|  |  |
| --- | --- |
|  | 16-bit Five Stage MIPS Processor |
|  |  |
|  | Jared Mann V00187636, Taylor Long V00862856  CENG 450: Computer Systems and Architecture  3/30/17 |

Contents

[I. Objective 2](#_Toc478475559)

[Abstract 2](#_Toc478475560)

[Project Objective 2](#_Toc478475561)

[II. Design Requirement 2](#_Toc478475562)

[General Design Requirements 2](#_Toc478475563)

[Instruction Set 2](#_Toc478475564)

[Registers and Memory 2](#_Toc478475565)

[Pipelining 2](#_Toc478475566)

[Program Execution 2](#_Toc478475567)

[General Implementation Choices 3](#_Toc478475568)

[Forwarding 3](#_Toc478475569)

[Branch Prediction 3](#_Toc478475570)

[III. System 3](#_Toc478475571)

[System Hierarchy Breakdown 3](#_Toc478475572)

[System Schematic Breakdown 5](#_Toc478475573)

[IV. Individual Module Design 5](#_Toc478475574)

[Top Level Processor Unit 5](#_Toc478475575)

[CPU 5](#_Toc478475576)

[ALU 5](#_Toc478475577)

[Muxes 5](#_Toc478475578)

[Program Counter 5](#_Toc478475579)

[Pipeline Registers 5](#_Toc478475580)

[RAM Module 5](#_Toc478475581)

[ROM Module 5](#_Toc478475582)

[Register File 5](#_Toc478475583)

[Control Unit 5](#_Toc478475584)

[V. Simulation Result 5](#_Toc478475585)

[Format A Instructions 5](#_Toc478475586)

[Format B Instructions 5](#_Toc478475587)

[Format L Instructions 5](#_Toc478475588)

[VI. Analysis and Discussion 5](#_Toc478475589)

[Hardware Verification Procedures 5](#_Toc478475590)

[VII. Conclusion 5](#_Toc478475591)

[VIII. Appendices 5](#_Toc478475592)

[IX. References 5](#_Toc478475593)

# Objective

## Abstract

Modern processors implement pipelining to increase throughput. Many current processors implement pipelining with various stage depths having more than 5 stages such as Intel, AMD, and even current ARM processors. Pipelining is a standard across all modern architecture designs.

## Project Objective

Build a processor; using VHDL to describe the processor, and Xilinx to synthesize and implement the design on a Spartan 3E FPGA.

# Design Requirement

## General Design Requirements

### Instruction Set

The project must implement a 16-bit RISC-like instruction set. Following the format outlined on the lab website. (Dimopolous & Hazmi, 2017) The instructions are all 1-word long.

The processor must handle three types of instructions:

* A-Format (arithmetic instructions)
* B-Format (branch instructions)
* L-Format (load and store instructions)

### Registers and Memory

Registers and Memory require certain implementations:

* 8 16-bit general purpose registers, each 1-word long, with word alignment.
* A memory address space that is byte addressable.

### Pipelining

The design must implement pipelining, using a 5 stage MIPS-like pipeline. All pipeline hazards must be handled.

### Program Execution

Regarding these types of instructions, the processor must be able to execute any program written using this instruction set. All RAW and WAR hazards must be handled.

## General Implementation Choices

### Data Forwarding and Hazard Detection

Operand addresses are monitored in each stage so that possible data hazards can be detected during instruction decode. Data forwarding control signals are then assigned where applicable.

### Branch Prediction

Our design implements an assumption of branches being not taken. If the branch is taken, the previous two pipeline registers are flushed and replaced with NOP instructions.

# System

## High Level System Hierarchy Breakdown

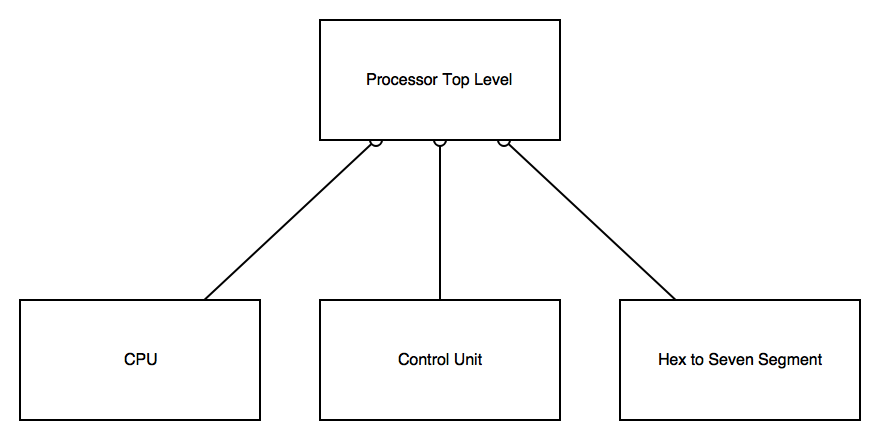


Figure 1 - System Hierarchy Breakdown

The processor is made up of a top-level file connecting the CPU, Control Unit, and Hex to Seven Segment sub-modules together. The CPU module contains its own submodules illustrated in the figure below.

