HW3 Yukai Jin yj25b

Test file	Forwarding disabled	EX/MEM -> EX Enabled (f 1)	EX/MEM & MEM/WB -> EX enabled (f 2)
Instruction.txt	18 cycles	14	12
Instruction2.txt	23 cycles	19	16
Instruction3.txt	16 cycles	13	11

Explanation:

We only need to take care of ALU (ADD, SUB, DEV, MUL) and LW, only theses instructions have RAW hazard. When implementing, as we have return False (no **Hazard**) to be the final situation (at the end of *hasDependency()* function), we need to consider what situations trigger the **Hazard** fault.

i. EX/MEM-> EX Enabled (f 1)

In the **f = 1** case, a forwarding path exists between the EX and MEM stages.
This allows results from ALU operations in the MEM stage to be **bypassed directly** to the next instruction in EX, avoiding unnecessary stalls.

However, there is an exception when the instruction currently in the **Decode (ID)** stage depends on **two different previous ALU results**.
In this case, one of the needed operands might only be available at the **WB** stage, even though the other operand can already be forwarded from the MEM stage.
Because the EX/MEM forwarding path cannot supply the WB-stage value yet, the dependent instruction must **wait one bubble cycle** until that WB result becomes available.

Additionally, when the decoding instruction depends on a **load (LW)** instruction, the load's data will only be ready **after the MEM stage**, meaning it must wait **two cycles** before execution can proceed.

Therefore, in the implementation, two separate if conditions are required:

- one to handle the **ALU result still in the WB stage**(no matter what in MEM), and
- one to handle the **LW result in either the MEM or WB stage**.

ii. EX/MEM & MEM/WB -> EX enabled (f 2)

In the **f = 2** case, forwarding paths exist from both **MEM** and **WB** to the **EX** stage.
This provides full bypass coverage for **all ALU operations**, since results from either stage can be forwarded to the dependent instruction in EX without delay.

As a result, the **only remaining stall condition** occurs when the dependent instruction needs data from a **load (LW)** instruction that is **currently in the MEM stage**, because the data from memory has not yet been fetched and cannot be forwarded.

Consequently, only **one if condition** is necessary in this configuration:

- to handle the single LW-in-MEM hazard.