Data Hazards

- Data hazards occur in executing arithmetic or data transfer operations, when the next instruction needs to wait for the previous instruction to complete its data read/write.

- For example, if we have the following two instructions:
  
  LOAD R1 A # A => R1
  SUB R0 R1 R2 # R0 - R1 => R2
  
  • R1 used by SUB won’t be ready till after executing LOAD R1 A.
  • So, we have to stall one stage for the SUB instruction. The pipelining for these two instructions look like
    
    load F D E W
    \  
    sub F D E W
  
  • We also call this type of data hazard “load-use data hazard,” which is a specific form of data hazard, in which the date being loaded by a load instruction has not yet become available when it is needed by another instruction.

- Another way to address the data hazards is to retrieve the missing data element from internal buffers (data path) rather than waiting for it to arrive from programmer-visible register or memory. We name this solution “forwarding.”

- For example, if we have the following two instructions:
  
  ADD R1 R2 R0 # R1 + R2 => R0
  SUB R0 R3 R4 # R0 - R3 => R4
  
  - The value of R0, which is R1 + R2, used by SUB won’t be ready till ALU finish the calculation.
  - If using forwarding, the pipelining for these two instructions look like
    
    add F D E W
    \  
    sub F D E W
  
  - In this way, there is no need to stall one stage for the SUB instruction.