جامعة بيرزيت
BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

ENCS4370| Computer Architecture

2024/2025

Spring Semester

Project No. 2

Design and Verification of a Simple Pipelined RISC Processor in Verilog

Deadline: May 25, 2025 at 23:59

---

1. Objectives

- To design the datapath and control path of a simple 32-bit pipelined RISC processor in Verilog
- To implement and verify a working processor using simulation

2. Processor Specifications

1. The instruction size and the word size is 32 bits
2. 16 32-bit general-purpose registers from R0 to R15
3. R15 is hardwired to be the PC (Program Counter)
4. R14 is conventionally designated as the return address register.
5. The processor has two separate memories, namely, instruction and data memory.
6. Word addressable memory
7. The ISA has one instruction type only

3. The Instruction Format

Opcode^6	Rd^4	Rs^4	Rt^4	Imm^{14}

- 6-bit opcode
- **4-bit Rd:** Destination register
- **4-bit Rs:** First source register
- **4-bit Rt:** Second source register
- **14-bit Imm:** The immediate value, which is zero-extended for logical instructions and sign-extended otherwise
- For BZ, BLZ, BGZ, J, and CALL instructions, the target address is calculated by adding the current value of the PC to the sign extended immediate value.

4. Instruction Encoding

For simplicity, you are only required to implement a subset of the processor's Instruction Set Architecture (ISA). The table below lists the instructions to be implemented, including their opcode values and corresponding representations in Register Transfer Notation (RTN). Any unused instruction field is encoded as zero in the machine code

Instruction	Meaning	Opcode Value
OR Rd, Rs, Rt	Reg(Rd) = Reg(Rs) | Reg(Rt)	0
ADD Rd, Rs, Rt	Reg(Rd) = Reg(Rs) + Reg(Rt)	1
SUB Rd, Rs, Rt	Reg(Rd) = Reg(Rs) - Reg(Rt)	2
CMP Rd, Rs, Rt	Reg(Rd) = 0 if(Reg(Rs) == Reg(Rt)) Reg(Rd) = -1 if(Reg(Rs) < Reg(Rt)) Reg(Rd) = 1 if(Reg(Rs) > Reg(Rt))	3
ORI Rd, Rs, Imm	Reg(Rd) = Reg(Rs) | Imm	4
ADDI Rd, Rs, Imm	Reg(Rd) = Reg(Rs) + Imm	5
LW Rd, Imm(Rs)	Reg(Rd) = Mem(Reg(Rs) + Imm)	6
SW Rd, Imm(Rs)	Mem(Reg(Rs) + Imm) = Reg(Rd)	7
LDW Rd, Imm(Rs)	In the first clock cycle: • Reg(Rd) = Mem(Reg(Rs) + Imm) In the second clock cycle: • Reg(Rd + 1) = Mem(Reg(Rs) + Imm + 1) • LDW (load double word) loads two consecutive words from memory to two consecutive destination registers in two consecutive clock cycles. • The destination register number in the instruction must be an even number. • If the processor detects an LDW instruction with an odd destination register number, it will	8

	throw an exception • When the processor detects an LDW instruction in the decode unit, it will not bring a new instruction in the next clock cycle, because in the next cycle, the processor will handle the second word	
SDW Rd, Imm(Rs)	In the first clock cycle: • Reg(Rd) ➔ Mem(Reg(Rs) + Imm) In the second clock cycle: • Reg(Rd + 1) ➔ Mem(Reg(Rs) + Imm + 1) • SDW (store double word) stores the content of two consecutive source registers into two consecutive words in memory in two consecutive clock cycles. • The source register number in the instruction must be an even number. • If the processor detects an SDW instruction with an odd source register number, it will throw an exception • When the processor detects an SDW instruction in the decode unit, it will not bring a new instruction in the next cycle, because in the next cycle, the processor will handle the second word	9
BZ Rs, Label	if (Reg(Rs) == 0) Next PC = branch target else Next PC = PC + 1	10
BGZ Rs, Label	if (Reg(Rs) > 0) Next PC = branch target	11

	else Next PC = PC + 1	
BLZ Rs, Label	if (Reg(Rs) < 0) Next PC = branch target else Next PC = PC + 1	12
JR Rs	Jump to the target address stored in the register Rs	13
J Label	Unconditionally jump to the target address specified by the label	14
CLL Label	Jump to the function at the specified label and store the return address in register R14	15

5. RTL Design

You are required to design and implement a 5-stage pipelined 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.

6. 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.

7. 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. 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.

9. Submission Guidelines

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

10. 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.

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**