top-logo-en.jpg

Koç University

College of Engineering

COMP 303 Computer Architecture

Fall 2019

Project Final Report

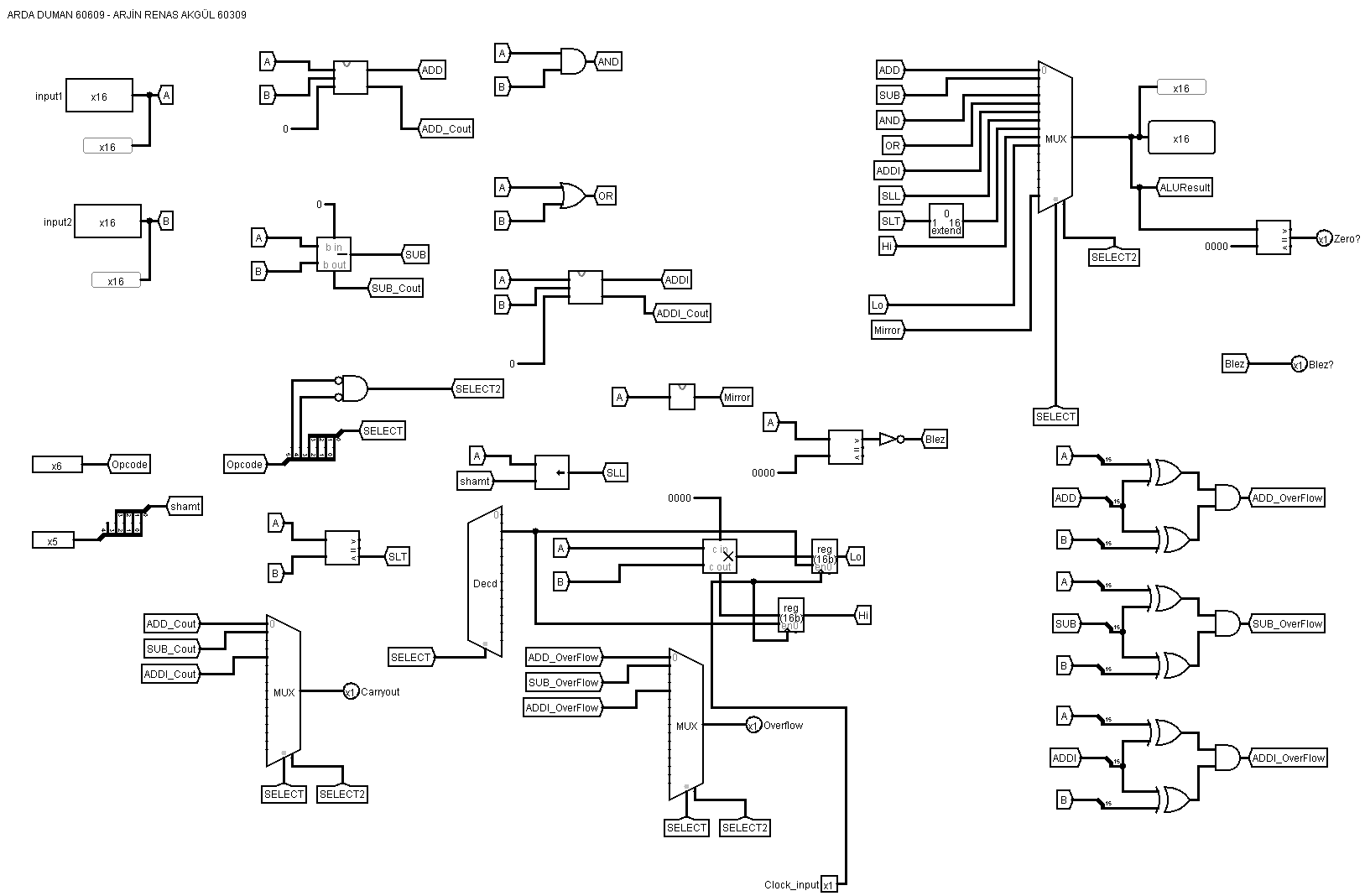
21/12/2019

Prepared by:

