

Lab 3: Single Cycle Implementation of RISC-V (part 1)

Objectives:

The objective of this lab and the incoming set of labs is to model and simulate all the components of the RISC-V processor as a step towards completing its single cycle implementation

This lab is organized as follows:

1. An introduction
2. Experiments to be conducted
3. Lab Report Requirements

Introduction:

As studied in the lectures, RISC-V can be implemented such that each instruction is executed in a single clock cycle as shown in Figure 1. As in the lectures we will focus on supporting the following RV32I ISA subset:

- Memory reference: LW (I-Format), SW (S-Format)
- Arithmetic/logical: ADD, SUB, AND, OR (R-Format)
- Control transfer: BEQ (SB-Format)

The goal of this lab is to model and simulate some of the non-memory components necessary for this implementation.

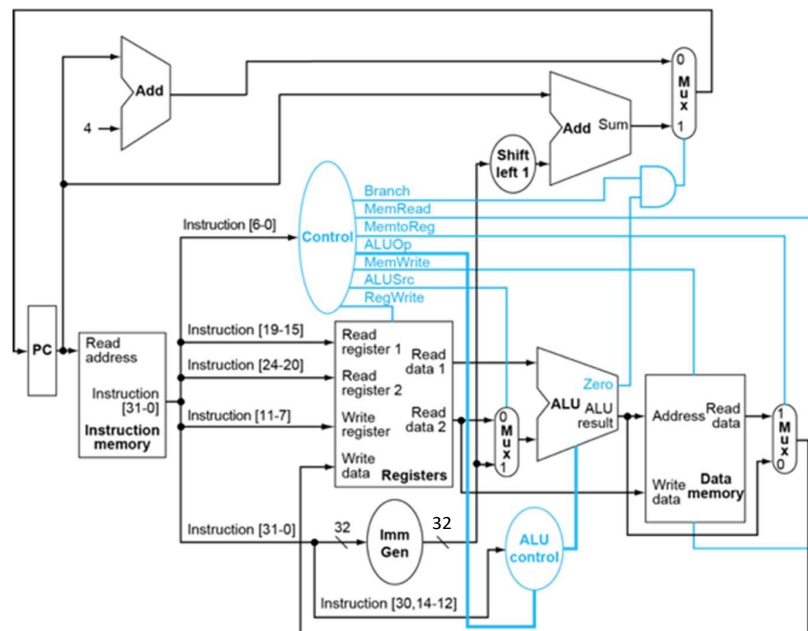


Figure 1 RISC-V Single Cycle Datapath

Table 1 shows the instruction encoding for the RV32I instructions highlighting the instructions we will implement in this lab. Note that:

- For our subset, the **7-bit opcode** field is enough to decode the instruction except for R-Format instructions (ADD, SUB, AND, OR) which all have the same value (**0110011**) for that field. For R-Format instructions, the actual operation is determined by the funct3 and funct7 fields (only bit 30 from the funct7 field is needed).
- The least significant 2 bits of the 7-bit opcode field are always 11 and as such can be excluded from any decoding circuit inputs.

Table 1 RV32I instructions encoding

imm[31:12]				rd	0110111	LUI	
imm[31:12]				rd	0010111	AUIPC	
imm[20:10:1 11:19:12]				rd	1101111	JAL	
imm[11:0]				rd	1100111	JALR	
imm[12:10:5]	rs2	rs1	000	imm[4:1:11]	1100011	BEQ	
imm[12:10:5]	rs2	rs1	001	imm[4:1:11]	1100011	BNE	
imm[12:10:5]	rs2	rs1	100	imm[4:1:11]	1100011	BLT	
imm[12:10:5]	rs2	rs1	101	imm[4:1:11]	1100011	BGE	
imm[12:10:5]	rs2	rs1	110	imm[4:1:11]	1100011	BLTU	
imm[12:10:5]	rs2	rs1	111	imm[4:1:11]	1100011	BGEU	
imm[11:0]		rs1	000	rd	0000011	LB	
imm[11:0]		rs1	001	rd	0000011	LH	
imm[11:0]		rs1	010	rd	0000011	LW	
imm[11:0]		rs1	100	rd	0000011	LBU	
imm[11:0]		rs1	101	rd	0000011	LHU	
imm[11:5]		rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]		rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]		rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI	
imm[11:0]		rs1	010	rd	0010011	SLTI	
imm[11:0]		rs1	011	rd	0010011	SLTIU	
imm[11:0]		rs1	100	rd	0010011	XORI	
imm[11:0]		rs1	110	rd	0010011	ORI	
imm[11:0]		rs1	111	rd	0010011	ANDI	
0000000	shamt	rs1	001	rd	0010011	SLLI	
0000000	shamt	rs1	101	rd	0010011	SRLI	
0100000	shamt	rs1	101	rd	0010011	SRAI	
0000000	rs2	rs1	000	rd	0110011	ADD	
0100000	rs2	rs1	000	rd	0110011	SUB	
0000000	rs2	rs1	001	rd	0110011	AND	
0000000	rs2	rs1	010	rd	0110011	SLT	
0000000	rs2	rs1	011	rd	0110011	SLTU	
0000000	rs2	rs1	100	rd	0110011	XOR	
0000000	rs2	rs1	101	rd	0110011	SRL	
0100000	rs2	rs1	101	rd	0110011	SRA	
0000000	rs2	rs1	110	rd	0110011	OR	
0000000	rs2	rs1	111	rd	0110011	AND	

