Organization of Digital Computer Lab
EECS112L/CSE 132L

Assignment 4
Multi-Cycle Processor

prepared by: Team Stressed
Student name:
Mansi Tyagi
Student ID:
23334840

Student name:
Erik Henriquez
Student ID:
57374677

Student name:
Kevin Chau
Student ID:
76934313

Student name:
Steven Chow
Student ID:
70916812

Student name:
Paul Dao
Student ID:
30658761

EECS Department
Henry Samueli School of Engineering
University of California, Irvine

February 22, 2016

1 Summary of Processor Design

1.1 What We Learned

A multicycle processor is able to use shorter clock cycles, since it doesn't need to accomodate load word, the slowest instruction. The multicycle processor splits the datapath so that instructions can be read or written into and the ALU can be accessed at each step.

1.2 Description

To change from a single-cycle processor to a multicycle processor we needed to eliminate two adders, expand the controller port list, and combine the instruction and data memories. Removing the excess adders reduced the number of costly components the datapath was using. Instead, because of the new datapath, the ALU, which includes an adder, is accessible at every step. The combined memories mimics reality, in which computers have a large memory, rather than two separate memories. This reduces the length of the datapath, increasing the frequency and decreasing the number of cycles. It also ensured that the memory, either instruction or data, could be accessed from any step. The controller signals have been adjusted to accomodate the combined Instruction and Data Memory to include a IorD signal, which indicates which memory values need to be stored or accessed from.

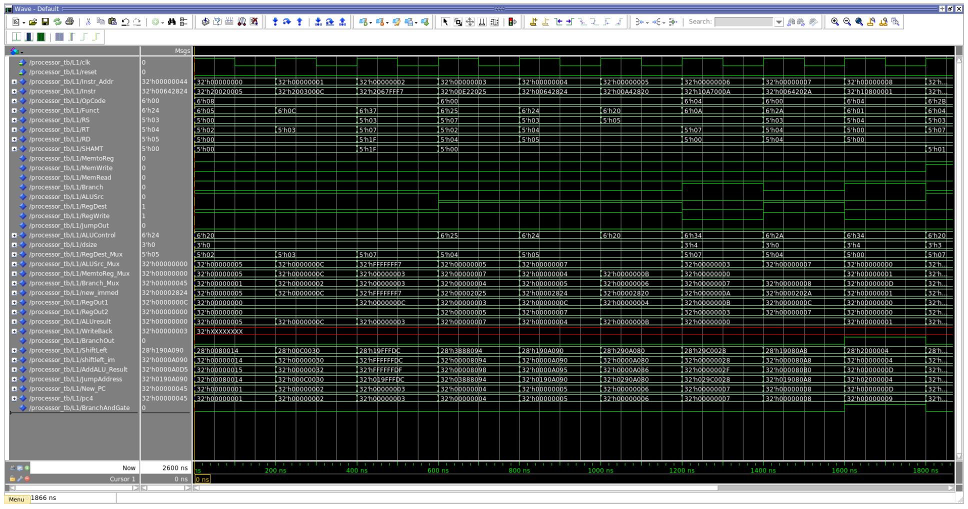
1.3 Testbench Architecture

For the most part, our testbench is maintaining the same cycles as it did for the single-cycle processor, simply because we didn't have time to change it. However, since we are now implementing a multicycle processor, different instructions will require different amounts of time to complete. Jump and branch instructions are the fastest instructions, requiring 3 cycles, while load word is the slowest at 5 cycles.

