**ECEGR 2220: Microprocessor Design**

**Spring 2018**

**LAB 6 REPORT**

**Name: Don Le**

**Performed by:**

**Don Le - Thanh Nguyen – Lauren Molina**

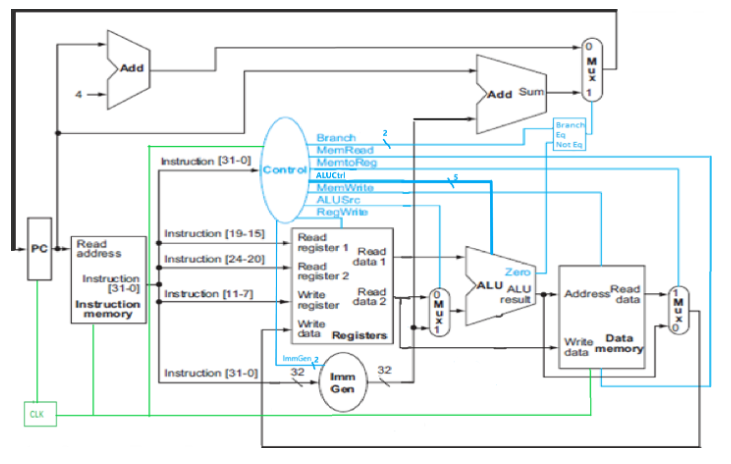
**Date of Report: 06/14/2018**

**SEATTLE UNIVERSITY**

**Department of Electrical and Computer Engineering**

**INTRODUCTION:**

In this lab, we wrote VHDL code to implement the RISC-V single cycle processor design below.



We used components that we created in previous labs which are:

* ALU.vhd
* Registers.vhd
* RAM.vhd,

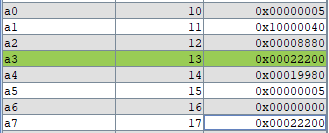
along with two new files: Processor.vhd and ProcElements.vhd, in which we mapped and encoded the components. To ensure we did everything correctly, we executed the test bench tProcessor.vhd to test our designs.

* **What is the program in Instruction RAM doing?**

Instruction RAM is reading the address from the program counter and outputting the instructions at that address.

* **What are the expected results in data RAM when the program completes?**

We ran the file app.s on RARS to get the expected results. Based on the execution, data Ram should contains these values when the program completes



**TRUTH TABLE**

**R-Type:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | R-type | Lw | Sw | Beq | Bne | Addi | Ori | Andi | lui |
| Branch[1-0] | 00 | 00 | 00 | 10 | 11 | 00 | 00 | 00 | 00 |
| MemRead | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| MemtoReg | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ALUctrl | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| MemWrite | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUSrc | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Regwrite | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| ImmGen[1-0] | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I-Type:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Branch  [1-0] | MemRead | MemtoReg | ALUctrl  [4-0] | MemWrite | ALUSrc | Regwrite | ImmGen  [1-0] | Funct7 | Funct3 | Opcode |
| ADDI | 00 | 0 | 0 | 00000 | 0 | 1 | 1 | 01 | X | 000 | 0010011 |
| ORI | 00 | 0 | 0 | 00011 | 0 | 1 | 1 | 01 | 0000000 | 110 | 0010011 |
| ANDI | 00 | 0 | 0 | 00010 | 0 | 1 | 1 | 01 | 0000000 | 111 | 0010011 |
| LW | 00 | 0 | 1 | 01100 | 0 | 1 | 1 | 01 | 0000000 | 010 | 0000011 |
| SLLI | 00 | 0 | 0 | 00001 | 0 | 1 | 1 | 01 | 0000000 | 001 | 0010011 |
| SLRI | 00 | 0 | 0 | 01001 | 0 | 1 | 1 | 01 | 0000000 | 101 | 0010011 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Branch  [1-0] | MemRead | MemtoReg | ALUctrl  [4-0] | MemWrite | ALUSrc | Regwrite | ImmGen  [1-0] | Funct7 | Funct3 | Opcode |
| SW | 00 | 0 | 0 | 10100 | 1 | 1 | 0 | 10 | x | 010 | 0100011 |

