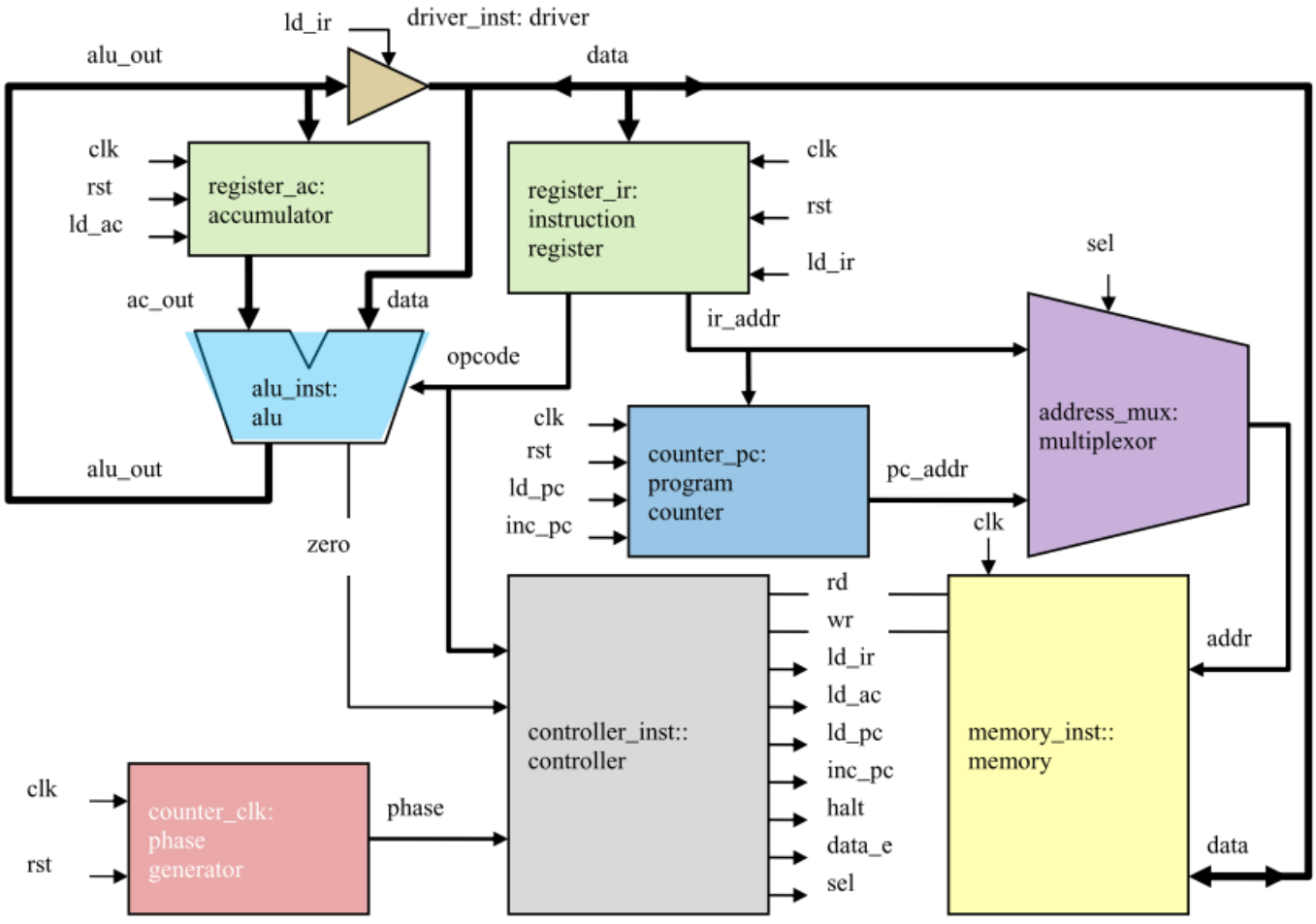
VeriRISC-CPU

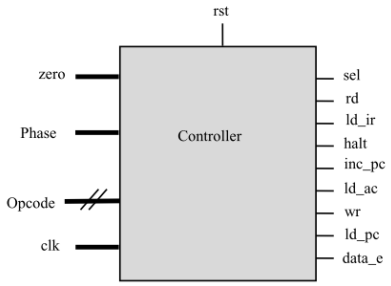
Modules Specifications

Veri_Risc:

The VeriRISC CPU is a minimalistic, reduced-instruction-set processor implemented in Verilog HDL. It features a streamlined architecture with a three-bit operation code and a five-bit operand, resulting in an instruction set limited to eight commands and an addressable space of 32 locations. This design facilitates ease of visualization and debugging, making it an ideal educational tool for learning about CPU architecture and digital design. The CPU operates based on a clock and a reset signal, and it outputs a halt signal to indicate the end of execution. The performance of the CPU is evaluated based on the number of clock cycles required to execute a program.



