| Class: | CPE-300L | Semester: | Spring 2020 |
|---|---|---|---|

| Points | | Document author: | Brysen Kokubun |
|---|---|---|---|
| | | Author's email: | Brysenkokubun96@gmail.com |
| | | | |
| | | Document topic: | Postlab 9 |

Instructor's comments:

## 1. Introduction / Theory of Operation

- For this lab experiment we learned about the CPU which implemented a single cycle implementation of limited subset of MIPS instructions. We verified the MIPS instructions using ModelSim and followed the waveform to ensure that the instructions were moving values in particular registers and memory addresses.

***a. What are main elements of single cycle Datapath?***

- Main elements of single cycle Datapath are: Program Counter register, Multiplexer, Adder, Control Unit, Register file, Shifter, ALU, Instruction memory, Data memory.

***b. What kind of circuit is the control unit circuit which generates the control signals; combinational or FSM? Why?***

- A control unit itself is a FSM because it is a model of computation in a finite number of states. But the signal from the control unit is combinational because the output is a function of present input.

***c. Analyze the Instruction set and report what are other functional units that are to be added to ALU. Note, that floating point operations are performed on a Coprocessor***

- add, subtract, and, or set less than, load, store, beq, add immediate, R-type

## 2. Prelab report

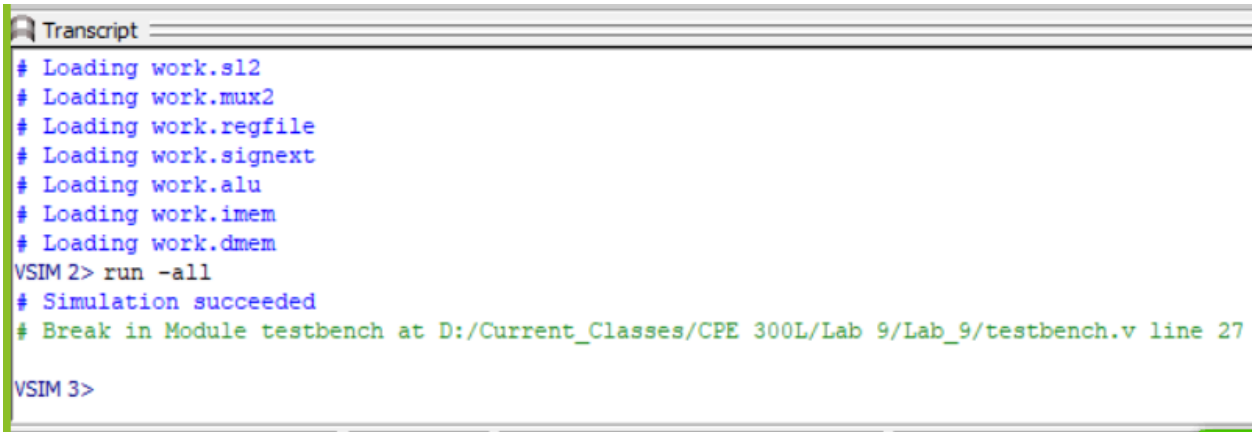
- Attached below

## 3. Experiment Results

### *Experiment #1: Testbench for MIPS subset*

```
module testbench();
reg clk;
reg reset;
wire [31:0] writedata, dataadr;
wire memwrite;

// instantiate device to be tested
top dut (clk, reset, writedata, dataadr, memwrite);

// initialize test
initial
begin
    reset <= 1; # 22; reset <= 0;
end
// generate clock to sequence tests
always
begin
    clk <= 1;
    # 5;
    clk <= 0;
    # 5; // clock duration
end

// check results
always @ (negedge clk)
begin
    if (memwrite)
    begin
        if (dataadr === 84 & writedata === 7)
        begin
            $display ("Simulation succeeded");
            $stop;
        end
        else if (dataadr !== 80)
        begin
            $display ("Simulation failed");
            $stop;
        end
    end
end

endmodule
```

