ECE 437L Midterm Report

Pooja Kale (mg273)

Brian Rieder (mg267)

TA: Nick Pfister

October 14, 2016

1 Executive Overview

This report compares the performance of a single cycle processor and a pipeline processor on a merge-sort assembly (asm) file, and analyzes the results. The comparisons of the two processor designs will be based on the maximum clock frequencies, average instructions per clock cycle, the performance of the designs in MIPS and the FPGA resources required for each design. The comparisons will be further tested using variable latency RAM in order to ensure independence from memory timing. The single cycle processor is a simple design that takes in one instruction at a time to decode it and produce a result. This design does not have any instruction dependencies eliminating the possibility of instruction or data hazards. The pipeline processor is a more intricate design with five stages: instruction fetch, instruction decode, execute, memory, and write-back. This allows the design to process five instructions simultaneously within different stages of their execution.

The two designs will be tested using the merge-sort assembly file. The benefit of using the mergersort assembly program, is it includes all possible assembly instructions that both processors can decode. The merge-sort file also includes multiple branches to test branch logic in both designs. The order of instructions in the program will test if the pipeline handles

data hazards, and structural hazards with inserting stalls or bubbles in the correct places and flushing data at the right time. The execution results show that the pipeline processor is the favorable design because it increases the throughput (or decreases the execution time) as compared to the single cycle. While the single cycle handles one instruction at a time the pipeline processor design processes 5 instructions in different stages in one cycle.

