Labs 5: Single Cycle Implementation of RISC-V (part 3)

Objectives:

The objective of this lab is to complete the implementation of the RISC-V single cycle datapath and to test it on the Nexys A7 trainer board.

This lab is organized as follows:

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

Introduction:

In the previous labs, you should have completed the Verilog modeling and simulation of all non-memory components shown in Figure 1 assuming that we only need to support 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 the memory components (both the instruction memory and the data memory) and to construct the entire datapath out of previously modelled components.

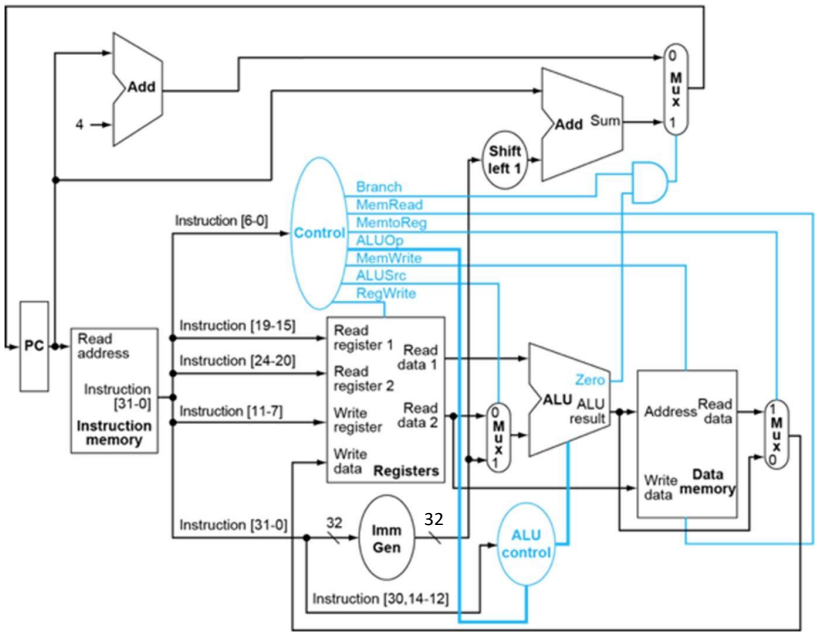

Figure 1 RISC-V Single Cycle Datapath

Table 1 shows the instruction encoding for the RV32I instructions highlighting the supported instructions:

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		rs1	000	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	SLL
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 conduct the following experiments:

1. Instruction Memory Modelling and Simulation
2. Data Memory Modelling and Simulation

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

Experiments:

Experiment 1: Instruction Memory Modelling and Simulation

RISC-V processor has a byte addressable instruction memory with a maximum capacity of 1 Giga Words (4 GBytes) with 32 address bits. Figure 2 shows the RISC-V instruction memory. This is not suitable for our experimental setup since we cannot implement this amount of memory on the FPGA board. Also, since there is an implicit assumption that only entire words can be read out of the instruction memory, we will implement **a word addressable** instruction memory with a maximum capacity of 64 words (256 bytes) with 6 address bits; however, since the PC contains the byte address of the instruction to be executed, we must divide it by 4 (discard

the least significant 2 bits) before connecting it to the read address input of the instruction memory to convert it to a word address.

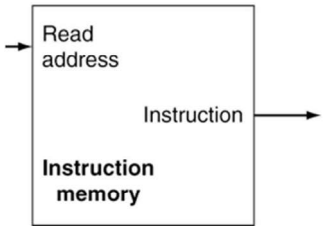

Figure 2 RISC-V Instruction Memory

The instruction memory module can be defined as follows:

```
module InstMem (input [5:0] addr, output [31:0] data_out);
    reg [31:0] mem [0:63];
    assign  data_out = mem[addr];
endmodule
```

Define a testbench module to simulate the instruction memory you defined. Since the instruction memory as defined above is essentially a read only memory (ROM), you will need to initialize some of the entries of the memory to test it. You can do so by adding an initial block inside the InstMem module.

Your testbench needs to include at least 5 test cases and verification codes.

Experiment 2: Data Memory Modelling and Simulation

Figure 3 shows the RISC-V data memory. For similar reasons to the instruction memory, we will implement **a word addressable** data memory with a maximum capacity of 64 words (256 bytes) with 6 address bits. Similarly, since the ALU computes the byte address of the data item to be loaded or stored, we must divide it by 4 (discard the least significant 2 bits) before connecting it to the address input of the data memory to convert it to a word address. Please note that the data memory also has a clock input that is not show in the diagram.

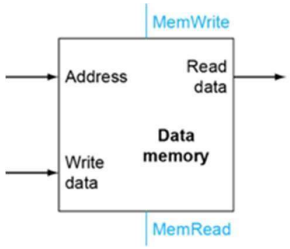

Figure 3 RISC-V Data Memory

The data memory module can be defined as follows:

```
module DataMem
    (input clk, input MemRead, input MemWrite,
    input [5:0] addr, input [31:0] data_in, output [31:0] data_out);

    reg [31:0] mem [0:63];

       ……

endmodule
```

Define a testbench module to simulate the data memory you defined. Please note that the data memory allows for **asynchronous reading** and **synchronous writing** and can be tested without being initialized. **(why?, what does asynchronous read means?)**

Lab5 Report Requirements

Report of Lab5 should include:

1. [0 pts] Your name and student ID.
2. [2 pts] A technical summary of experiments 1 and 2 as conducted in the lab (steps, results, components, code functionality, etc.)
3. [4 pts] All Verilog code written (including testbenches) for experiments 1 and 2
4. [4 pts] Snapshot of simulation output corresponding to each submitted testbench with a **brief interpretation** of the snapshot