DAY 14-111 DAYS VERIFICATION CHALLENGE

Topic : Pipelining hazards

Skill : Computer Architecture

DAY 14 CHALLENGE:

1. Explain the different types pipelining hazards with examples (Structural, Control, Data hazards)

1. Structural Hazards

Structural hazards occur when hardware resources are insufficient to support all the simultaneous operations in a pipeline. These hazards happen when two or more instructions need the same resource at the same time.

Example: Consider a simple pipeline where both instruction fetch (IF) and memory access (MEM) stages need to access the memory. If there is only one memory unit, a structural hazard will occur if an instruction needs to be fetched at the same time another instruction needs to access memory.

- Instruction 1: Load from memory (needs MEM stage)
- Instruction 2: Fetch next instruction (needs IF stage)

If both instructions need to access memory simultaneously, the pipeline must stall one of the instructions until the memory resource is free.

2. Control Hazards

Control hazards, also known as branch hazards, occur due to the pipelining of branch instructions. When the pipeline makes the wrong decision on branch prediction, it needs to correct its actions, leading to stalls.

Example: Consider a pipeline where branch prediction is used to guess the outcome of branch instructions. If the pipeline predicts that the branch will not be taken, but it turns out the branch should be taken, the instructions that were fetched and partially executed must be discarded, and the correct instructions must be fetched.

- Instruction 1: Branch if zero (BEQ)

- Instruction 2: Add (predicted as the next instruction if the branch is not taken)
- Instruction 3: Subtract (actual instruction if the branch is taken)

If the branch is taken, the pipeline must flush the Add instruction and fetch the Subtract instruction instead, causing a delay.

3. Data Hazards

Data hazards occur when instructions that exhibit data dependence modify data in different stages of a pipeline. There are three types of data hazards:

- **RAW (Read After Write):** An instruction tries to read a source before a previous instruction writes to it.
- **WAR (Write After Read):** An instruction tries to write a destination before it is read by a previous instruction.
- **WAW (Write After Write):** An instruction tries to write an operand before it is written by a previous instruction.

Example (RAW Hazard): Consider a pipeline where one instruction depends on the result of a previous instruction.

- Instruction 1: ADD R1, R2, R3 (R1 = R2 + R3)
- Instruction 2: SUB R4, R1, R5 (R4 = R1 - R5)

Instruction 2 needs the result of Instruction 1. If Instruction 2 reaches the EX (Execute) stage before Instruction 1 completes the WB (Write Back) stage, a RAW hazard occurs. The pipeline must stall Instruction 2 until Instruction 1 completes.

2. What are ways to resolve different pipeline hazards? What are pros & cons of these resolution techniques?

1. Structural Hazards

1. **Resource Duplication:**
    - **Description:** Add more instances of the resource (e.g., separate instruction and data caches).
    - **Example:** Using separate caches for instructions and data.
    - **Pros:** Eliminates resource contention, improves throughput.
    - **Cons:** Increases hardware cost and complexity.
2. **Resource Scheduling:**
    - **Description:** Schedule operations to avoid conflicts.
    - **Example:** Staggering the access times of conflicting resources.

- **Pros:** Efficient resource utilization.
- **Cons:** Adds control complexity.

2. Control Hazards

Techniques:

1. **Branch Prediction:**
   - **Description:** Predict the outcome of branch instructions.
   - **Example:** Using a branch prediction algorithm to guess if a branch will be taken.
   - **Pros:** Reduces stalls if predictions are accurate.
   - **Cons:** Incorrect predictions cause pipeline flushes and delays.
2. **Branch Target Buffer (BTB):**
   - **Description:** Cache branch target addresses.
   - **Example:** Storing target addresses of recently taken branches for quick access.
   - **Pros:** Quickly resolves branch targets, reduces delays.
   - **Cons:** Requires additional hardware.
3. **Delayed Branching:**
   - **Description:** Rearrange instructions to fill branch delay slots.
   - **Example:** Placing useful instructions after a branch instruction.
   - **Pros:** Utilizes otherwise wasted cycles.
   - **Cons:** Complicates instruction scheduling, requires compiler support.

3. Data Hazards

Techniques:

1. **Forwarding (Bypassing):**
   - **Description:** Pass results directly to dependent instructions.
   - **Example:** Forwarding the result of an arithmetic operation to the next instruction needing it.
   - **Pros:** Minimizes pipeline stalls due to data dependencies.
   - **Cons:** Adds complexity to pipeline control logic.
2. **Pipeline Stalling (NOPs):**
   - **Description:** Insert no-operation instructions to delay dependent instructions.
   - **Example:** Adding NOPs between dependent instructions.
   - **Pros:** Simple implementation, ensures correct execution.
   - **Cons:** Reduces pipeline efficiency and overall throughput.
3. **Instruction Reordering (Out-of-Order Execution):**

- **Description:** Execute independent instructions while waiting for data.
- **Example:** Executing other instructions while waiting for a memory load to complete.
- **Pros:** Maximizes pipeline utilization.
- **Cons:** Significantly increases hardware complexity and control logic.

4. **Register Renaming:**
   - **Description:** Use additional registers to avoid false dependencies.
   - **Example:** Renaming registers to prevent conflicts in instruction execution.
   - **Pros:** Resolves WAR and WAW hazards, increases parallelism.
   - **Cons:** Requires additional hardware for renaming logic, adds complexity.

3. Identify the pipelining hazards (also tell the sub-category e.g., Data hazard - WAW) in below sequence of instructions. Explain why these instructions are resulting in pipelining hazard.

a. SUB R1, R4, R3

ADD R1, R2, R3

Data hazards and sub category is WAW- write after write
because first SUB operation write in R1 and after this ADD again write in R1

b. ADD R1, R2, R3

SUB R4, R1, R3

Data hazards and sub category is WAR- write after read
because first ADD operation write in R1 and after this SUB again read in R1

c. SUB R4, R1, R3

ADD R1, R2, R3

Data hazards and sub category is RAR- read after read
because first SUB operation read R3 and after this ADD again read in R3