In this lab we are going to model and test the following components:

1. A 32-bit register with reset and load control (to be used for PC and to construct the register file)
2. A 32-bit 2x1 multiplexer
3. A 32-bit shift left 1 module
4. The immediate generator & sign extender module

Note: Students should be working in pairs to implement the following experiments
Experiments:

Experiment 1: A N-bit register with reset and load control

Create and simulate a module representing a n-bit register with load control like the one shown in Figure 2. N is a generic parameter to have a default value of 8. Use flip flop modules, 2x1 multiplexers (implemented as shown in Figure 3). To avoid repeating your code 8 times, use a generate loop.

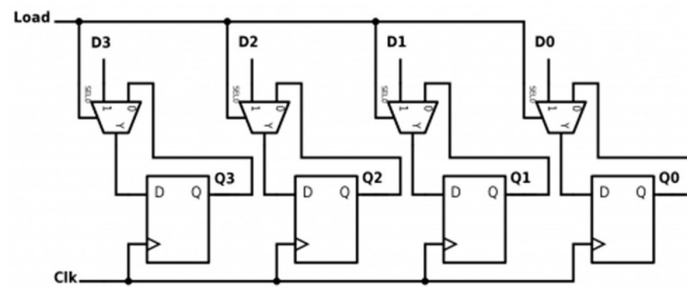


Figure 2 Register with load control

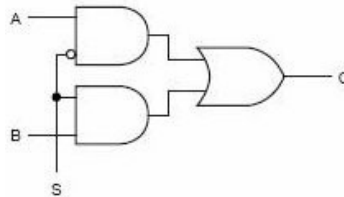


Figure 3 2x1 Multiplexer

You can define the flip flop module with reset (not shown in the above diagram) as follows:

```
module DFlipFlop
    (input clk, input rst, input D, output reg Q);

    always @ (posedge clk or posedge rst)
    if (rst) begin
        Q <= 1'b0;
    end else begin
        Q <= D;
    end
end
endmodule
```

Experiment 2: A n-bit 2x1 multiplexer

Create and simulate a module representing a n-bit 2x1 multiplexer. The module's structure should be an array of 2x1 multiplexers. Again use generate loop to do it.

Experiment 3: A n-bit shift left 1 module

Create and simulate a module representing a n-bit shift left 1 module. Shifting should be done by rewiring inputs. **(Don't use shift operator)**

Experiment 4: The immediate generator & sign extender module

This module has to take into consideration different instruction formats to be able to generate the appropriate immediate constant. Figure 4 shows the instruction formats for all supported instructions with immediate operands. LW uses the I-Format, SW uses the S-Format, while BEQ uses the SB-Format. So an immediate generator should receive the entire instruction to decide which 12 bits to use (based on the opcode and funct3 bits if needed). More specifically, the immediate generator must choose between sign extending a 12-bit field in instruction bits 31:20 for LW, a concatenation of bits 31:25 and 11:7 for SW, or a concatenation of bits 31, 7, 30:25, and 11:8 for BEQ. It turns out that opcode bit 6 is equal to 0 for LW and SW and equal to 1 for BEQ. Also opcode bit 5 is 0 for LW and 1 for SW. So the immediate generator can rely on these 2 bits to decide how to form the 12 bits to extend.

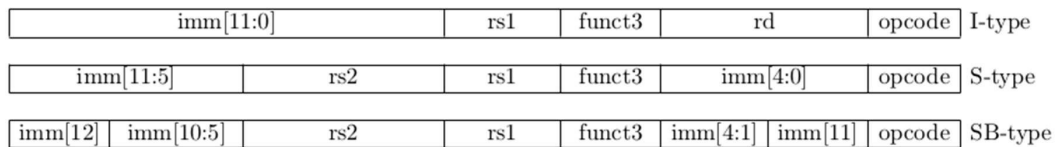


Figure 4 Instruction Formats for Supported Instructions with Immediate Operands

Here is the immediate generator module skeleton, that you should complete and simulate.

```
module ImmGen (output [31:0] gen_out, input [31:0] inst);  
    ...  
    ...  
    ...  
endmodule
```

Lab Report Requirements

Each student is required to submit an individual report.

Report should include:

1. [0 pts] Your name and student ID, team Name.
2. [2 pts] A technical summary of experiments conducted in the lab (steps, results, components, code functionality, etc.)
3. [3 pts] Verilog code for all modules of each experiment.
4. [2 pts] Verilog code for the testbench module of each experiment's top-level module.
5. [2 pts] Snapshot of simulation output corresponding to the submitted testbench with a **brief interpretation** of the snapshot
6. [1 pts] Draw a schematic design to our N-Bit register to allow extra input Shift, so the register would have **reset, load and shift control**. This extra input if = 1 it shifts the data

in the register 1 bit to the left. (you can draw it by hand and scan it, just make sure it is clear enough), if both shift and load = 1, load will have higher priority than shift and the load operation is carried on not the shift.