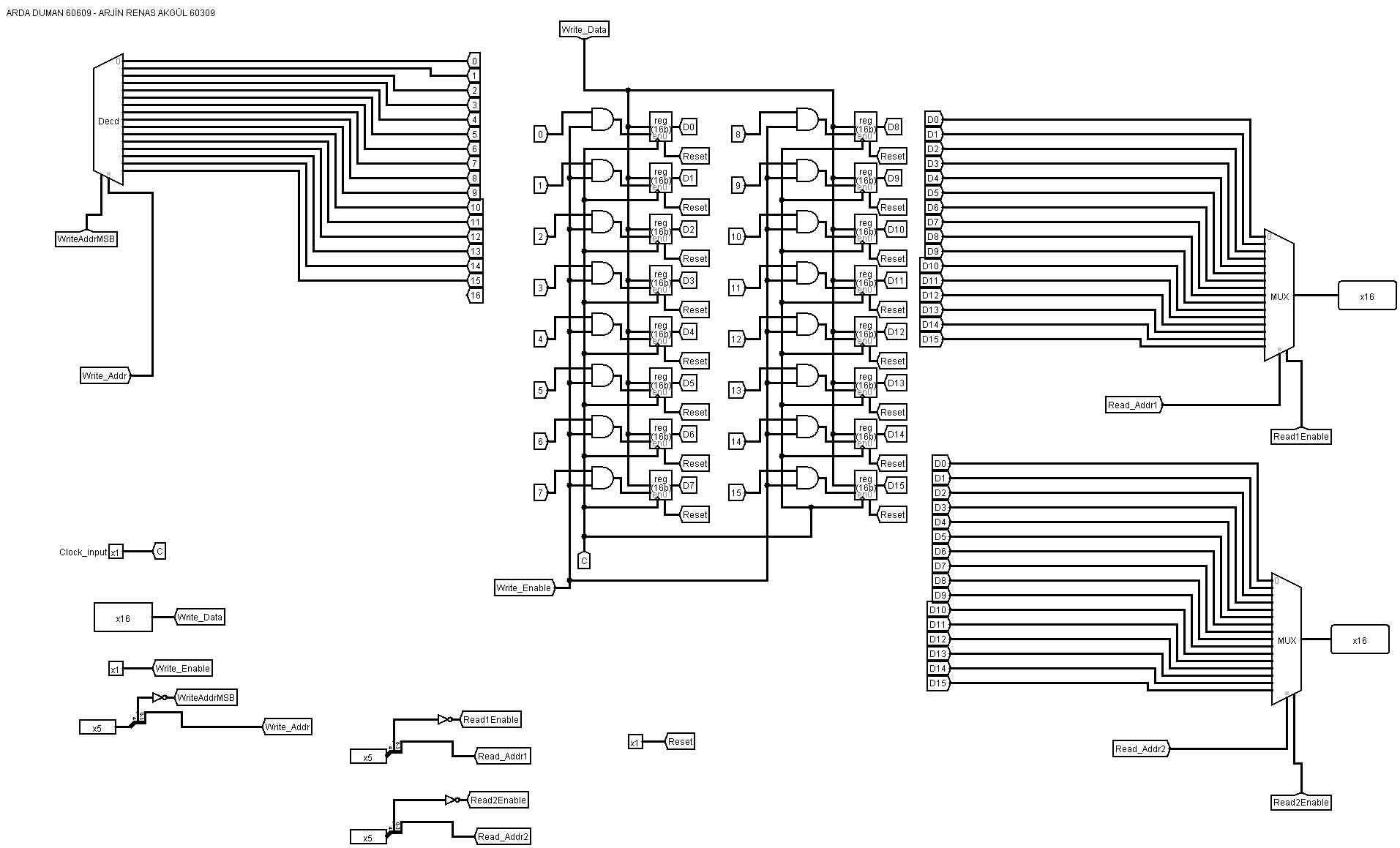
Arda Duman 60609

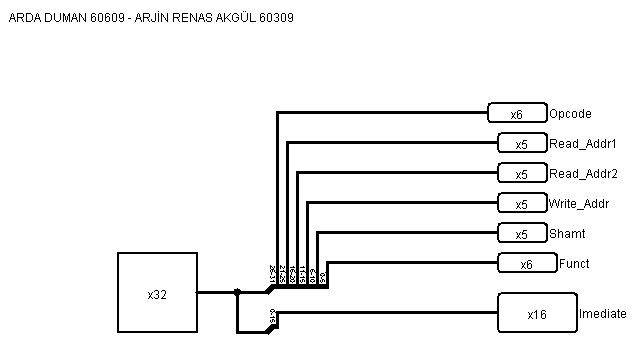
Arjin Renas Akgül 60309

**ALU:**

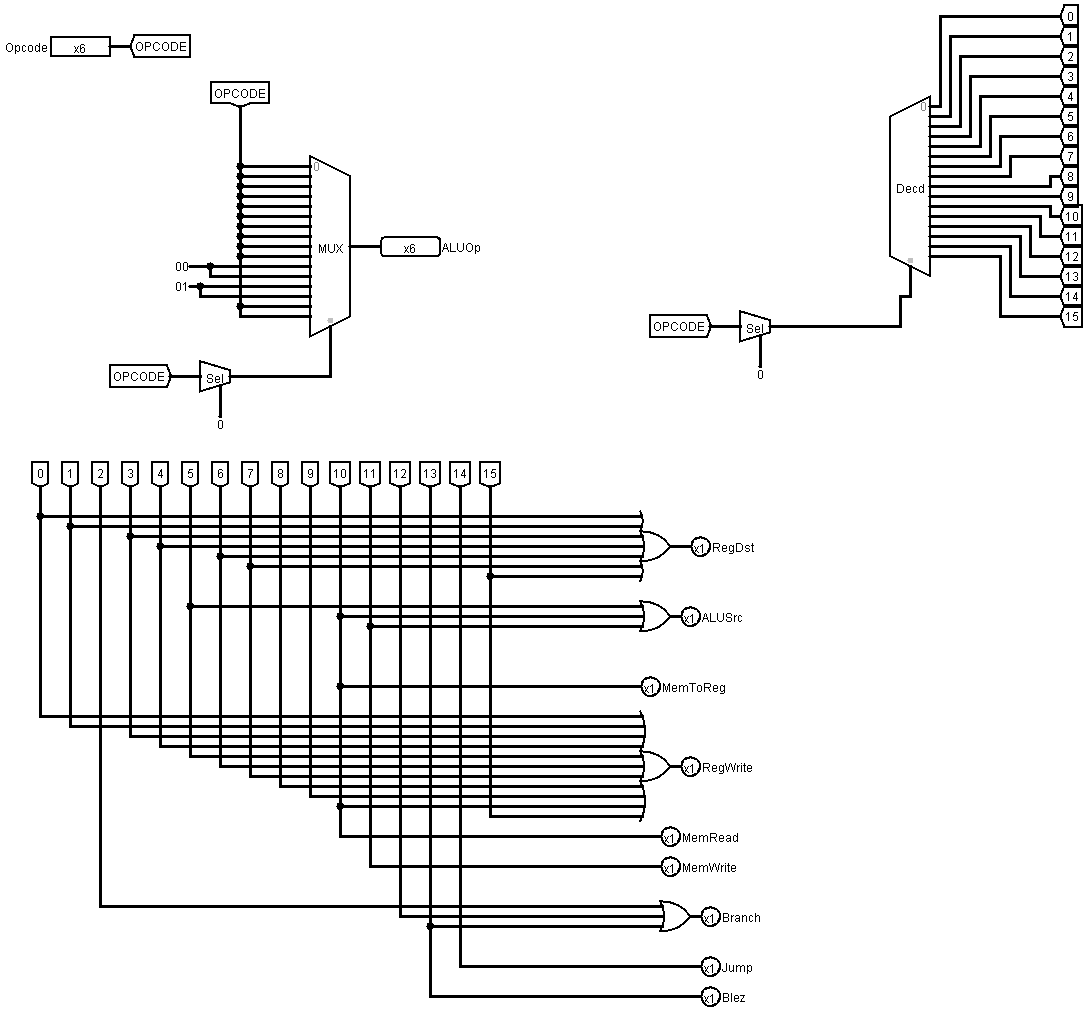
****

**Register File:**

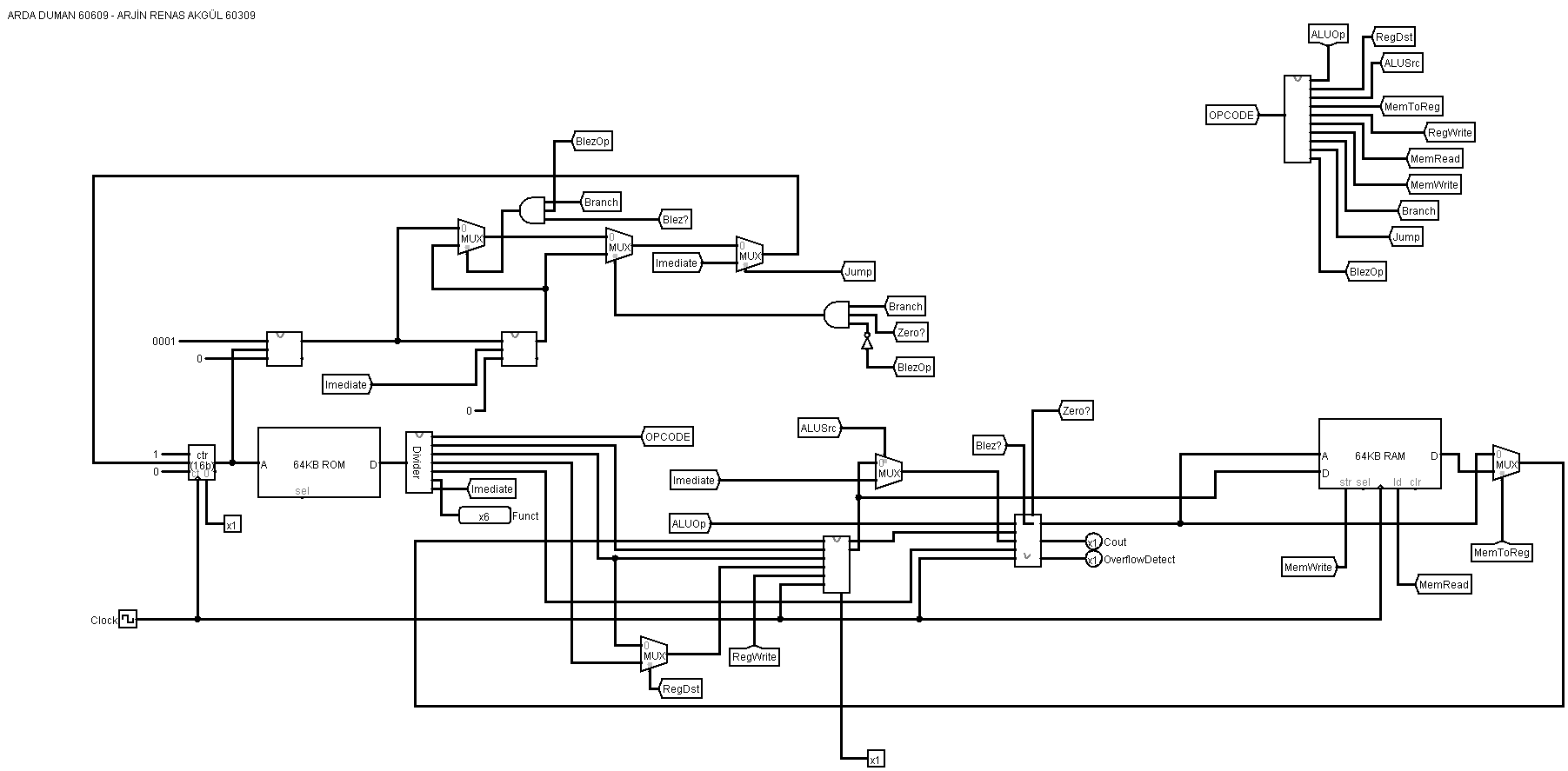
****

**Divider: **

**Control Unit:**

****

**Single Cycle Processor:**

****

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type | Instruction | Opcode | RegDst | AluSrc | MemtoReg | RegWrite | MemRead | MemWrite | Branch | Jump | Blez |
| R | add | 000000 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | sub | 000001 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | mult | 000010 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R | and | 000011 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | or | 000100 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| I | addi | 000101 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | sll | 000110 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | slt | 000111 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | mfhi | 001000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R | mflo | 001001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| I | lw | 001010 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| I | sw | 001011 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| I | beq | 001100 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| I | blez | 001101 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| J | j | 001110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R | myIns | 001111 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Control Unit Table:**

**Mirror Operation (My Instruction):**

****

Our Custom instruction is an R-Type instruction that takes only A as an input and reverses it. For example, if A is 0000000000000001, the result will be 1000000000000000.

We used splitter from the wiring library of the Logisim. We take rd and rs in asm code, so that our operation will be like myIns rd rs in asm.

**Design:**

As reference to the information and example related to our single cycle processor design, we used lecture materials. First, we designed our ALU to calculate add, sub, and, or, addi, sll, slt mfhi, mflo and myIns instructions. We have a Control Unit design to select which part of the Single Cycle Processor will work. Control Unit selects read, write, ALU operations, branch or jump operations by taking opcode as input. Control Unit behaves like shown in the table above. In our Single Cycle Processor, we write hexadecimal codes to our ROM according to the asm code. Then by these codes we put our registers to register files and operate the instructions. Then again according to our ROM, Single Cycle Processor decides to store the values into Data Memory.