Taylor Skilling and Will Johnson
EECE 3324
December 7th, 2016

Single-Cycle MIPS Processor

For the final project of Computer Architecture, a single-cycle MIPS processor was created in Verilog and tested with a given program, which found the minimum and maximum values of an array. Building off of the components generated in Homework 6, a Verilog file was created to control each piece of the processor and accomplish the given task successfully. Key metrics were gathered from the program and are presented below along with a visualization of the program. The major modules of the processor are defined in detail below.

Major Components

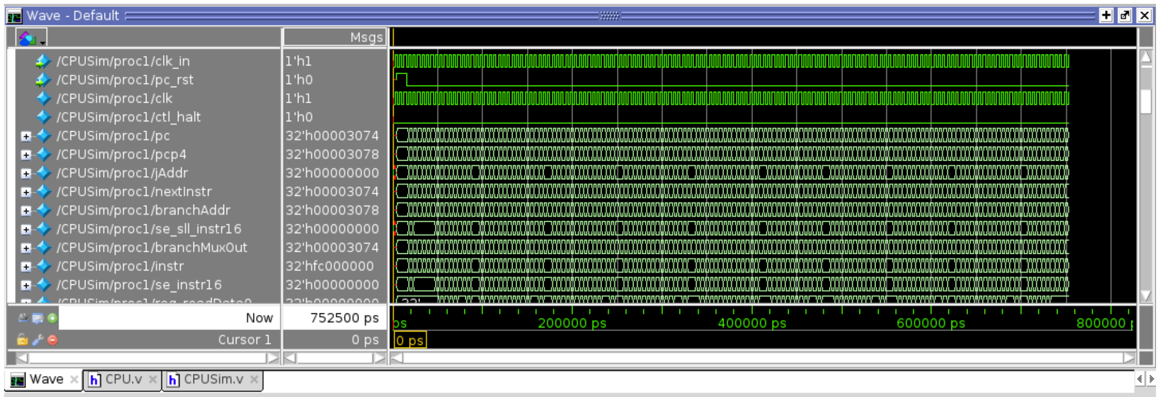
- **CPU** – The CPU module acted as the starting point of the processor, initializing and instantiating each individual module. This module also creates the wires that span the major modules, such as the ALU and register file.
- **ALU** – The ALU, or arithmetic logic unit, performs addition, subtraction, and set less than instructions on given inputs.
- **ALU CONTROL** – The ALU control unit determines from the instruction which operation the ALU should perform.
- **CONTROL UNIT** – This module determines the signals, which enabled the different modules in the CPU based on the instruction's *opcode*.
- **REGISTER FILE** – Contains 32 32-bit registers and read and write capabilities
- **MEMORY** – A given file, which reads a hexdump with the program and sends the correct instruction when given an instruction address. It also manages the data memory and the instruction memory.

Development Process

As the major components were already created in a previous assignment, the design task was centered on piecing these pieces together in a coherent way and ironing out the nuances of a full processor. By printing a large diagram of a single-cycle processor on a large sheet of paper and designating inputs and outputs with a standard naming convention for each major component, connecting components was made significantly easier. An extra part of the ALU test bench was added to ensure the signed SLL operation performed as expected. A behavioral model was implemented for most modules, as it closely resembles C, a familiar programming language. Care was taken to ensure that each module had correct functionality, especially the control unit, which required a small addition to support the *addi* instruction. The processor was implemented successfully following these design strategies and the results are shown below.

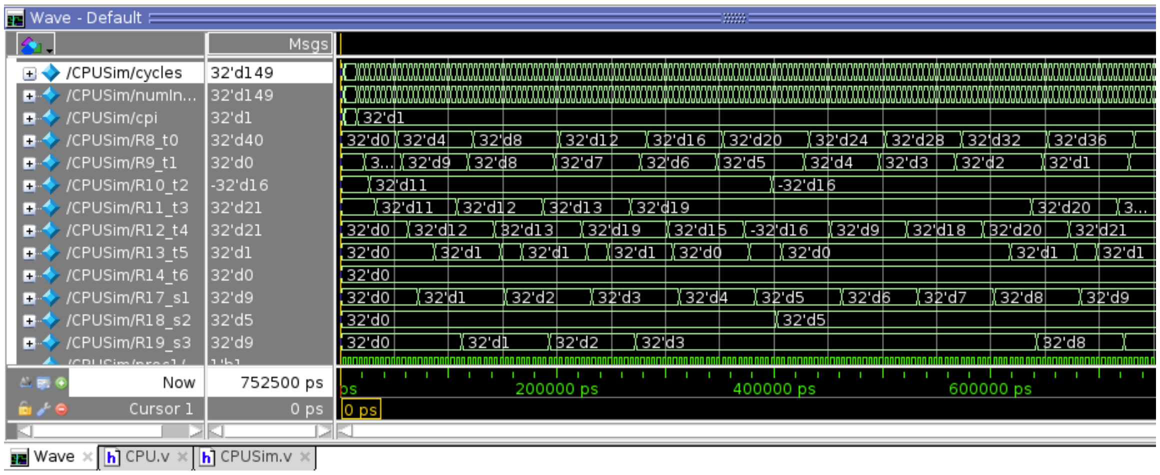
Simulation Results

The two figures below show the correct results of the single-cycle processor including the 149 cycles, a CPI of 1, and the correct register values



```
Wave - Default
                                   Msgs
/CPUSim/proc1/clk_in               1'h1
/CPUSim/proc1/pc_rst               1'h0
/CPUSim/proc1/clk                  1'h1
/CPUSim/proc1/ctl_halt             1'h0
/CPUSim/proc1/pc                   32'h00003074
/CPUSim/proc1/pcp4                 32'h00003078
/CPUSim/proc1/jAddr                32'h00000000
/CPUSim/proc1/nextInstr            32'h00003074
/CPUSim/proc1/branchAddr           32'h00003078
/CPUSim/proc1/se_sll_instr16       32'h00000000
/CPUSim/proc1/branchMuxOut         32'h00003074
/CPUSim/proc1/instr                32'hfc000000
/CPUSim/proc1/se_instr16           32'h00000000
                           Now     752500 ps
                      Cursor 1     0 ps
```

Figure 1 – Waveform screenshot showing 149 cycles and instructions, resulting in a CPI of 1. Also shown are the correct register output values at the end of the simulation.



```
Wave - Default
                          Msgs
/CPUSim/cycles            32'd149
/CPUSim/numIn...          32'd149
/CPUSim/cpi               32'd1
/CPUSim/R8_t0             32'd40
/CPUSim/R9_t1             32'd0
/CPUSim/R10_t2            -32'd16
/CPUSim/R11_t3            32'd21
/CPUSim/R12_t4            32'd21
/CPUSim/R13_t5            32'd1
/CPUSim/R14_t6            32'd0
/CPUSim/R17_s1            32'd9
/CPUSim/R18_s2            32'd5
/CPUSim/R19_s3            32'd9
                 Now      752500 ps
            Cursor 1      0 ps
```

Figure 2 – Waveform screenshot showing the correct halt instruction and other register values.