Ran the testbench.v file and it successfully ran and executed. Within ModelSim's waveform you can see the values being placed into various registers. "See link for explanation" https://drive.google.com/open?id=1MLjSfFfp92l4MILbcCEHbbDRKMlKXa-e




### *Experiment#2: MIPS implementation on DE2*

- Excused from Biniyak

***Experiment#3: TimeQuest analysis***

Setup: clk

Command Info | Summary of Paths

| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay |
|---|---|---|---|---|---|---|---|---|
| 1 | -6.882 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1446 | clk | clk | 1.000 | -0.290 | 7.628 |
| 2 | -6.854 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1414 | clk | clk | 1.000 | -0.300 | 7.590 |
| 3 | -6.755 | mips:mips\|datapath:dp\|flopr:pcreg\|q[6] | dmem:dmem\|RAM~1446 | clk | clk | 1.000 | -0.290 | 7.501 |
| 4 | -6.745 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1422 | clk | clk | 1.000 | -0.300 | 7.481 |
| 5 | -6.727 | mips:mips\|datapath:dp\|flopr:pcreg\|q[6] | dmem:dmem\|RAM~1414 | clk | clk | 1.000 | -0.300 | 7.463 |
| 6 | -6.726 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1510 | clk | clk | 1.000 | -0.290 | 7.472 |
| 7 | -6.713 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1478 | clk | clk | 1.000 | -0.301 | 7.448 |
| 8 | -6.687 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~466 | clk | clk | 1.000 | -0.283 | 7.440 |

**Path #1: Setup slack is -6.882 (VIOLATED)** **Path #1: Setup slack is -6.882 (VIOLATED)**

TimeQuest shows that there is a negative slack of -6.882 which means that the 1ns clock period is too fast. Data is being received AFTER the data is required.

Setup: clk

Command Info | Summary of Paths

| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.118 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1446 | clk | clk | 8.000 | -0.290 | 7.628 |
| 2 | 0.146 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1414 | clk | clk | 8.000 | -0.300 | 7.590 |
| 3 | 0.245 | mips:mips\|datapath:dp\|flopr:pcreg\|q[6] | dmem:dmem\|RAM~1446 | clk | clk | 8.000 | -0.290 | 7.501 |
| 4 | 0.255 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1422 | clk | clk | 8.000 | -0.300 | 7.481 |
| 5 | 0.273 | mips:mips\|datapath:dp\|flopr:pcreg\|q[6] | dmem:dmem\|RAM~1414 | clk | clk | 8.000 | -0.300 | 7.463 |
| 6 | 0.274 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1510 | clk | clk | 8.000 | -0.290 | 7.472 |
| 7 | 0.287 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~1478 | clk | clk | 8.000 | -0.301 | 7.448 |
| 8 | 0.313 | mips:mips\|datapath:dp\|flopr:pcreg\|q[4] | dmem:dmem\|RAM~466 | clk | clk | 8.000 | -0.283 | 7.440 |

**Path #1: Setup slack is 0.118** **Path #1: Setup slack is 0.118**

After altering the clock period to 8.00ns TimeQuest shows that there is a positive slack of 0.118. This means that Data is being received BEFORE the data is required. Therefore, all timing requirements are met.

There were 13 instructions/cycles within the Verilog file.
Each instruction took an 8ns clock period cycle to meet timing requirements.
Therefore, the whole Verilog file takes 104ns of cycle time to run.

## 4. Question

***1. What distinguishes MIPS from other processor architectures?***

- MIPS architecture has Load/Store architecture where load and store instructions are only allowed to access data memory. Also, MIPS architecture supports the implementation of multiple banks registers.

***2. List 3 advantages of MIPS and 3 disadvantages***

3 advantages of MIPS:

- The architecture will allow systems to address large amounts of memory
- Faster I/O speeds to mem drives, disks, cards.
- Can handle more memory and larger files

3 disadvantages of MIPS:

- Larger memory increases memory requirements for processes
- Larger memory can create problems for efficiency of cache/flash
- Incompatibility with other architectures and device drivers.