2 Processor Design

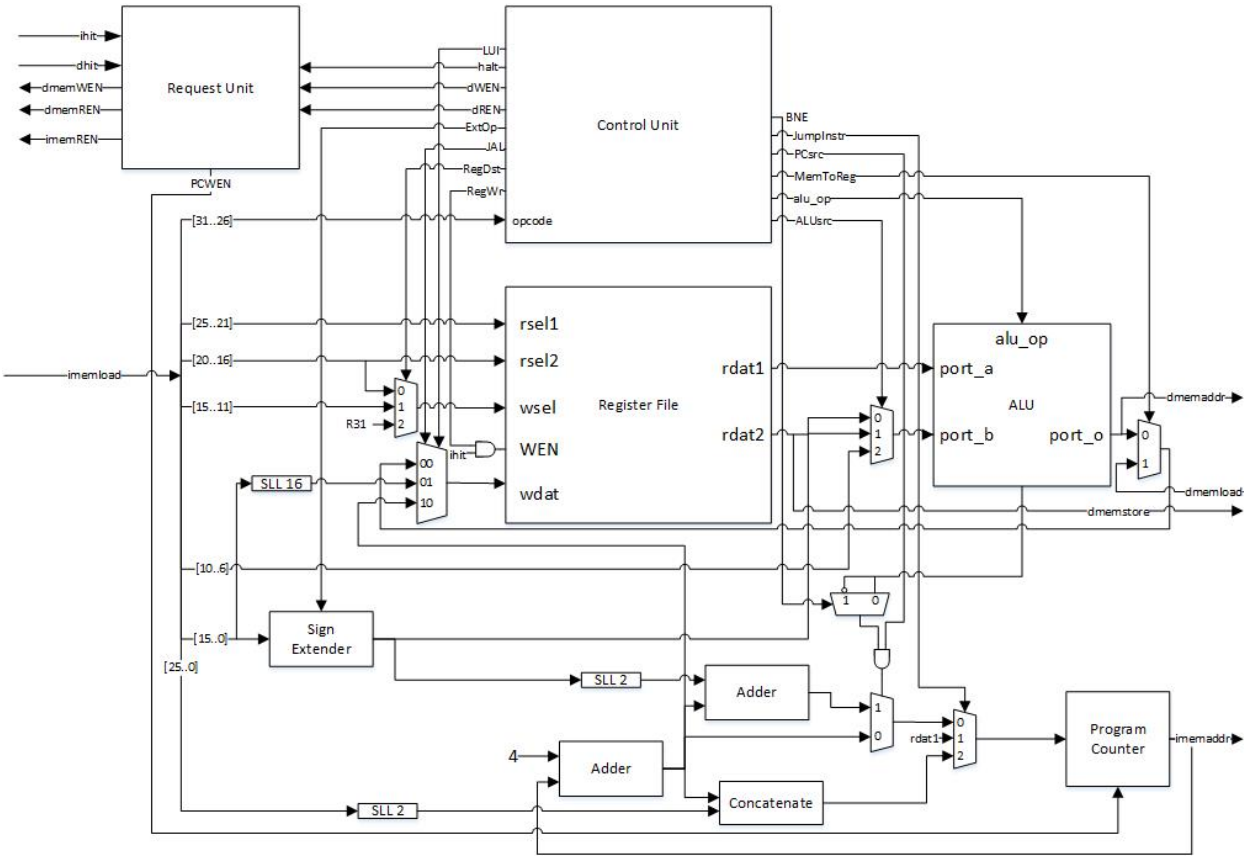
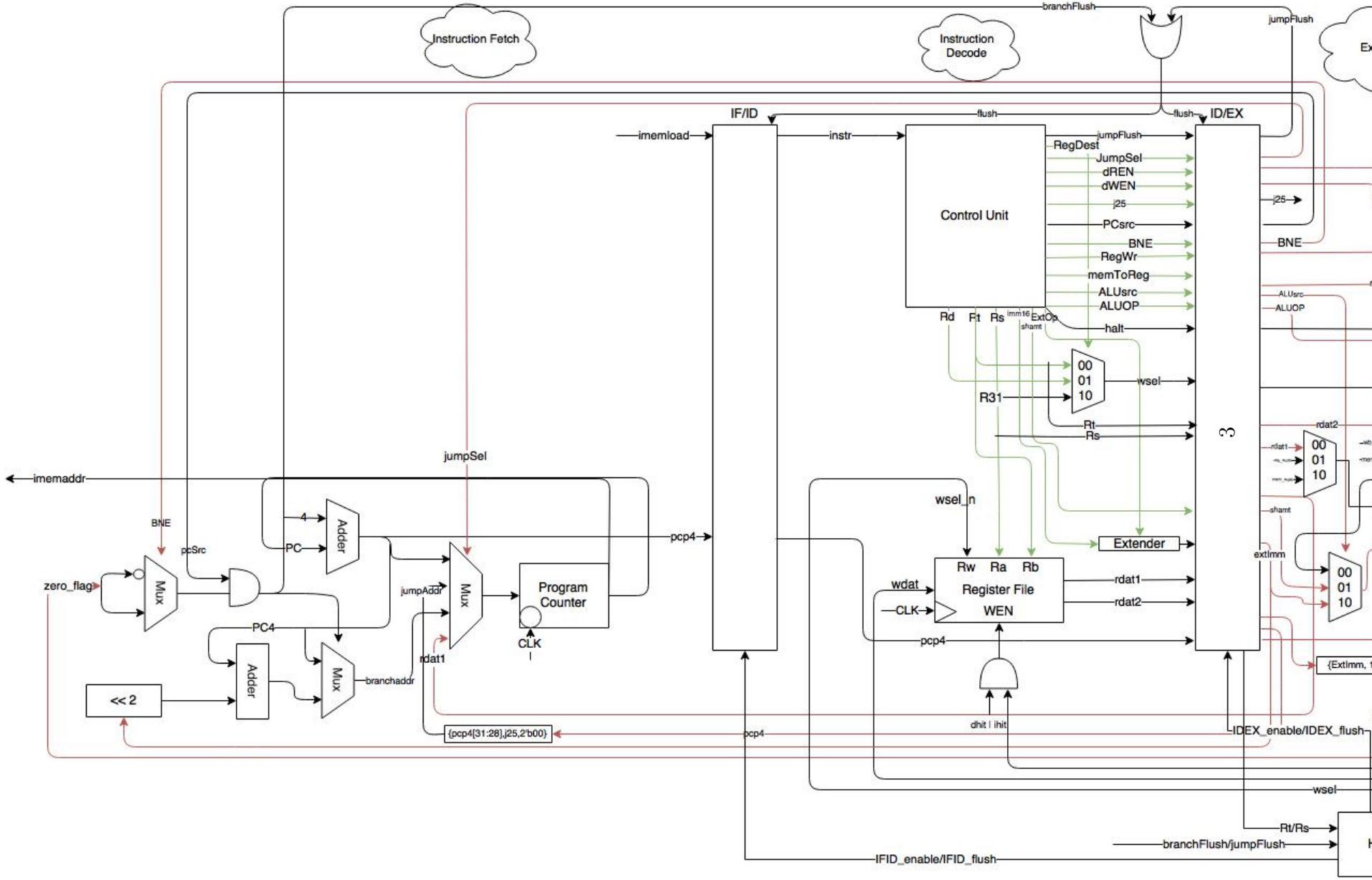


Figure 1: Single Cycle Block Diagram

Text for discussing your designs

Text for discussing your designs


Instruction Fetch
Instruction Decode
branchFlush
jumpFlush
IF/ID
flush
ID/EX
imemload
instr
Control Unit
RegDest
JumpSel
dREN
dWEN
j25
PCsrc
BNE
RegWr
memToReg
ALUsrc
ALUOP
halt
Rd
Rt
Rs
imm16
ExtOp
shamt
00
01
10
wsel
R31
Rt
Rs
jumpSel
imemaddr
BNE
pcSrc
4
PC
Adder
Mux
zero_flag
jumpAddr
Program Counter
CLK
PC4
<< 2
branchaddr
rdat1
{pcp4[31:28],j25,2'b00}
pcp4
wsel_n
Extender
Rw
Ra
Rb
Register File
WEN
wdat
rdat1
rdat2
dhit | ihit
IDEX_enable/IDEX_flush
IFID_enable/IFID_flush
Rt/Rs
branchFlush/jumpFlush
ALUsrc
ALUOP
shamt
extImm
{ExtImm,

3 Processor Debug

Fix the processor!!!

4 Results

Throughput	??/??=??instructions/ns
Latency	??*??=??ns/instruction

Table 1: Processor Specs

Text for your results

5 Conclusion

Text for your conclusion

6 Contributions

Text for individual contributions