**Pipeline in a Processor:**

**Pipeline:-**

1)Instruction latency for a particular instruction in the pipelined microarch remains the same.

2)The throughput (number of instructions out of the pipeline per cycle after the first instruction) increases in a pipelined microarchitecture.

Pipeline stalls in a processor can be due one or more of the following:

🡪Not every dependency is bad for the pipeline & but more importantly dependencies which causes pipeline to be stalled/freeze/flush could in more than one-way result in a pipeline hazard.

🡪Pipeline interlock hardware inserts bubbles/stalls into the pipeline.

🡪Dependence is the property of program alone.

🡪Hazard: - some dependencies results in incorrect program execution.

1)Control dependency: - Caused due to dependency on unconditional/conditional branch outcome.

2)Data Dependencies: -

a) RAW (True/flow dependency)

b) WAW (Output dependency) [This could happen during out-of-order execution]

c)WAR (Anti-dependency) [This could happen during out-of-order execution]

d)RAR (This will not cause any dependency/lock the pipeline. Hence this type is irrelevant).

Ex.: Consider the following instruction sequence executing on the 5-stage MIPS

pipeline *with no forwarding*.

**LOAD R2,0(R1)**

**ADD R1,R2,R3**

**STORE R2,0(R1)**

1. [25] Identify all *dependences* (by instruction pair and register or memory location involved) and their type (e.g. true dependence, anti-dependence, output dependence). For each dependence, indicate whether there is a possible hazard in this pipeline and identify, if there is a potential hazard, the type (RAW, WAW, WAR).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1st instruction** | **2nd instruction** | **Register/**  **Memory** | **Type of Dependence** | **Possible**  **Hazard** | **Type of**  **Hazard** |
| **LOAD** | **ADD** | **Reg R2** | **True** | **Yes** | **RAW** |
| **ADD** | **STORE** | **Reg R1** | **True** | **Yes** | **RAW** |
| **LOAD** | **ADD** | **Reg R1** | **Anti-** | **No** | **-** |
| **LOAD** | **STORE** | **Reg R2** | **True** | **Yes** | **RAW** |
| **LOAD** | **STORE** | **0(R1)** | **Anti-** | **No** | **-** |

**b)** [20] Complete the pipeline execution table below to show the execution of this

code fragment on the 5-state MIPS pipeline ***without forwarding***. Indicate any stalls.

**Ans.:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction/**  **Cycle** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **LOAD R2, 0 (R1)** | **IF** | **ID** | **EX** | **MEM** | **WB** |  |  |  |  |  |  |
| **ADD R1, R2, R3** |  | **IF** | **ID** | **Stall** | **Stall** | **EX** | **MEM** | **WB** |  |  |  |
| **STORE R1, 0 (R2)** |  |  | **IF** | **Stall** | **Stall** | **ID** | **Stall** | **Stall** | **EX** | **MEM** | **WB** |

**c)** [20] Redo the pipeline execution diagram above assuming all possible forwarding paths have been added to the pipeline. Identify the cycle and stages involved in all forwarding (e.g. “Value destined for Rn forwarded from EX to EX state in cycle 10”) or indicate it with an arrow in the diagram.

**Ans.:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction/**  **Cycle** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **LOAD R2, 0 (R1)** | **IF** | **ID** | **EX** | **MEM** | **WB** |  |  |  |  |  |  |
| **ADD R1, R2, R3** |  | **IF** | **ID** | **Stall** | **EX** | **MEM** | **WB** |  |  |  |  |
| **STORE R1, 0 (R2)** |  |  | **IF** | **Stall** | **ID** | **EX** | **MEM** | **WB** |  |  |  |

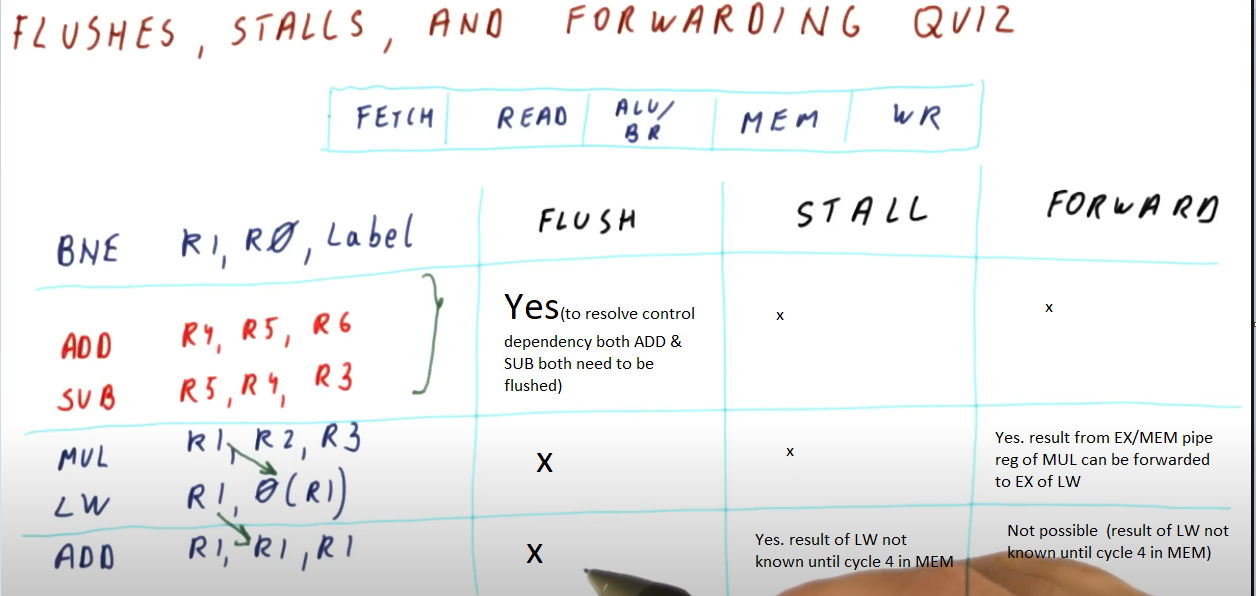
Handling Hazard Situations:

1) Flush dependent instruction (mostly needed when a control dependency becomes a hazard)

2) Stall dependent instruction (Could be useful when RAW dependency turns into a hazard)

3)Fix values read by dependent instructions (Forwarding – could be useful for some data hazards)

Ex:



**How deep the pipeline can be ideally?**

No. of stages => balance of CPI & cycle time

🡪Too few stages increase cycle time while too more stages increase CPI (due to potential hazards).

🡪There has to be a balance between CPI & cycle time.

🡪Ideally 10-15 stages balances both performance & power in modern processors.