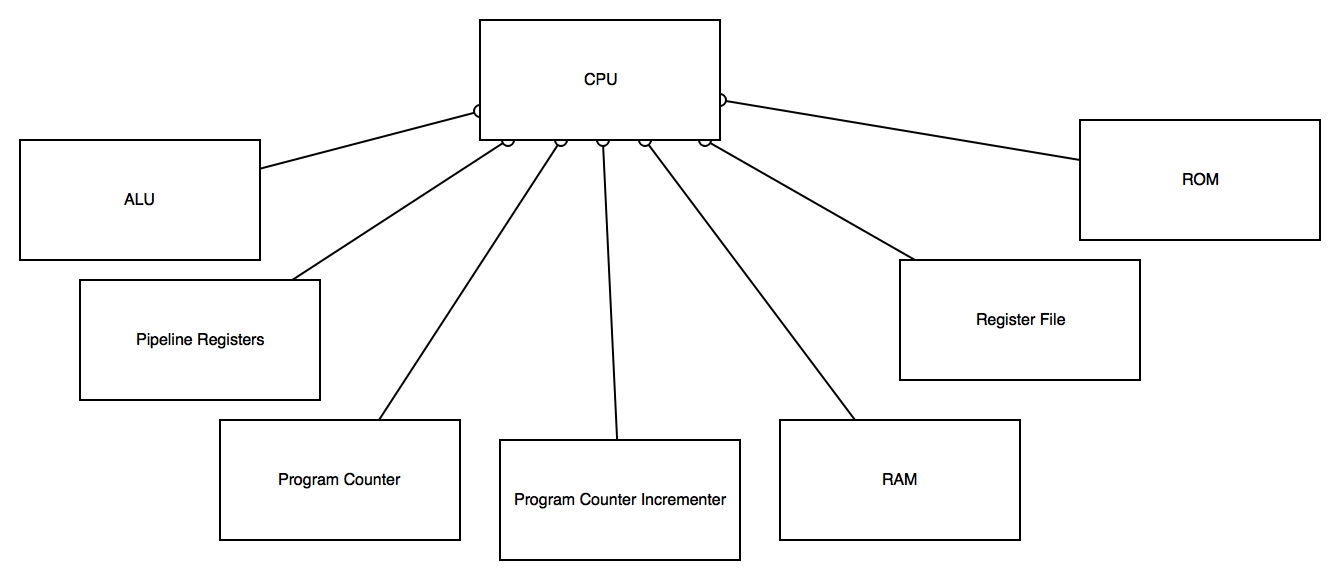
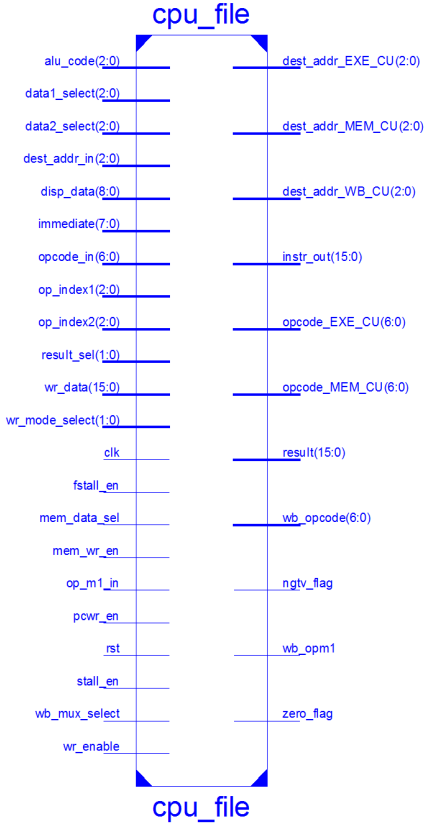


Figure 2 - CPU Hierarchy Breakdown

## High Level System Schematic Breakdown



# Individual Module Design

## Top Level Processor Unit

Connects the CPU to the Control Unit.

## CPU

The CPU contains the data path of the processor. Control signals from the control unit implement the instructions read from the program memory.

### ALU

Executes arithmetic instructions.

### Multiplexers

Data select signals from the control unit modify the flow of data and addresses based on the needs of the instruction and are used for the implementation of data forwarding.

### Program Counter

The Program counter continually increments through program memory addresses. The count can be held during a stall and branch instructions are able to overwrite the current address value.

### Pipeline Registers

These registers contain the necessary information that is needed by an instruction for the next stage of the pipeline. Information about the instruction that the control unit needs to monitor is also carried forward into the next stage through these registers. The results of the previous stage are written into the register on a falling clock edge and are released into the next stage on a rising clock edge. Reset signals zero out the entire register which effectively replaces the contents with a NOP instruction.

### RAM Module

This is the data memory. Load and store instructions interact with this module.

### ROM Module

This is the program memory and contains the instructions that will be executed by the processor. The program counter sets the address to fetch the instruction from.

### Register File

Contains the 8 general purpose registers used by the processor.

## Control Unit

The design goal for the control unit was to have it be purely combinatorial.

### Data Forwarding and Hazard Detection

Write addresses throughout the pipeline are monitored and compared to the operands of the incoming instruction during decode. A 6-bit code is used to by a process to determine whether a hazard is present and whether to perform data forwarding or a stall.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | 0 |
| RA | | RB | | RC | |

|  |  |
| --- | --- |
| 00 | No Hazard |
| 01 | Execution |
| 10 | Memory |
| 11 | Write Back |

Based on the current instruction in decode the process checks the code and data forwards from the appropriate stage when possible. For hazards where the data isn’t ready until a later stage stalls are introduced until forwarding can occur. Forwarding is implemented by setting the data select lines or the multiplexers appropriately.

## Hex to Seven Segment Display

# Simulation Result

## Format A Instructions

## Format B Instructions

## Format L Instructions

# Analysis and Discussion

## Hardware Verification Procedures

# Conclusion

# Appendices

## Module VHDL Code

## RTL Schematics

# References