2 Sample Program

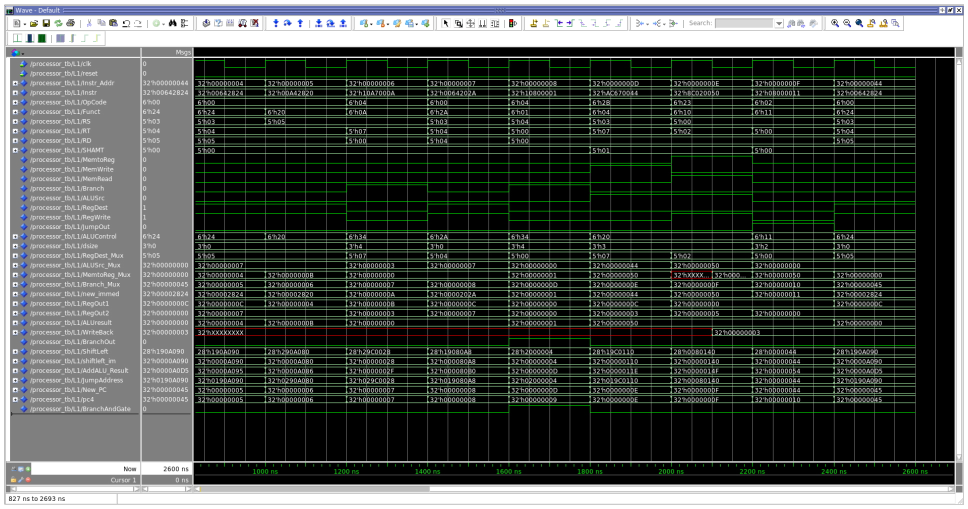
2.1 Simulation Waveform

After designing and testing our Processor, we generated the following waveforms:



```
Wave - Default
                                      Msgs
/processor_tb/L1/clk                  0
/processor_tb/L1/reset                0
/processor_tb/L1/Instr_Addr           32'h00000044
/processor_tb/L1/Instr                32'h00642824
/processor_tb/L1/OpCode               6'h00
/processor_tb/L1/Funct                6'h24
/processor_tb/L1/RS                   5'h03
/processor_tb/L1/RT                   5'h04
/processor_tb/L1/RD                   5'h05
/processor_tb/L1/SHAMT                5'h00
/processor_tb/L1/MemtoReg             0
/processor_tb/L1/MemWrite             0
/processor_tb/L1/MemRead              0
/processor_tb/L1/Branch               0
/processor_tb/L1/ALUSrc               0
/processor_tb/L1/RegDest              1
/processor_tb/L1/RegWrite             1
/processor_tb/L1/JumpOut              0
/processor_tb/L1/ALUControl           6'h24
/processor_tb/L1/dsize                3'h0
/processor_tb/L1/RegDest_Mux          5'h05
/processor_tb/L1/ALUSrc_Mux           32'h00000000
/processor_tb/L1/MemtoReg_Mux         32'h00000000
/processor_tb/L1/Branch_Mux           32'h00000045
/processor_tb/L1/new_immed            32'h00002824
/processor_tb/L1/RegOut1              32'h0000000C
/processor_tb/L1/RegOut2              32'h00000000
/processor_tb/L1/ALUresult            32'h00000000
/processor_tb/L1/WriteBack            32'h00000003
/processor_tb/L1/BranchOut            0
/processor_tb/L1/ShiftLeft            28'h190A090
/processor_tb/L1/shiftleft_im         32'h0000A090
/processor_tb/L1/AddALU_Result        32'h0000A0D5
/processor_tb/L1/JumpAddress          32'h0190A090
/processor_tb/L1/New_PC               32'h00000045
/processor_tb/L1/pc4                  32'h00000045
/processor_tb/L1/BranchAndGate        0

Now                                   2600 ns
Cursor 1                              0 ns
1866 ns
```



```
Wave - Default
                                          Msgs
/processor_tb/L1/clk                      0
/processor_tb/L1/reset                    0
/processor_tb/L1/Instr_Addr               32'h00000044
/processor_tb/L1/Instr                    32'h00642824
/processor_tb/L1/OpCode                   6'h00
/processor_tb/L1/Funct                    6'h24
/processor_tb/L1/RS                       5'h03
/processor_tb/L1/RT                       5'h04
/processor_tb/L1/RD                       5'h05
/processor_tb/L1/SHAMT                    5'h00
/processor_tb/L1/MemtoReg                 0
/processor_tb/L1/MemWrite                 0
/processor_tb/L1/MemRead                  0
/processor_tb/L1/Branch                   0
/processor_tb/L1/ALUSrc                   0
/processor_tb/L1/RegDest                  1
/processor_tb/L1/RegWrite                 1
/processor_tb/L1/JumpOut                  0
/processor_tb/L1/ALUControl               6'h24
/processor_tb/L1/dsize                    3'h0
/processor_tb/L1/RegDest_Mux              5'h05
/processor_tb/L1/ALUSrc_Mux               32'h00000000
/processor_tb/L1/MemtoReg_Mux             32'h00000000
/processor_tb/L1/Branch_Mux               32'h00000045
/processor_tb/L1/new_immed                32'h00002824
/processor_tb/L1/RegOut1                  32'h0000000C
/processor_tb/L1/RegOut2                  32'h00000000
/processor_tb/L1/ALUresult                32'h00000000
/processor_tb/L1/WriteBack                32'h00000003
/processor_tb/L1/BranchOut                0
/processor_tb/L1/ShiftLeft                28'h190A090
/processor_tb/L1/shiftleft_im             32'h0000A090
/processor_tb/L1/AddALU_Result            32'h0000A0D5
/processor_tb/L1/JumpAddress              32'h0190A090
/processor_tb/L1/New_PC                   32'h00000045
/processor_tb/L1/pc4                      32'h00000045
/processor_tb/L1/BranchAndGate            0

Now                                       2600 ns
Cursor 1                                  0 ns
827 ns to 2693 ns
```

As you can see, our branch and jump instructions worked as expected. Of the 18 instruction in the example program, only 12 were actually executed. We included a 13th cycle in our waveform to display the "garbage" instruction after our jump instruction.

3 Synthesis

We were unable to finish synthesizing the multicycle processor we created.

4 Known Issues

Due to our unfamiliarity with SystemVerilog, we chose not to use the testbench to preload instruction memory. For the Questasim simulation, we read the example program instructions from a file. For synthesis, we created a second version of instruction memory that has instructions preloaded into memory in order to synthesize the Processor properly.

5 Conclusion

As a multicycle processor, our project supports the full MIPS instruction set and allows faster instructions to use shorter clock cycles.