## 5. Conclusions

- To conclude this lab experiment we learned about the components of the MIPS architecture and how MIPS instructions sets were implemented on the CPU. There weren't really any problems that occurred for this lab experiment except not having the lab in class. We used ModelSim to compile the Verilog files and used the waveform tool in order to verify the testbench and view the values within registers/memory addresses. TimeQuest was used to see the timing requirements for the MIPS instruction sets. The needed amount of clock period has to be 8.00ns in order for the timing requirements to be fulfilled. Overall this lab taught us about the CPU and how to implement a single cycle implementation of limited subset of MIPS instructions.

| Class: | CPE-300L | Semester: | Spring 2020 |
|---|---|---|---|

| Points | | Document author: | Brysen Kokubun |
|---|---|---|---|
| | | Author's email: | Brysenkokubun96@gmail.com |
| | | | |
| | | Document topic: | Prelab 9 |

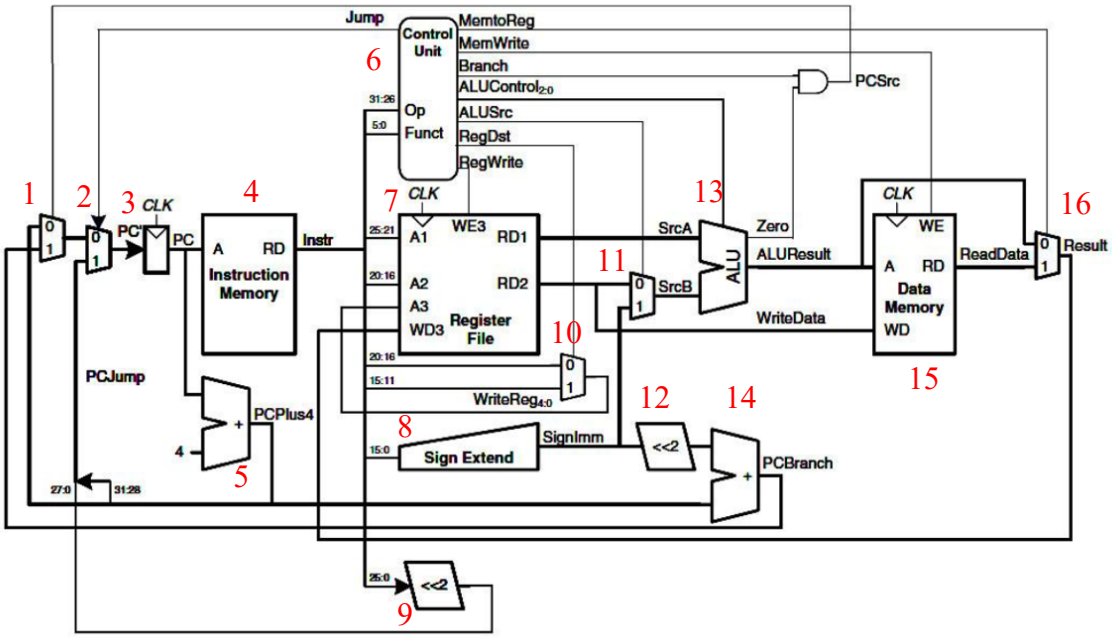
Instructor's comments:

## Introduction / Theory of operation

- For this lab experiment we will be learning on how to design a CPU by implementing a single cycle implementation of limited subset of MIPS instructions.

