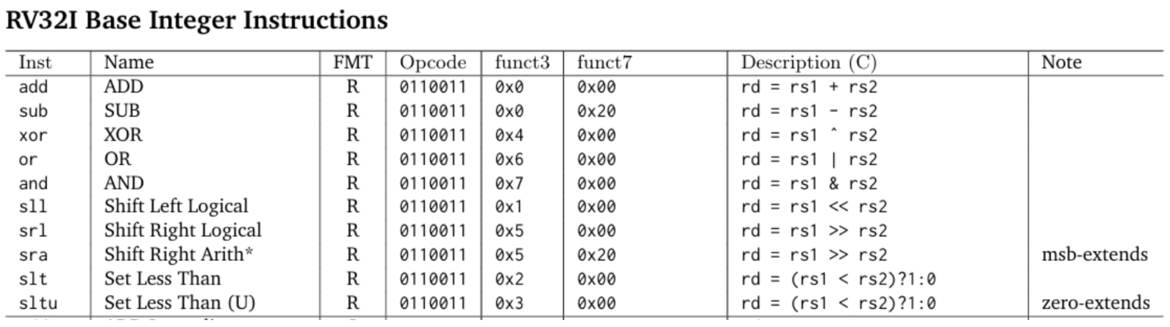
Problem Statement

Marks: 5M (automated evaluation) + 5M (viva). Please strictly follow the implementation regulations, as they will be verified during the viva.

Implement a **64-bit Arithmetic and Logical Unit (ALU)** that supports all required **R-type** instructions listed in Figure 1.



RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ˆ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends

Figure 1: Instructions to be implemented

Figure 1 provides the instruction set to be implemented along with their opcodes and relevant fields. In this assignment, **instruction decoding is not required**. Instead, the ALU will be tested by providing:

- two 64-bit input operands, and
- a 4-bit ALU control signal (raw or encoded using labels).

You are expected to follow a **modular design approach**. Implement each instruction/operation as a separate Verilog module along with its corresponding testbench. Finally, integrate all modules into a wrapper file named `alu.v`, which instantiates all relevant submodules and connects them appropriately.

Design Constraints:

- The implementation must be **structural** throughout.
- You are **not allowed** to directly use behavioral operators such as `+`, `-`, `&`, etc. on 64-bit inputs to implement the required 64-bit operations.
- All inputs and outputs must be treated as **signed**, and subtraction must follow **2's complement** arithmetic.
- For shift operations, it is recommended to use a **Barrel Shifter** based design.

Inputs:

- Two 64-bit operands
- 4-bit ALU control signal

Outputs:

- 64-bit result
- Status flags: overflow, carry, negative, and zero

NOTE:

- The assignment will be evaluated using a **custom testbench**.
- A **template testbench** is attached for reference.
- Your submission must be **compatible** with this testbench for validating final ALU functionality.
- Please use the **exact same variable names** for all `wire/reg` as specified in the template.
- The ALU wrapper design file must be named exactly as `alu.v`.