I have designed and simulated the 5-stage Pipeline RISC processor, which can execute a subset of MIPS32. The five stages are Instruction Fetch, Instruction Decode, Execute, Memory Access, and Write Back, and all run parallel to make an efficient system. My focus was to implement an efficient pipeline architecture and address all types of hazards, such as data, control, and structural hazards.

**Let me break down the pipeline stages and explain to you**

**The first stage is Instruction Fetch:**

**The purpose** of this stage is to fetch the instructions and increment the Program Counter.

* PC holds the address of current instruction.
* PC increments by 4, at each clock cycle as its 32-bit processor.
* If the branch has been taken, then the PC is incremented by the targeted address.

Challenge:

My challenge here is to handle the branch, so for that, I have taken a 1-bit FF., and first, at every stage, I am checking that FF. If its value is 1 then my PC is going to point on the targeted address.

**The second stage is Decode:**

**The purpose** of this stage is to decode the fetched instruction which includes the operand and operations.

In this stage, I have extracted the ALU Function, source, destination register, and immediate fields from instructions.

So for this stage, I have implemented the register bank, ALU, and Memory.

**Challenge:** If any of the sources is the result of previous instruction which is still in I pipeline. So must implement logic to avoid the instruction dependency.

**The third stage is Execute:**

**The purpose** of this stage is to perform arithmetic, logical, and relational operations on the decoded registers and calculate the memory load/store address. Compute the branch target

So in this, ALU is going to perform the operations based on the decoded signal. So for all the ALU operations I have already mentioned the parameter in the fetch stage and written the logic in the form of the case in this stage.

For branch instruction, branch has evaluated and the target address is computed for PC.

**Challenge:** Handle the branch prediction properly. SO here forwarding unit ensure that operands are forwarded for a later stage.

**The 4th stage is MEMORY ACCESS:**

**The purpose** of this stage is to implement a data memory array and add logic for load/store instructions. I ensured proper alignment of memory accesses by shifting addresses accordingly.

**Challenge**: The major challenge here is when two or more tasks are trying to access the same resource. Structural hazard

**The 5th stage is Write back:**

I have implemented the Multiplexer to differentiate between ALU results need to write back on the destination register in register bank.

Let me discuss hazard detection and overcome the issues of hazards

Data hazards: I have implemented the forwarding logic to avoid data instruction dependency. This logic aims to ensure that if RAW hazards occur, then bypass the results of the previous one. Here I am going to compare the source register value of the current instruction with the destination value of the previous instruction. If it matches then do not require WB or MEM. Directly send the result from EX\_MEM Latch to Input to ALU in ID\_EX latch.

**Control Hazards:** "For control hazards caused by branches, I calculated the branch target address during the ID stage and implemented a branch prediction mechanism to ensure that if branch has taken than PC should be computed to different address. Otherwise pC works sequential manner

**Structural Hazards :** The design avoided structural hazards by ensuring separate instruction and data memory access in the pipeline stages."

**Basic Concepts**

1. **What is pipelining in the context of MIPS32?**
2. **How does pipelining improve performance in a processor?**
3. **What are the main stages of a MIPS32 pipeline?**
4. **Explain the role of the instruction register (IR) and program counter (PC) in a MIPS32 pipeline.**
5. **What are hazards in pipelining, and how do they impact the processor design?**

**Design and Implementation**

1. **Describe the structure of the MIPS32 pipeline you designed.**
2. **What strategies did you use to handle data hazards in your design?**
3. **How did you implement control hazard detection and resolution?**
4. **Did you implement forwarding (bypassing)? If so, how did it work in your design?**
5. **Explain how branch prediction was handled in your MIPS32 pipeline.**

**Debugging and Testing**

1. **What test cases did you develop to verify the functionality of your pipeline processor?**
2. **How did you verify the correct implementation of forwarding and stalling mechanisms?**
3. **What tools did you use to simulate your design, and why did you choose them?**
4. **Describe an instance where you encountered a bug in the design. How did you identify and fix it?**
5. **How did you ensure that your design adheres to the MIPS32 instruction set?**

**Performance and Optimization**

1. **How did you measure the performance of your pipeline?**
2. **What metrics did you use to evaluate CPI (Cycles Per Instruction)?**
3. **Did you encounter a scenario where pipelining reduced performance instead of improving it? Why?**
4. **What optimizations did you implement to improve pipeline efficiency?**
5. **How does pipeline depth impact performance and clock cycle in your design?**

**Advanced Topics**

1. **How did you handle exceptions and interrupts in the MIPS32 pipeline?**
2. **Did you implement multi-cycle or pipelined execution for memory operations?**
3. **What are structural hazards, and how were they avoided in your design?**
4. **How would you extend your design to support additional features, such as floating-point instructions or SIMD operations?**
5. **How would you modify the pipeline to improve power efficiency?**

**General Understanding**

1. **How does the MIPS32 pipeline differ from other processor architectures?**
2. **Explain the difference between a single-cycle processor and a pipelined processor.**
3. **What challenges did you face during the project, and how did you overcome them?**
4. **How would you compare the performance of a 5-stage MIPS32 pipeline with a superscalar processor?**
5. **What lessons did you learn from this project that you can apply to other design or verification tasks?**

**Project-Specific**

1. **What motivated the design decisions you made in your MIPS32 pipeline?**
2. **What was the most challenging part of implementing this project, and why?**
3. **What hardware description language (HDL) did you use, and how did it aid the design process?**
4. **How did you document and present your results?**
5. **If given more time, what improvements would you make to your design?**