**Group-6**

**Project:- Converting RISC-V Verilog Design to System Verilog**

Project Description:- In this project, the objective is to understand and convert the existing Verilog-modeled RISC-V Architecture into System Verilog. The design will be made as abstract as possible using the System Verilog’s constructs. Since we will be modeling a single-cycle processor, we will simulate the design with a test bench module.

By

Kai Roy ( [roykai@pdx.edu](mailto:roykai@pdx.edu) )

Sri Sai Sumanth Yadalapalli ( [srisaiy@pdx.edu](mailto:srisaiy@pdx.edu) )

Shruti Kotha ([skotha@pdx.edu](mailto:skotha@pdx.edu) )

Vijaya Manikanta Kotagiri ( [kotagiri@pdx.edu](mailto:kotagiri@pdx.edu) )

**11/14**

* Kai shared the following:
  + <https://drive.google.com/drive/folders/1RwhvwubAhtlEVX4t0P24BCEKA36ujnp-?usp=sharing>
  + <https://www.youtube.com/watch?v=LKB5I12LctU&list=PL3by7evD3F53Dz2RiB47Ztp9l_piGVuus>
* Sumanth suggested starting with the Instruction types

**11/16**

* A short meeting to discuss code structure
  + Creating a typedef enum for the opcodes
  + Creating a struct for the “idata” variable
* Assigned Instruction Blocks to each person
  + R\_type - Sumanth
  + I\_type - Sumanth
  + L \_type - Vijay
  + S\_type - Shruti
  + B\_type - Kai
* Each Block should be tested with a small testbench before merging into CPU module
* Plans to create a GitHub Repo for the project
  + <https://github.com/KaiRoy/ece571-riscv-sv-conversion>
* General Notes on Structure:

Packed Struct {

Logic [20:0] var

Logic [4:0] rd

Optype opcode //[6:0]

}

Typedef enum logic[6:0] {

Rtype = 7'b0110011

…

}

Aka Rd = idata.rd

**11/19**

* Shruti - I have completed the conversion for instruction S\_type
  + I have tested it with the testbench pushed code to Git.
  + I am working on optimizing the code further.

**11/20**

* Kai - Converted the B\_type instructions

**11/26**

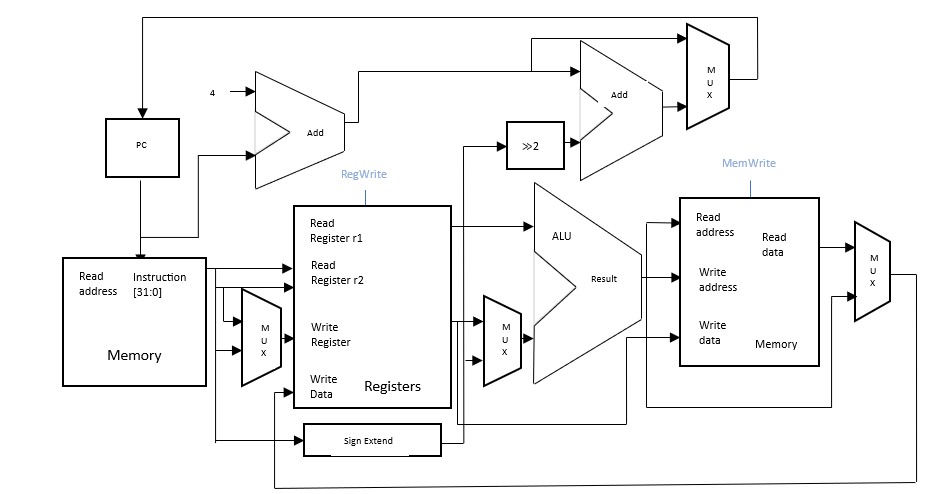
* Sumanth - I have completed the conversion for R & I Instruction set into System Verilog
* Given all test vectors to verify the instruction set design
* Uploaded required docs in Github
* Created the Interface files to the main CPU module
* Tested the interface module with all instances in CPU design
* We need to test the main source code with all involved sets and files.
* Vijay - I have completed the Conversion for Instruction L\_type
* Tested the Load type instruction set code.
* Uploaded the files to Git and committed it.
* Reading and understanding of the architecture to optimize further.
* Working on creating the Schematic of the L type(load) instruction fetching.

