**Hazard Detection**

**Data Hazard**

**Data hazard occurs in the following situation:-**

* Load-use and the next instruction is using one of the operand of the current instruction
* LDM Rdst,Imm : the Rdst is in the position of the Rsrc ( Special Case ).
* LDD Rsrc,Rdst

**Solution** is to stall the pipeline one cycle (stall bit is added in the control word).

**How to stall the pipeline?!**

* Disabling The **IR** & **PC** (Don’t take the new instruction that the **PC** stored its address and don’t increment **PC**).
* Reset The Buffer between the decoder stage and execution stage (act as **NOP**).

**Stall Logic**

Dependency = (EXE\_Rsrc=DEC\_Rsrc AND LDM) OR

(EXE\_Rsrc=DEC\_Rdst AND LDD)

Stall = Dependency AND EXE\_Memory\_Read

**Structural Hazard**

**Structural hazard occurs in the following situation:-**

* Instruction Fetch is conflicting with data memory access.
* **Solution** is to use two separated memory (Data and Instruction).
* Register File is accessed by the instruction in the **decode** stage (reading) and at the same time other instruction in the **write-back** stage.
* **Solution** is to ensure that writing is done in the falling edge and reading is done in the rising edge.

**Control Hazard**

**Control hazard occurs in the following situation:-**

* **Branch Taken (Jump)**
* Static prediction is to not take the branch and continue the program **(NOT-TAKEN).**
* We know that the instruction is branch in the decode stage.
* In the execution we raise one bit called **branch\_taken**.
* **Load Immediate**
* Some instructions take a load value from the next instruction so I’ve to take this value and put it in the control signal then flushing the instruction (actually the instruction is a value not an instruction).

**Solution** is to flush the new instruction or (value).

**How to flush the instruction or value?!**

* Reset the IR (acts as **NOP**).

**Flush Logic**

Flush = immediate\_load OR branch\_taken