## Prelab main content

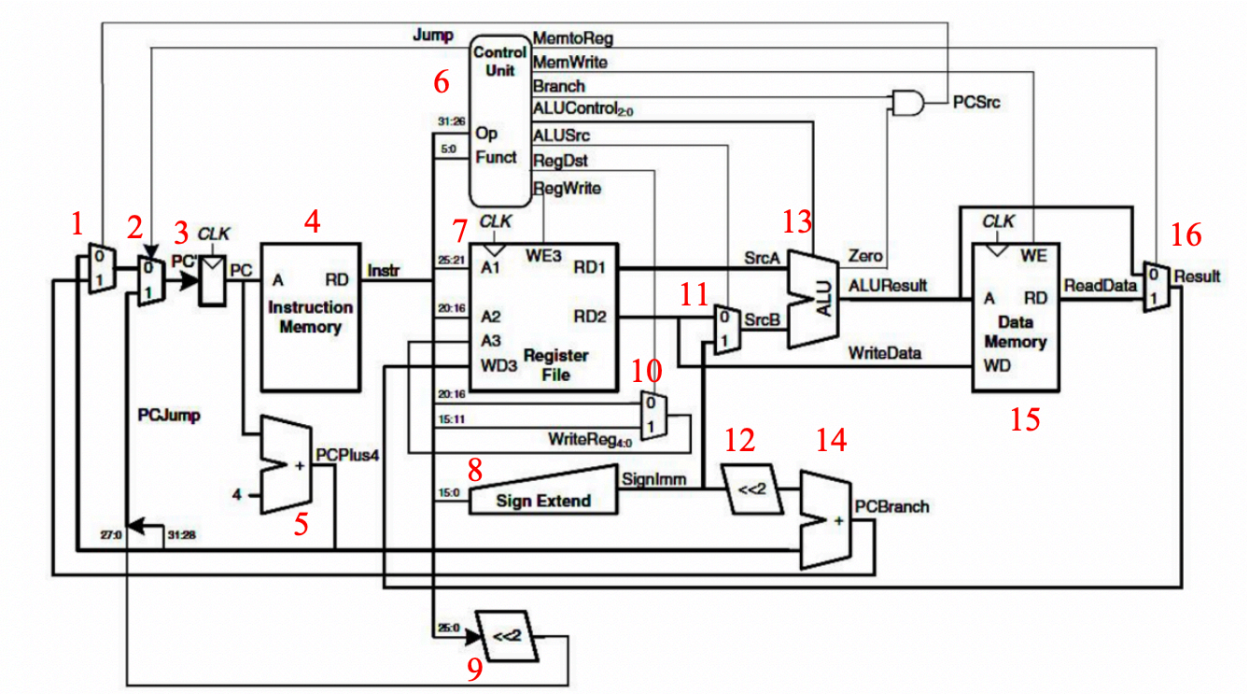
***1. Describe, what is the function of each block (there are 16 blocks) in Fig.1?***




1) Mux_1: Mux selection from PCSrc output for Mux_2
2) Mux_2: Mux selection from Jump output for PC'
3) Flip_flop: Program counter
4) Instruction memory: for Register destination of Program counter
5) Adder: Program counter + 4
6) Control Unit: Controls the operations of the other components
7) Register file: Addresses/write destinations/write enable/read destinations
8) Sign Extend: Address offset for I-type, destination/shift amount/function for R-type
9) Shifter: shifts 2 bits left
10) Mux_3: Mux selection from RegDst output for register file A3

11) Mux_4: Mux selection from ALUSrc output for ALU input SrcB
12) Shifter: shifts 2 bits left
13) ALU: performs arithmetic operations
14) Adder: (PC+4) + (Sign extend shifted 2 bits left)
15) Data Memory: write data/aluResult to Read data or pass through to Mux_5
16) Mux_5: Mux selection from memtoReg output for register file WD3

***2. Study the given code and write the name of module that generates each block (there are 16 blocks) in Fig.1.***




1) Module mux2.v
2) Module mux2.v
3) Module flopr.v
4) Module imem.v
5) Module adder.v
6) Module controller.v
7) Module regfile.v
8) Module signext.v
9) Module sl2.v
10) Module mux2.v
11) Module mux2.v
12) Module sl2.v
13) Module alu.v
14) Module adder.v
15) Module dmem.v
16) Module mux2.v