**S-Type:**

**B-Type:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Branch  [1-0] | MemRead | MemtoReg | ALUctrl  [4-0] | MemWrite | ALUSrc | Regwrite | ImmGen  [1-0] | Funct7 | Funct3 | opcode |
| BEQ | 01 | 0 | 0 | 10000 | 0 | 0 | 0 | 00 | X | 000 | 1100011 |
| BNE | 10 | 0 | 0 | 01000 | 0 | 0 | 0 | 00 | x | 001 | 1100011 |

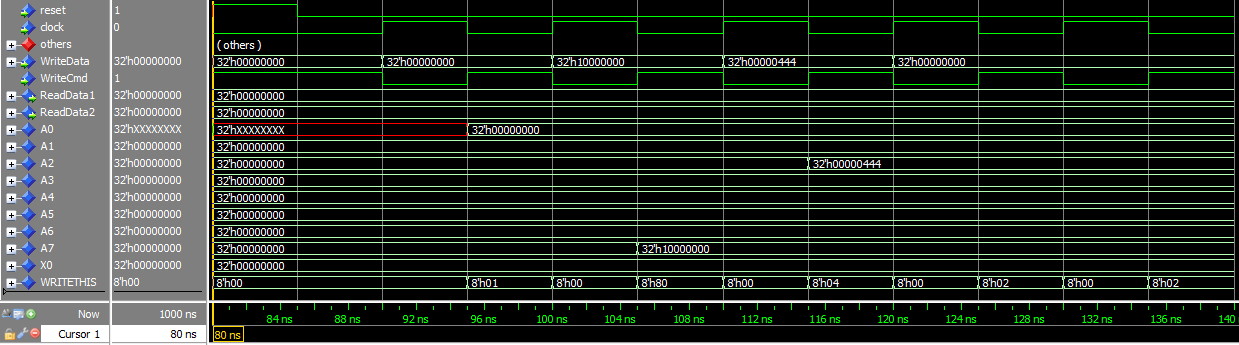
**U-Type:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Branch  [1-0] | MemRead | MemtoReg | ALUctrl  [4-0] | MemWrite | ALUSrc | Regwrite | ImmGen  [1-0] | Funct7 | Funct3 | opcode |
| LUI | 00 | 0 | 0 | 01111 | 0 | 1 | 1 | 11 | x | x | 0110111 |

