What are Pipeline Hazards?

As we all know, the CPU's speed is limited by memory. There's one more case to consider, i.e. a few instructions are at some stage of execution in a pipelined design. There is a chance that these sets of instructions will become dependent on one another, reducing the pipeline's pace. Dependencies arise for a variety of reasons, which we will examine shortly. The dependencies in the pipeline are referred to as hazards since they put the execution at risk.

We can swap the terms, dependencies and hazards since they are used interchangeably in computer architecture. A hazard, in essence, prevents an instruction present in the pipe from being performed during the specified clock cycle. Since each of the instructions may be in a separate machine cycle, we use the term clock cycle.

Types of Pipeline Hazards in Computer Architecture

The three different types of hazards in computer architecture are:

1. Structural

2. Data

3. Control

Dependencies can be addressed in a variety of ways. The easiest is to introduce a bubble into the pipeline, which stalls it and limits throughput. The bubble forces the next instruction to wait until the previous one is completed.

Structural Hazard

Hardware resource conflicts among the instructions in the pipeline cause structural hazards. Memory, a GPR Register, or an ALU might all be used as resources here. When more than one instruction in the pipe requires access to the very same resource in the same clock cycle, a resource conflict is said to arise. In an overlapping pipelined execution, this is a circumstance where the hardware cannot handle all potential combinations. Know more about structural hazards here.

Data Hazards

Data hazards in pipelining emerge when the execution of one instruction is dependent on the results of another instruction that is still being processed in the pipeline. The order of the READ

or WRITE operations on the register is used to classify data threats into three groups. Know more about data hazards here.

Control Hazards

Branch hazards are caused by branch instructions and are known as control hazards in computer architecture. The flow of program/instruction execution is controlled by branch instructions. Remember that conditional statements are used in higher-level languages for iterative loops and condition testing (***correlate with while, for, and if case statements***). These are converted into one of the BRANCH instruction variations. As a result, when the decision to execute one instruction is reliant on the result of another instruction, such as a conditional branch, which examines the condition's consequent value, a conditional hazard develops. Know more about control hazards in pipelining here.