BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

ENCS4370| Computer Architecture - Project No. 2

Fall Semester 2025/2026

Design and Verification of a Simplified Predicated RISC Processor using Verilog

Deadline: January 2, 2026 at 23:59

---

1. Objectives

- Understand RISC processor architecture and instruction set design
- Design the datapath and control path of a simple 32-bit predicated RISC processor
- Implement the designed RISC processor in Verilog
- Verify the processor functionality through simulation

2. Processor Specifications

1. The instruction size and the word size is 32 bits
2. 32 32-bit general-purpose registers from R0 to R31
3. R0 is hardwired to zero
4. R30 is hardwired to be the PC (Program Counter)
5. R31 is hardwired to the return address register.
6. The processor has two separate memories, namely, instruction and data memory.
7. Word addressable memory
8. The processor supports predicated execution, where a predicate register **Rp** determines whether an instruction executes. Rp can be any general-purpose register. If the content of Rp is zero, the instruction is not executed. However, if the content of Rp is non-zero, the instruction executes. To execute an instruction unconditionally, the programmer sets Rp to R0.

9. The processor supports three types of instructions:

- Register Type (R-Type)

$Opcode^5$	Rp^5	Rd^5	Rs^5	Rt^5	$Unused^7$

- Immediate Type (I-Type)

$Opcode^5$	Rp^5	Rd^5	Rs^5	$Immediate^{12}$

- Jump Type (J-Type)

$Opcode^5$	Rp^5	$Offset^{22}$

10. Instruction Encoding

The table below lists the instructions to be implemented, including their opcode values and Register Transfer Notation (RTN). This set is sufficient to write meaningful programs.

Instruction	Type	Meaning	Opcode
ADD Rd, Rs, Rt, Rp	R-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] + Reg[Rt]	0
SUB Rd, Rs, Rt, Rp	R-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] - Reg[Rt]	1
OR Rd, Rs, Rt, Rp	R-Type	if (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] | Reg[Rt]	2
NOR Rd, Rs, Rt, Rp	R-Type	if (Reg[Rp] ≠ 0) Reg[Rd] = ~(Reg[Rs] | Reg[Rt])	3
AND Rd, Rs, Rt, Rp	R-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] & Reg[Rt]	4
ADDI Rd, Rs, Imm, Rp	I-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] + Imm	5
ORI Rd, Rs, Imm, Rp	I-Type	if (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] | Imm	6
NORI Rd, Rs, Imm, Rp	I-Type	if (Reg[Rp] ≠ 0) Reg[Rd] = ~(Reg[Rs] | Imm)	7
ANDI Rd, Rs, Imm, Rp	I-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Reg[Rs] & Imm	9
LW Rd, Imm(Rs), Rp	I-Type	If (Reg[Rp] ≠ 0) Reg[Rd] = Mem(Reg[Rs] + Imm)	10
SW Rd, Imm(Rs), Rp	I-Type	If (Reg[Rp] ≠ 0) Mem(Reg[Rs] + Imm) = Reg[Rd]	11
J Label, Rp	J-Type	If (Reg[Rp] ≠ 0), jump to the target address	12
CALL Label, Rp	J-Type	If (Reg[Rp] ≠ 0) , jump to the function, store the return address in register R31	13
JR, Rs, Rp	R-Type	If (Reg[Rp] ≠ 0) , jump to the address in register Rs	14

11. The jump target address is computed by adding the sing-extended instruction's offset field to the current value of the PC.
12. The immediate value is zero-extended for logical instructions, and sign-extended for all other instruction types.
13. All negative numbers are represented in two's-complement form.

3. RTL Design

You are required to design and implement the given processor in Verilog, consisting of the following stages: fetch, decode, execute (ALU), memory access, and write-back. Your design must include both a datapath and a control path that support all of the specified instructions.

4. Design Verification

To verify the RTL design, develop a comprehensive testbench for simulation-based verification of your processor. Additionally, create multiple binary test programs using the provided ISA to demonstrate the correct execution of instructions. These programs should cover all supported instructions and processor features.

5. Project Report

The report must be written in an organized way, and it should include the following items:

1. Detailed description of the datapath, its components, and the assembly of these components.
2. A complete description of the control signals, truth table, state diagrams, Boolean equations, logic diagrams, etc.
3. High quality Datapath and control path diagrams
4. The implementation details, and the design choices you made with justification
5. A list of sources for any parts of your design and/or code that are not yours (if any).
6. Test programs in assembly language and binary format with detailed description of these programs
7. Simulation results and screenshots
8. **Disclosure of any AI tools usage. Clearly state if you used any AI tools in the design, coding, or report.**

6. Teamwork

1. Work in teams of up to three students. Teams of more than three students are not allowed.
2. Team members are required to coordinate the work equally among themselves so that everyone is involved in all the following activities:
   - Design and implementation
   - Simulation and testing
   - Report writing
   - Project demonstration and discussion
3. Clearly show the work done by each team member.
4. The team members can be from different sections.

7. Submission Guidelines

Please attach a single ZIP file containing the project source code and the report document.

8. Grading Criteria

Each of the following items is a prerequisite to the next one, e.g., we cannot consider the RTL code if there is no design, and we cannot consider the verification part if there is no RTL code, and so on.

Item	**Grade**
Control signals generation (truth tables, Boolean equations, states diagrams, logic diagrams, etc.)	10
Processor Microarchitecture Design (complete datapath and control path) that supports all specified instructions	20
Complete **modular** RTL design of the above microarchitecture that supports all specified instructions	20
Verification environment (testbench) around the RTL design such that you can write code sequences in the ISA, store them in the processor's instruction memory, execute them and show results using waveform diagrams.	20
Code organization and documentation	5
Detailed report that includes control signals truth tables, Boolean equations, states diagrams, detailed and clear microarchitecture design schematics, test cases, simulation screenshots, etc.	15
Discussion (answering questions, the way of answering questions, etc.)	10
Total	**100**