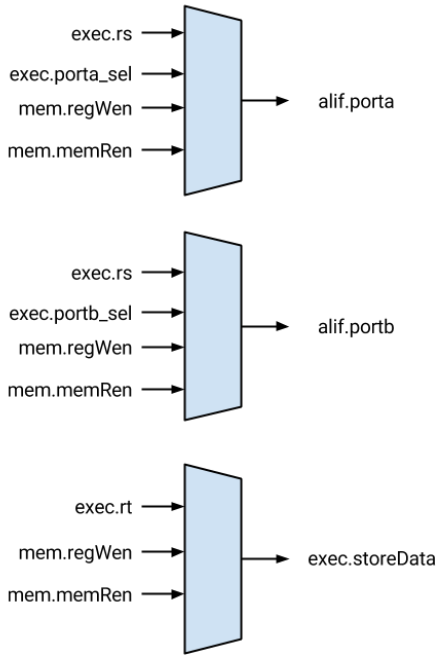

Figure 6: Forwarding Unit

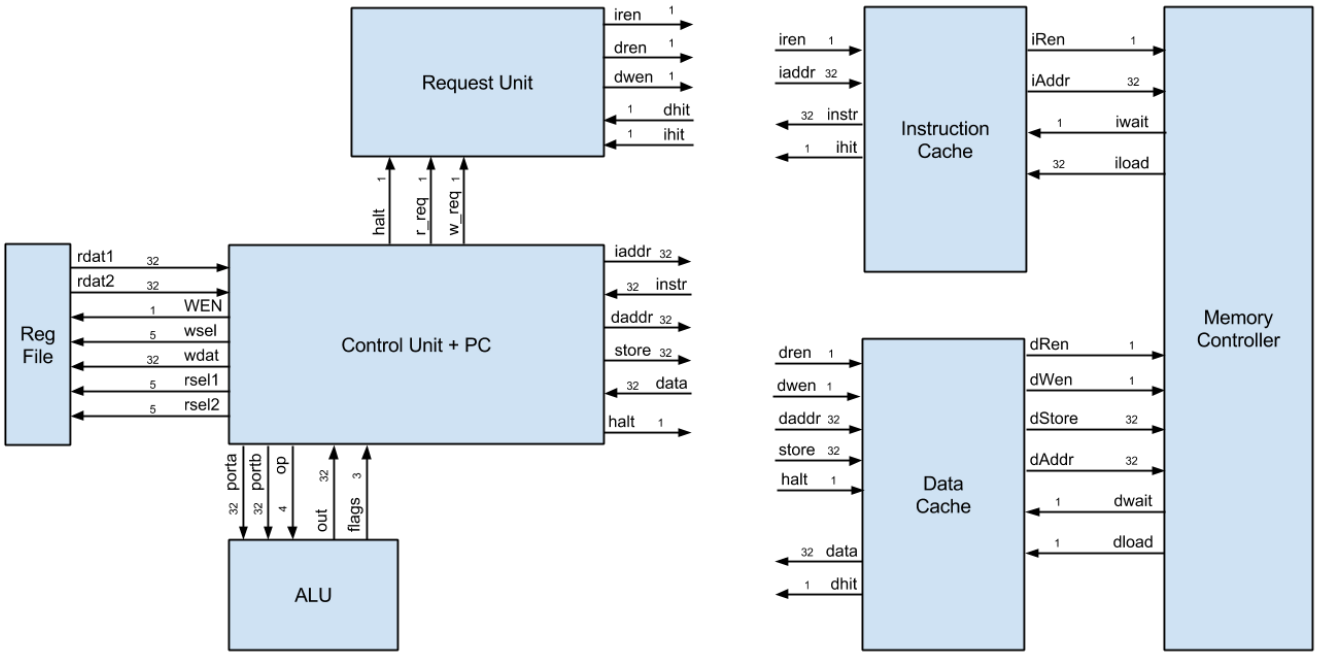

Figure 7: Singlecycle block diagram

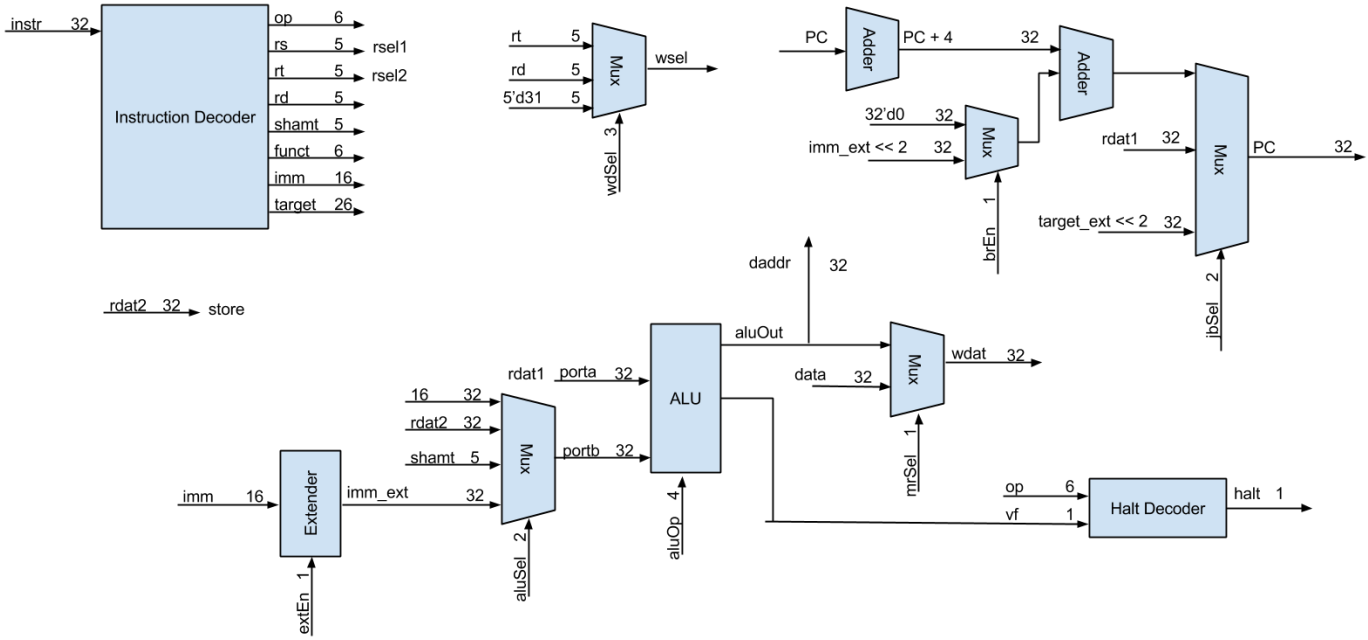

Figure 8: Singlecycle RTL diagram

The previous eight figures showcase the different parts of our design. The first six are the five stages of the pipeline and the forwarding unit and the last two are a block level and RTL level specification of the single cycle. The single cycle design closely matches the design that was presented in lectures and in the textbook (with the exception of separation of memory) but the pipeline implementation has one notable feature worthy of discussion.

This feature, shown in Figure 1, is branch prediction. A four entry branch target table and a two bit saturating counter let the pipeline processor choose a branch in the IF stage and resolve the prediction later in the MEM stage. This seemingly small optimization actually gives our processor a significant performance boost compared to our design without prediction. In fact, we believe this to be the key factor in making our processor the most performant in the class (overall latency and throughput).

Results

Measurement	Singlecycle	Pipeline
Frequencey (Mhz)	33.19	57.33
Throughput (instrs/cycles)	5399/6896=0.78	5399/3828=1.41
Latency (ns)	275840/5399=51	107212/5399=19
MIPS	0.024	0.050
FPGA Resources (LUs)	3,216/114,480 (3%)	3,742 / 114,480 (3%)

Table 1: Processor Specs

The results in Table 1 clearly show the obvious: the pipelined processor is better in almost every respect than the single cycle processor. The only category in which the pipelined design could be considered to be worse is in utilization of FPGA resources. Even this is not a concern however, because the additional resources are simply the flip flops necessary for pipeline latching, forwarding, and the branch prediction logic.

Conclusion

This report sought to compare the design and performance of the single cycle and pipeline processors. First, the motivation and critical differences in each were discussed. Then, block diagrams of each design which showed the dramatic differences in implementation were presented. Finally, the same program was executed on both processors and the results were compared.

In that comparison, we showed that the pipeline design was very clearly better than the single cycle. In every meaningful category, it essentially doubled the performance of the single cycle design. A more complex design is necessary to achieve these gains, but this is a worthy trade off to make in light of the stark difference in performance.