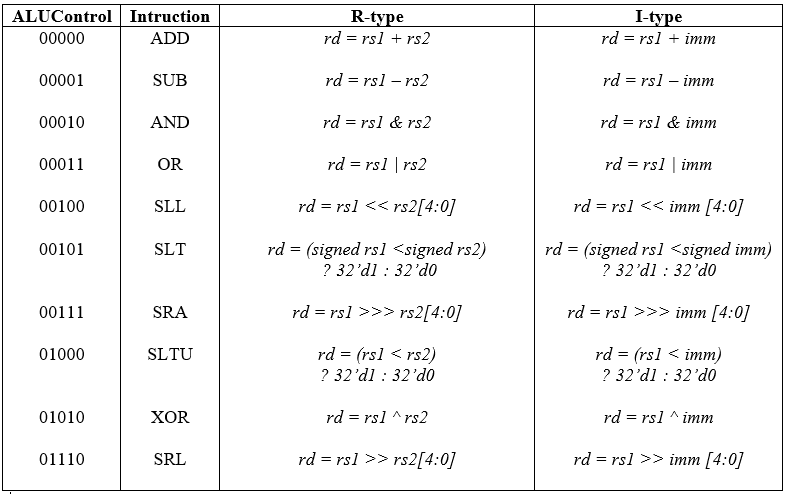
**3.1 Data path**

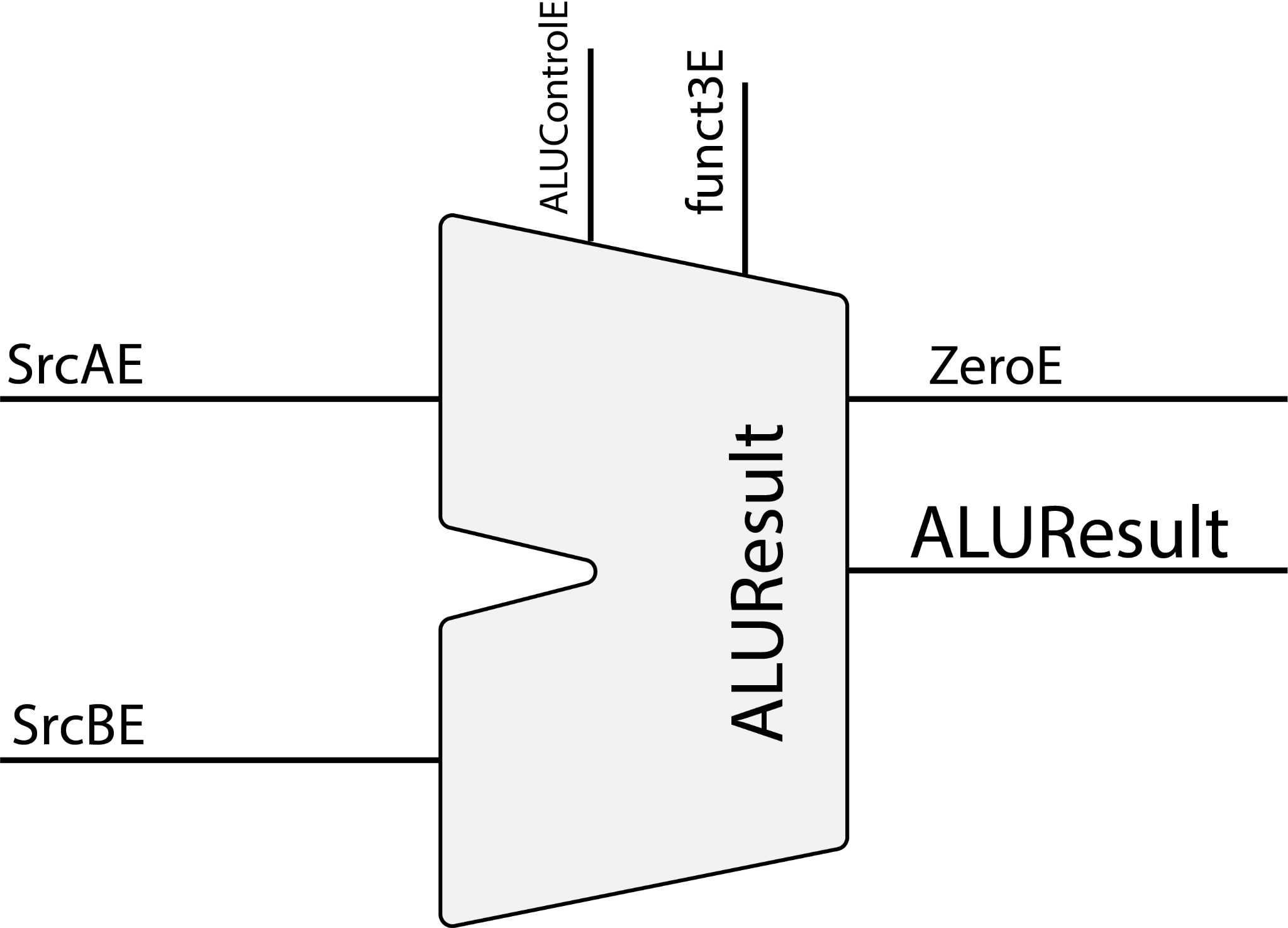
**3.1.1 Arithmetic Logic Unit**



**Table. Operations of RV32I ALU**

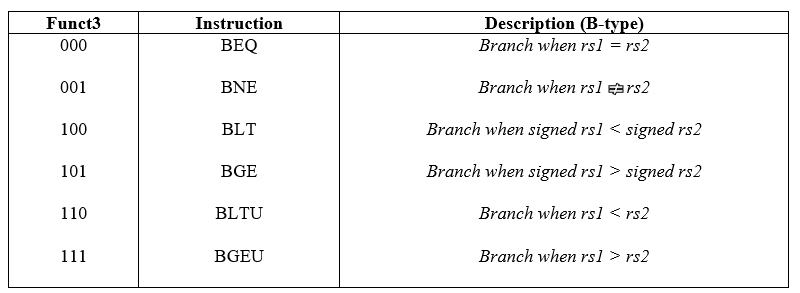
* Arithmetic Operation: The ALU is responsible for performing arithmetic operations, such as: addition, and subtraction.
* Logical Operation: The ALU can perform logical operations, including bitwise AND, OR, and XOR, which are important for manipulating data and making decisions in programs.
* Comparison Operations: The ALU is used to compare two values, determining if they are equal, greater than, less than, or not equal.
* Shift Operations: The ALU can shift and rotate bits within data, which is useful for tasks like shifting bits to the left or right.

Based on the information above, ALU design looks like:

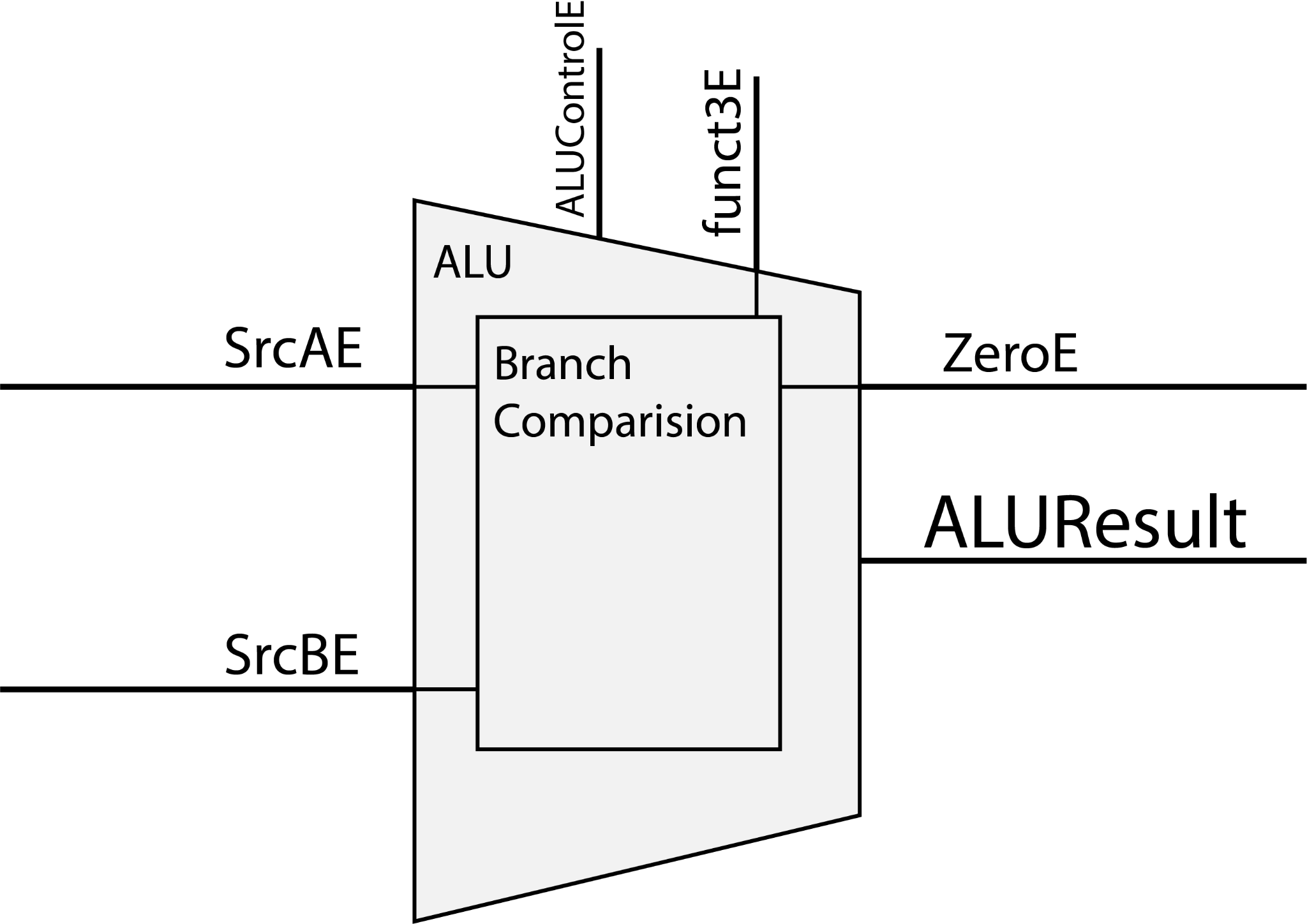


**3.1.2 Branch Comparison**

RISC- V processor has 6 conditional branch instructions (B-type), each of which take two source registers and a label indicating where to go. Different types of branches are distinguished by 3 bits of the funct3. Funct3 consists of bits 12 to 14 of the instruction. This is conditional branch instructions table below:

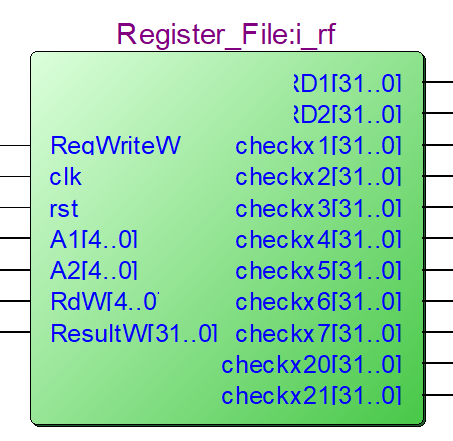


**Table. Branch Instructions**

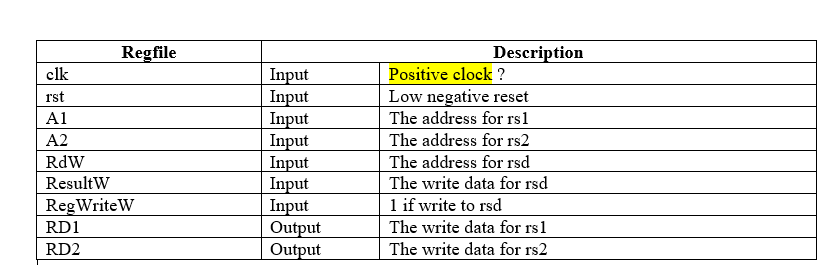
Based on the information above, Branch Comparison design looks like:

**3.1.3 Register file**

The used RF has two output ports and one writing port; these numbers are sufficient to avoid any structural hazard due to this component. The only special register is the one number 0. It is hardwired to zero, this ensures that it cannot be changed even after a program error, in this way there is always a register that can be used to store an immediate value in a register, to move a register into another, to use a direct addressing mode for a load/ store, or to perform others types of instruction or pseudo-instruction that need a zero in one of the two registers.



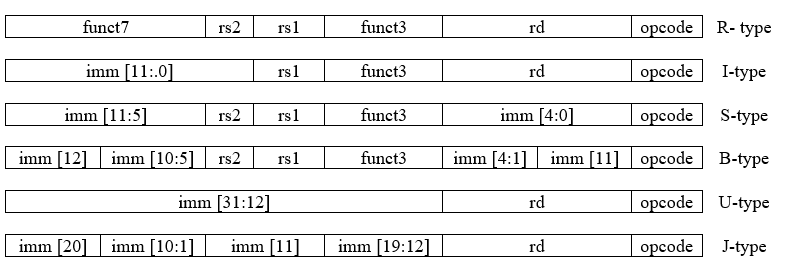
**Figure . RTL view of Regfile**



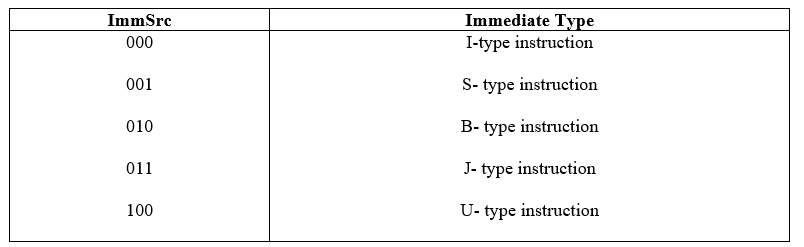
**3.1.4 Immediate Generator**

The immediate generator block in a RISC-V processor is responsible for generating immediate values used in instructions. Immediate values are constants or literal values that are directly specified within the instruction itself. The immediate generator block plays a crucial role in arithmetic, logical, and data manipulation instructions by providing the constant values that are operated upon. This block contributes to the efficiency and flexibility of RISC-V instructions, allowing for quick and direct specification of values within the instruction stream.

Here, project implement I-type, S-type, B-type, U-type and J-type immediate of R32IV, and the specific structure of immediate types and the the detailed table design:



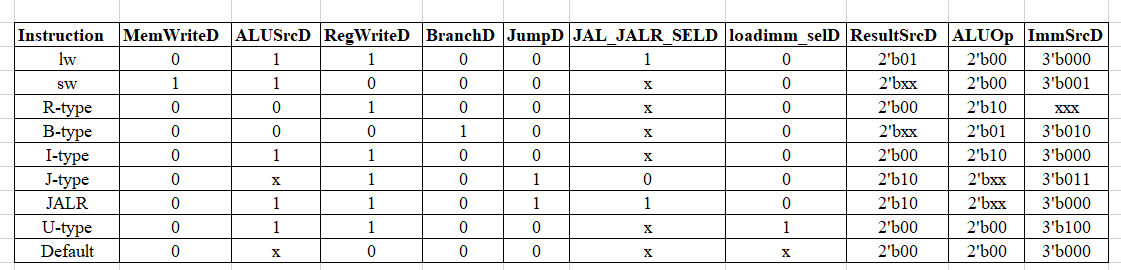
**Table . Immediate Structure**

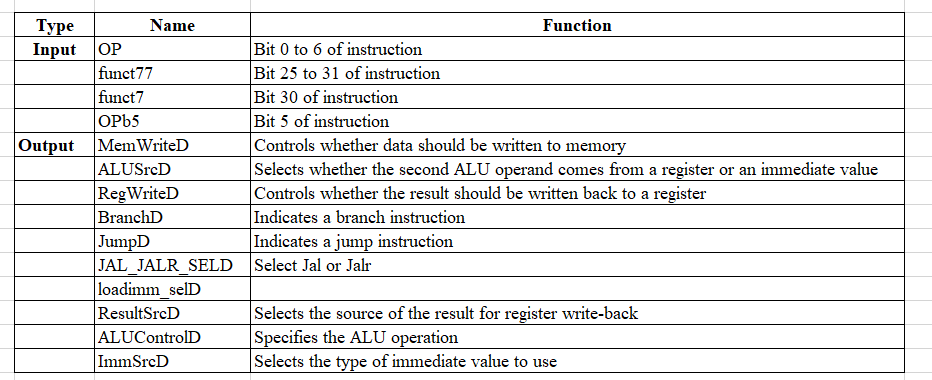
****

**Table . Immediate Types**

**4.2 Sequential units**

4.2.1 Control Unit





4.2.2 Hazard Detect Unit

**Analysis**

Pipeline dangers occur when the subsequent instruction cannot be carried out in the subsequent clock cycle. The three categories of these scenarios are control hazard, data hazard, and structural hazard.

**Structural hazard:** happens when a planned instruction cannot execute in the proper clock cycle because the hardware does not support the combination of instructions that are set to execute. In the present case, our design uses two memories, in the same clock cycle fetching data from multiple adjacent instructions will not be a problem. However, without two memories, our pipeline could have a structural hazard.

**Data hazard:** the condition in which an instruction that is ready but not yet available prevents it from executing within the correct clock cycle. When a command depends on a previous one that is still being processed, data hazards occur.

**Control hazard:** also known as branch hazard, occurs when an instruction does not execute in the correct pipeline clock cycle because the one that was fetched does not need to be fetched; in other words, the instruction addresses do not flow as predicted by the pipeline. This occurs when the pipeline processor carries out a branch instruction; if the incorrect next instruction is carried out, there may be a penalty branch.

**Solutions:**

**Structural hazard:** this type of danger is resolved by applying the Harvard structure and two distinct memories.

**Data hazard:**

- **Forwarding:** the idea of forwarding is to provide data to the ALU's input for instructions that come after, even if the producing instruction hasn't reached WB yet to write the memory or registers.