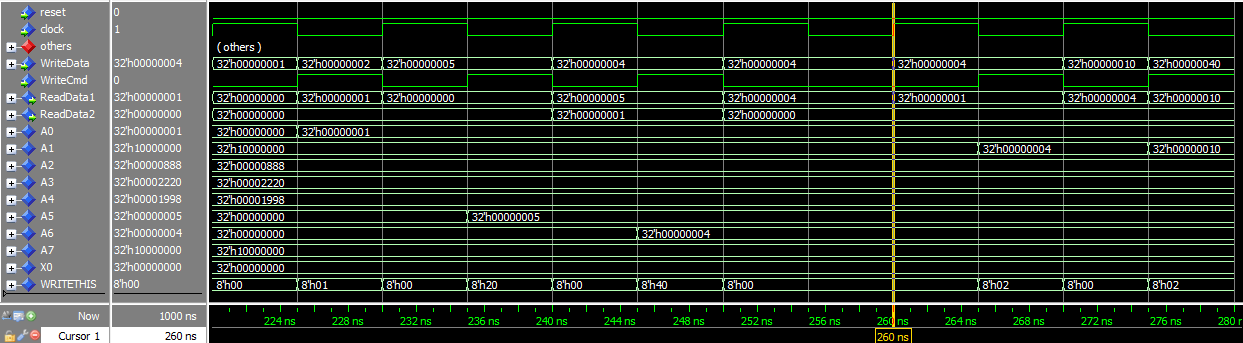
**For ImmGen:**

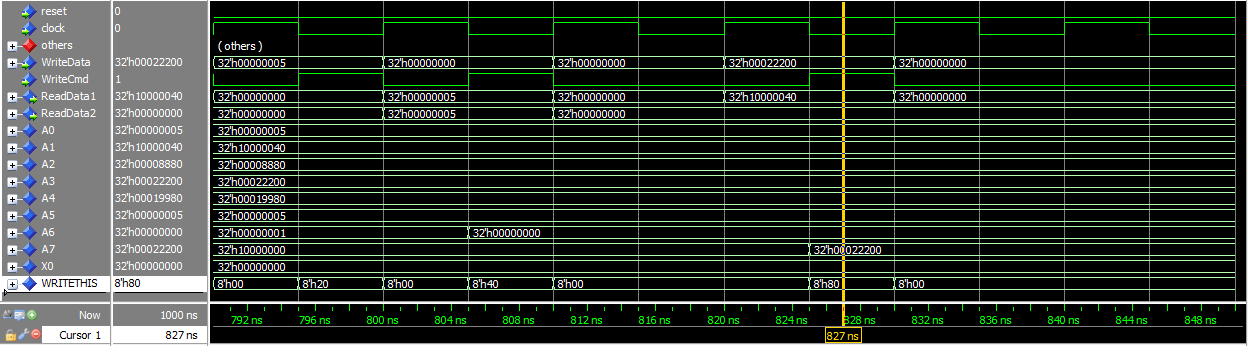
|  |  |
| --- | --- |
| Instruction type | ImmGen value |
| B type | 00 |
| I type | 01 |
| S type | 10 |
| U type | 11 |

**EXECUTION AND TESTING:**

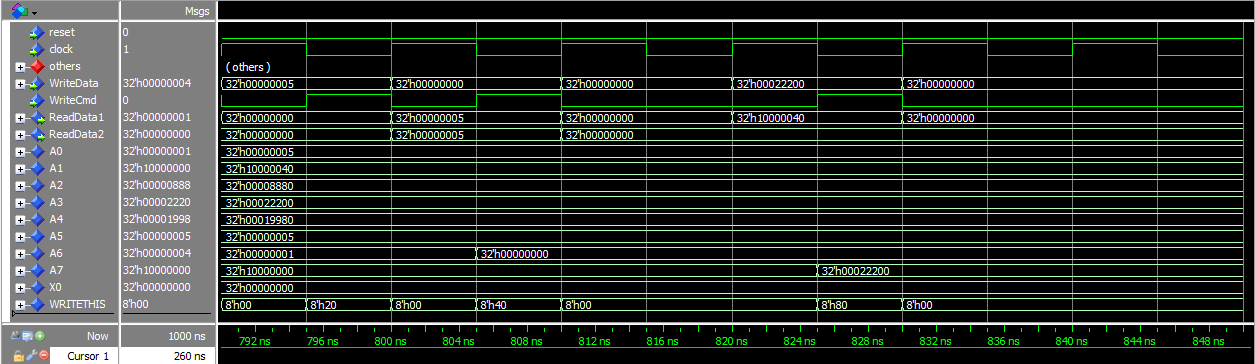
As the first instruction started as **80ns** during the execution, from this point, the RAM started to save data in slots A0 to A7.

The execution ran smoothly through instructions and data had been saved in RAM.

**At 260ns**, since the BNE condition was not met, the program jumped back to SLLI. This shows that the BNE works.

After the test program completes, we got the data memory contents as below. These matched the expected results that we got from app.s execution,

After the end of the program, the final result of A7 is 0x**00022200.** This matches the result we got from running app.s on RARS.



To improve the design of the test program, there could be more test cases such as ANDI, ORI which was not tested in the current test bench. Also, there could be more BNE and BEQ.

* Link for our github:
* Link to our master branch:

https://github.com/SU-ECEGR-2220/AVOCADOS/tree/master/Lab%206

* Link to individual branches:

Thanh: https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Thanh/Lab%206

Don: https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Don/Lab%206

Lauren: https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Lauren/Lab%206

* Note:
* Each individual built the program then saved and test on his/her own branch first. Then we came up with the best version of the program and pulled it to the master branch. You can find the contribution of each member towards the project in the individual branches. Thank you!