Jordan National Semiconductor Design Competition (JOSDC'2023)

Theme: Design a RISC CPU using FPGAs.

Goal: Building a single-cycle RISC processor on Field Programmable Gate Array (FPGA) that supports MIPS ISA. Participants should build a processor using a subset of MIPS ISA as shown in table 1. The designed processor should be emulated on FPGA and Simulated on ModelSim.

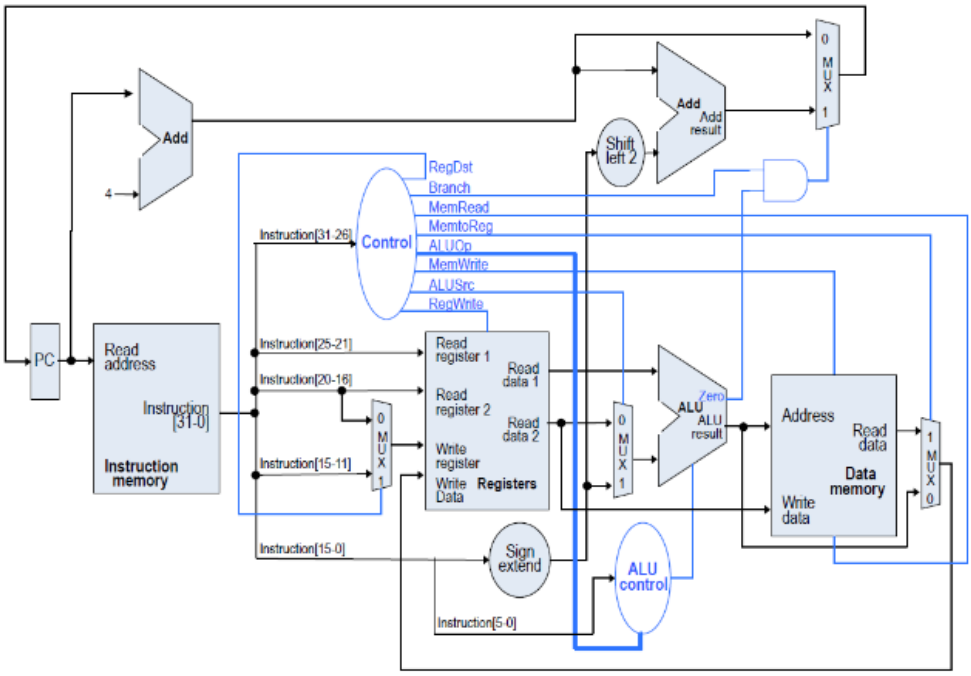

Figure 1: Single-Cycle MIPS Processor

Components Description:

Participants should design a processor consisting of the following components:

- Instruction memory (read-only)
    - The PC will be incremented by 1 every clock cycle unless there is a JMP or Branch instruction.
- Data memory (Read/Write)
    - Should be synchronized with the reference clock to handle Load/Store instructions properly.
- Register File:
    - The MIPS ISA defines 32 32-bit general purpose registers (GPR).

    - Must be able to read two values simultaneously, and asynchronously (no clock required for reading).
    - Writes must happen only on the positive clock edge and only when the write enable is asserted.
    - Correctly handles reads and writes from register zero.
- Arithmetic and Logic Unit (ALU)
    - This is where all computations will be executed. Also, it should include the control unit to handle the execution process accordingly.

The project must have a clear description of all components, datapath, and control signals as well as their functionality. You may need to modify some parts of your datapath, as well as expand your control unit. Depending on your implementation, you may need to modify your ALU. If the instruction ROM doesn't fit the program you are running you may need to increase the address width.
All blocks should be implemented in Verilog, should be synthesizable, and each unit implemented separately (separate modules). By the end of this competition, your processor should be able to perform the following basic instructions: **LW, SW, ADD, ADDI, SUB, AND, OR, NOR, XOR.** In addition, the processor should include support for branch and jump instructions listed in Table 1.

Table 1: MIPS Subset Instructions

Branches & Jumps	Arithmetic & Logic	Load & Store
BEQ	ADD	LW
BNE	SUB	SW
BLT	AND, ANDI,	
BGE	OR ORI	
BLE	XOR	
BGT	NOR	
JUMP	Shift Right Logical	
JR	Shift Left Logical	
JAL	ADDI, ADDU	
	SUBU	

Bonus Task:

- Design a 5-stage pipeline processor: adding pipeline stages will speed up your single-cycle MIPS processor. For this competition, you will be adding 5 pipeline stages to your design. You should be able to show that your processor runs faster and simulate your design. It is recommended to add the pipeline stages one by one and measure the processor speed each time.