**Reference Links:**

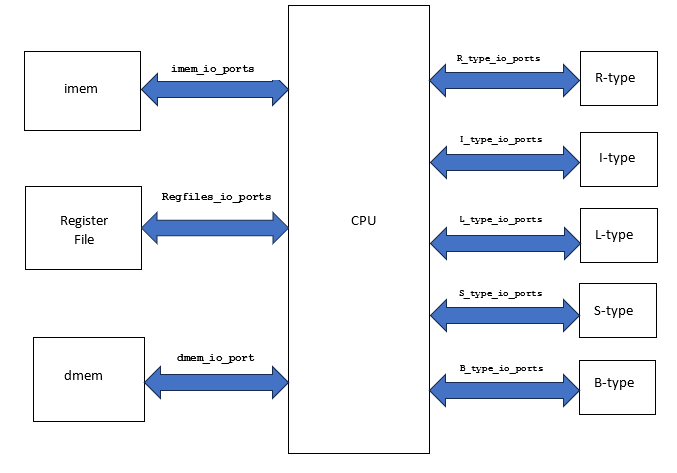
[emil-risc-v/Single\_Cycle\_CPU/Single\_cycle\_implementation at master · emilbiju/emil-risc-v · GitHub](https://github.com/emilbiju/emil-risc-v/tree/master/Single_Cycle_CPU/Single_cycle_implementation)

* **Shruti & Vijay:** Worked on block diagram based on inputs from everyone.

RISC-V Architecture for single-cycle

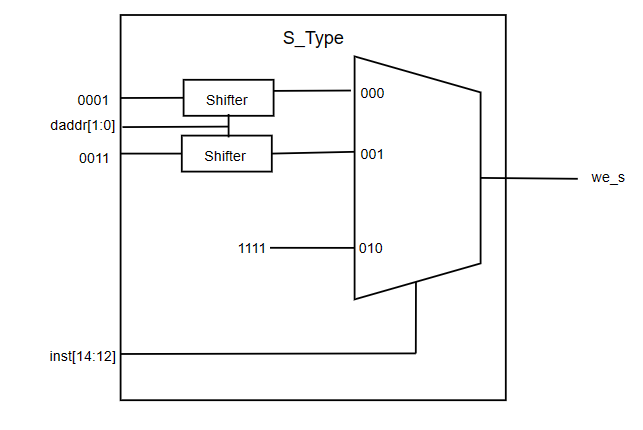


High-Level of Abstraction

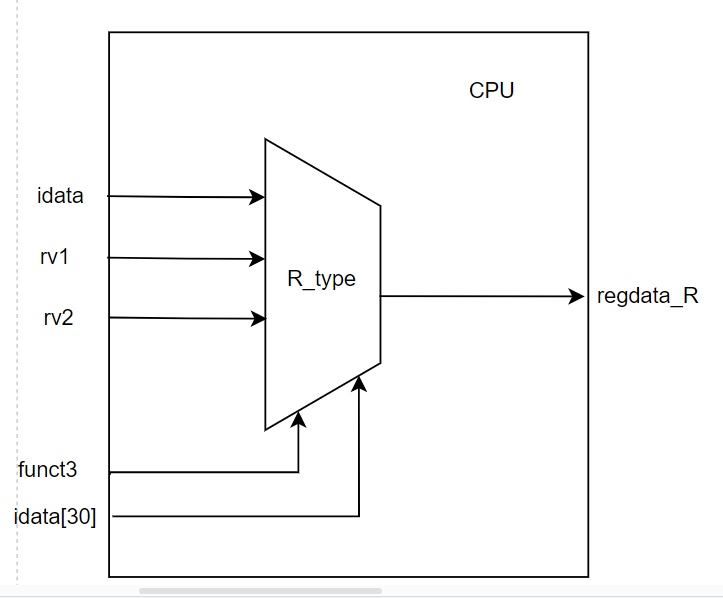


Individual Instruction Schematics:

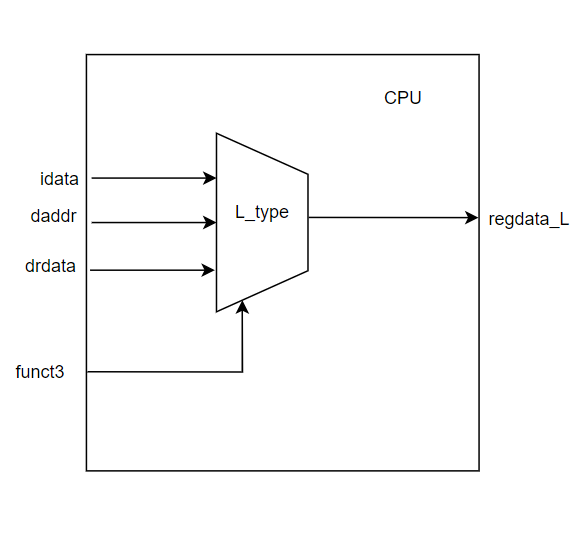
S\_Type:



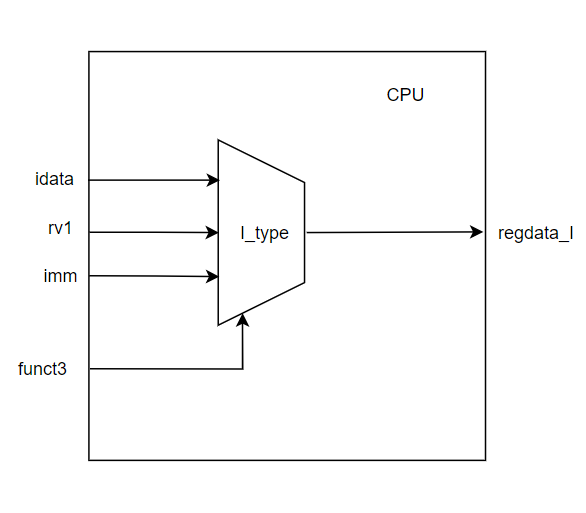
R-Type:



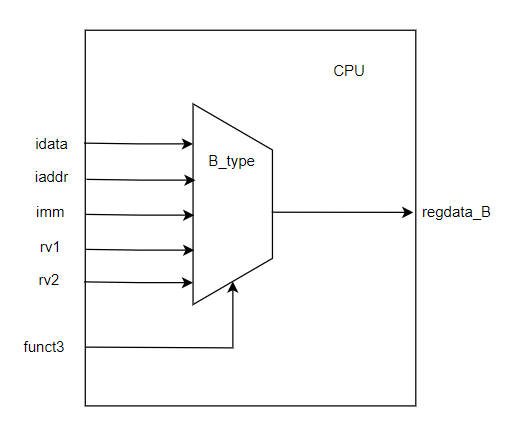
L\_Type:



I\_Type:



B\_Type:



**12/1:**

* Testing imem initialization from file (Kai, Sumanth)
  + Encountered Errors from file formatting
  + Easily fixed
* Testing the CPU module and its connections with the individual blocks (Kai, Sumanth)
  + Was able to connect to Branch block correctly
    - Many errors occurred due to timing and syntax.
    - Initial did not simulate correctly, changed timing to 1ns/1ns
  + Currently only runs one instruction for some reason
    - Internal clk was always ‘x’
    - Source was the regfile module, where the local clk variable was overriding the interface clk. (
      * **Error:** assign regfile.clk = clk
      * **Fixed:** assign clk = regfile.clk
* Testing R\_Type Instruction (Kai, Sumanth)
  + Issue with the regfile not saving data
    - Cause: previous fix to regfile, interface name is reg\_if not regal
    - Fix: assign clk = reg\_if.clk
* Testing L\_Type Instruction (Kai)
  + No Issues
* Testing S\_Type Instruction (Kai)
  + **Issue:** Only the first byte of a word would store correctly
  + **Cause:** Series of if/else was being used when only a series of if statements was needed

if(we[0]==1)

m[add0] <= dwdata[7:0];

else if(we[1]==1)

m[add1] <= dwdata[15:8];

else if(we[2]==1)

m[add2] <= dwdata[23:16];

else if(we[3]==1)

m[add3] <= dwdata[31:24];

* + **Fix:**

if(we[0]==1)

m[add0] <= dwdata[7:0];

if(we[1]==1)

m[add1] <= dwdata[15:8];

if(we[2]==1)

m[add2] <= dwdata[23:16];

if(we[3]==1)

m[add3] <= dwdata[31:24];

* Testing I\_Type Instruction (Kai)
  + No Issues
* All modules function as expected
  + Only JALR, JAL, LUI, AUIPC remain untested.

5th December:

* Started working on PPT and made the initial draft of PPT.