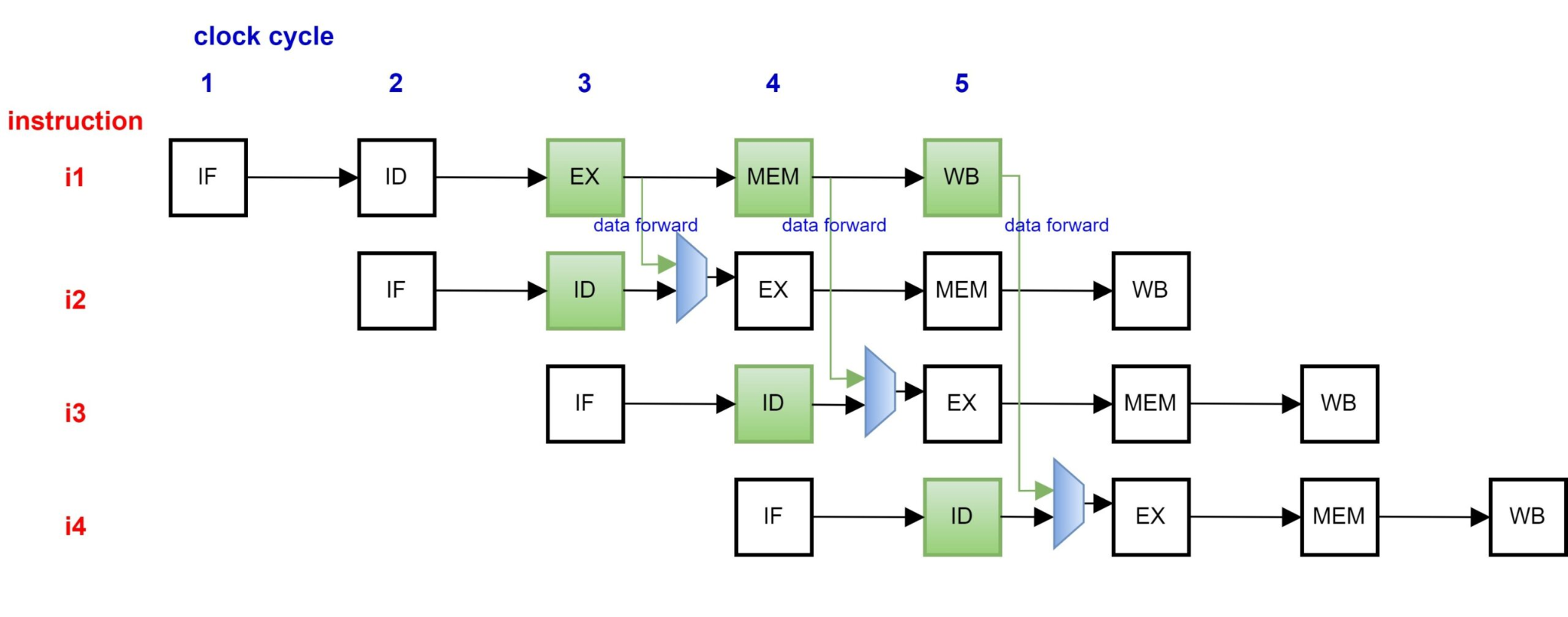
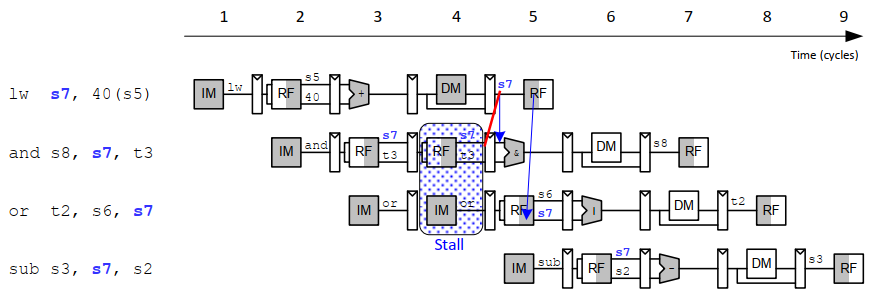


Figure . Data Forwarding to mitigate RAW Hazards in 5-stage Pipeline

(<https://chipmunklogic.com/digital-logic-design/designing-pequeno-risc-v-cpu-from-scratch-part-3-dealing-with-pipeline-hazards/>)

- **Stalling:** the temporary halting of instruction flow across the pipeline. If certain dependencies prevent forwarding, the pipeline can be stopped or halted until the required data is available. This is delaying further instructions until the data is ready by inserting bubbles or "nop" (no operation) instructions.



**Figure** . **Example of stalling to solve lw Data Dependency**

**4.3 Memory**

4.3.1 IMEM

Accepts instruction address from CPU and sends instructions value to the CPU

In this project, instructions will be sent to the IMEM through instruction.mem by using the command:

| readmemh | read the data of the file |
| --- | --- |

Here, when the readmemh reads the regfile txt file, the address set default the number from 0 with step unit being 1. However, the address of RISC - V is an arithmetic progression of 4, when we want to access the address of RISC - V, we just need to have address from bit 31 to bit 2 instead of from bit 31 to bit 0.

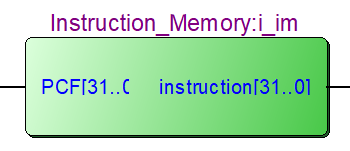


Figure . RTL view of IMEM

4.3.2 DMEM (Load Store Unit)