- Floating Point Unit: You need to provide a full description of this unit and simulate your design.
- Design an out-of-order processor: You should be able to show that your processor runs faster and simulate your design.

Design Development Milestones:

The competition starts October 1st – December 24th. During the three months, you will implement a simple processor described above in addition to the bonus tasks of your choice. You need to create all components listed in this document and create a top-level entity (e.g. proc.v).

First month's expectation:

During the first month, you are expected to have a clear idea of your project development:

- Start working on your report template and build it concurrently as you build your design.
- Divide the project into subtasks and make sure all members of your team have a well-defined role based on their skills, level, and background. Please note that this is the team lead's responsibility to keep the team in sync.
- As a first step, you should implement the single-cycle MIPS CPU Datapath. As in Figure 2.
- Make sure you have a clear description of each module.
- Show me that your register file is working.

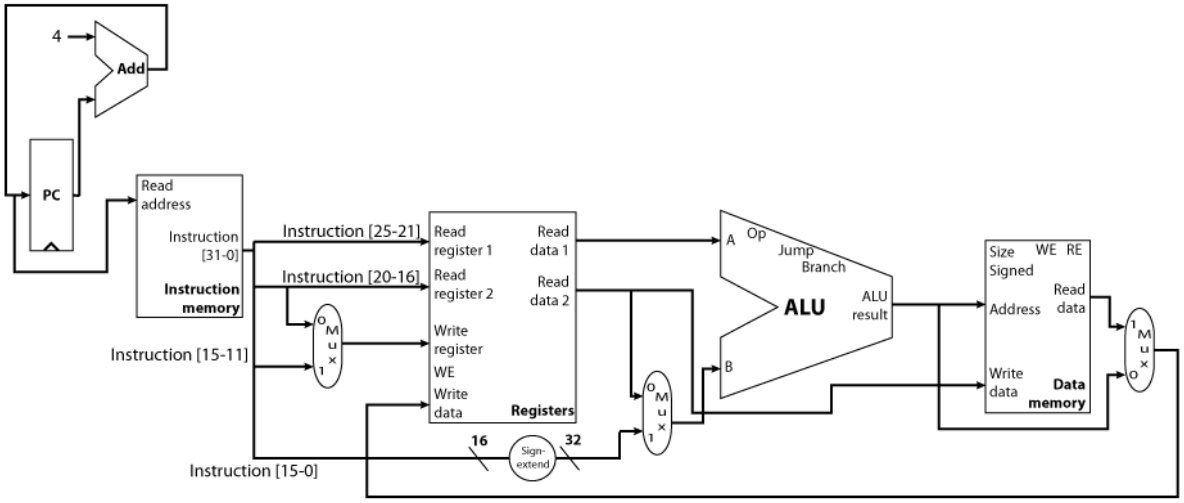

Figure 2 Single-cycle CPU Datapath

- Now you need to implement the control unit for the Single-Cycle MIPS processor. The processor you built so far should be able to run arithmetic, logical, and load/store operations (refer to Table 1 above regarding the subset of instructions you need to implement).
- Since you have your control unit ready, write a benchmark that tests a single instruction. Verify it is working correctly using Modelsim and then verify the rest of the instructions. Remember, you don't have branch instructions implementation yet, so focus on testing the arithmetic, logical, and load/store operations.
- Simulate your design and show us the waveform with all signals.
- Your Verilog code should be synthesizable.
- Submit a report describing your progress for this month.

Second month's expectation:

- Continue working on your report.
- To complete the second milestone of this project, you might need to modify many parts of your datapath, and expand your control unit. You might also need to modify your ALU. You will need to create new modules.
- You are required to implement the "Branches & jumps" instructions listed in the table above.
- Show your schematic design.
- Make sure your processor can run the same benchmark from the first milestone as well as the new benchmark that tests branches.
- Your Verilog code should be synthesizable. You should be able to simulate your design as well.
- Submit a report describing your progress for this month.

Third month's expectation:

- The month of innovations. You are expected to provide a detailed report of your bonus tasks as well as a set of benchmarks for testing. It is your responsibility to show your work in detail.
- Submit a final report describing your progress for this month.