Modules Specifications:



1. **Controller Module**:

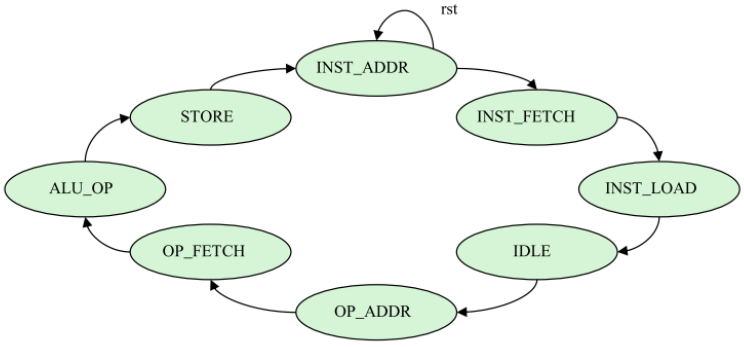
-The controller is clocked on the rising edge of clk.

-rst is synchronous and active high.

-zero is an input which is 1 when the CPU accumulator is zero and 0 otherwise.

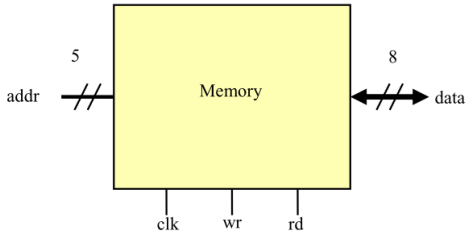
-opcode is a 3-bit input for CPU operation, as shown in the following table:

Opcode/ Instruction	Opcode Encoding	Operation	Output
HLT	000	PASS A	in_a => alu_out
SKZ	001	PASS A	in_a => alu_out
ADD	010	ADD	in_a + in_b => alu_out
AND	011	AND	in_a & in_b => alu_out
XOR	100	XOR	in_a ^ in_b => alu_out
LDA	101	PASS B	in_b => alu_out
STO	110	PASS A	in_a => alu_out
JMP	111	PASS A	in_a => alu_out



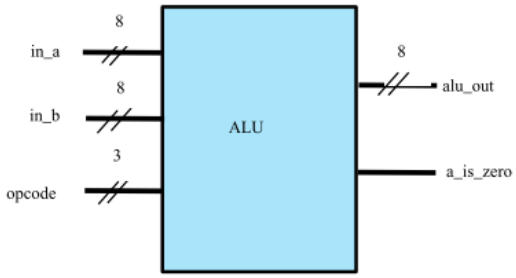
Outputs	Phase						Notes		
	INST_ ADDR	INST_ FETCH	INST_ LOAD	IDLE	OP_ ADDR	OP_ FETCH	ALU_ OP	STORE	
sel	1	1	1	1	0	0	0	0	ALU_OP = 1 if opcode is ADD, AND, XOR or LDA
rd	0	1	1	1	0	ALUOP	ALUOP	ALUOP	
ld_ir	0	0	1	1	0	0	0	0	
halt	0	0	0	0	HALT	0	0	0	
inc_pc	0	0	0	0	1	0	SKZ && zero	0	
ld_ac	0	0	0	0	0	0	0	ALUOP	
ld_pc	0	0	0	0	0	0	JMP	JMP	
wr	0	0	0	0	0	0	0	STO	
data_e	0	0	0	0	0	0	STO	STO	

2. **Memory Module**: Stores instructions and data. It supports read and write operations within the 32-location address space.



   - -addr is parameterized to 5 and data is parameterized to 8. -wr (write) and rd (read) are single-bit input signals.
   - -The memory is clocked on the rising edge of clk.
   - Memory write: data_in is written to memory[addr] on the positive edge of clk when wr (write) =1.
   - Memory read: data_out is assigned from memory[addr] when rd (read) =1.

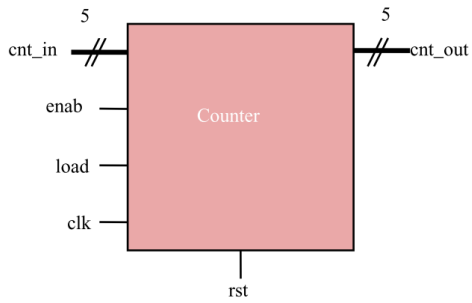
3. **ALU (Arithmetic Logic Unit)**: Executes arithmetic and logical operations based on the opcode and operand. It processes data and provides results for use in further operations.



   - in_a, in_b, and alu_out are all 8-bit long. The opcode is a 3-bit value for the CPU operation code, as defined in the following table.
   - a_is_zero is a single bit asynchronous output with a value of 1 when in_a equals 0. Otherwise, a_is_zero is 0.
   - The output alu_out value will depend on the opcode value as per the following table.
   - To select which of the 8 operations to perform, you will use opcode as the selection lines.
   - The following table states the opcode/instruction, opcode encoding, operation, and output:

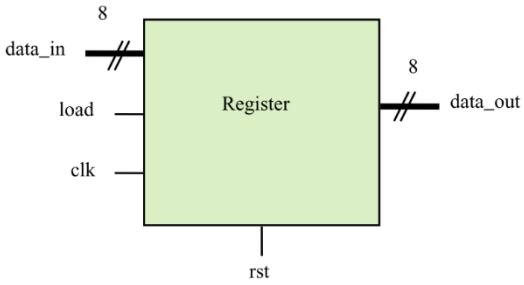
Opcode/ Instruction	Opcode Encoding	Operation	Output
HLT	000	PASS A	in_a => alu_out
SKZ	001	PASS A	in_a => alu_out
ADD	010	ADD	in_a + in_b => alu_out
AND	011	AND	in_a & in_b => alu_out
XOR	100	XOR	in_a ^ in_b => alu_out
LDA	101	PASS B	in_b => alu_out
STO	110	PASS A	in_a => alu_out
JMP	111	PASS A	in_a => alu_out

4. **Program Counter (PC)**: Keeps track of the address of the next instruction to be fetched. It is incremented to ensure sequential execution of instructions.



5. **Phase Generator**: Manages the timing and sequencing of the fetch-and-execute cycle. It generates control signals to synchronize the various phases of instruction processing.
   - The counter is clocked on the rising edge of clk.
   - rst is active high.
   - cnt_in and cnt_out are both 5-bit signals.
   - If rst is high, output will become zero.
   - If the load is high, the counter is loaded from the input cnt_in.
   - Otherwise, if enab is high, cnt_out is incremented, and cnt_out is unchanged.

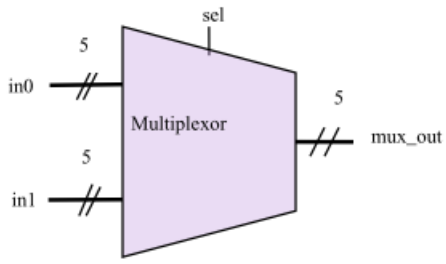
6. **Accumulator**: Holds intermediate results of operations performed by the ALU. It is essential for performing cumulative calculations and temporary data storage.



7. **Instruction Register**: Stores the current instruction fetched from memory. It ensures that the CPU operates on the correct instruction during each cycle.

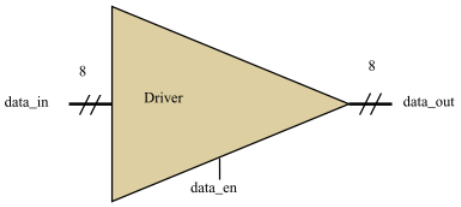
   - data_in and data_out are both 8-bit signals.
   - rst is synchronous and active high.
   - The register is clocked on the rising edge of clk.
   - If load is high, the input data is passed to the output data_out.
   - Otherwise, the current value of data_out is retained in the register.

8. **Multiplexer**: Directs the flow of data within the CPU. It selects between different data inputs based on control signals, ensuring that the correct data is routed to the ALU, memory, or other components.



   - The address multiplexor selects between the instruction address during the instruction fetch phase and the operand address during the instruction execution phase.
   - MUX width is parameterized with the default value of 5.
   - If sel is 1'b0, input in0 is passed to the output mux_out.
   - If sel is 1'b1, input in1 is passed to the output mux_out.

9. **Driver Module**: Interfaces with external components or systems, providing necessary signals and data for the CPU's operation.



   - data_in and data_out are both parameterized widths of 8-bit.
   - If data_en is high, the input data_in is passed to the output data_out.
   - Otherwise, data_out is high impedance.