Topic: Out of Order Processor.

Aimed for: Performance *Author: Abdalrahman Ebdah*

Objective

Design a multi-issue processor with out of order execution using dynamic scheduling scheme to get higher execution performance.

Performance Enhancement

In processor design, performance is often evaluated by metrics such as Cycles Per Instruction (CPI). To improve CPI and overall execution efficiency, one effective technique is to increase the number of instructions issued per cycle. This approach allows the processor to perform more work in each clock cycle, ultimately reducing the time needed for instruction completion and improving performance.

Dependencies

In multi-issue processors with out-of-order execution, there are several types of dependencies that must be carefully managed to avoid execution hazards. These dependencies include:

- **Data Dependency**: This occurs when instructions depend on the results of previous instructions. There are three types of data dependencies:
  - *Read-after-write (RAW)*: When an instruction depends on the result of a previous instruction.
  - *Write-after-read (WAR)*: When an instruction writes to a register that a previous instruction is reading.
  - *Write-after-write (WAW)*: When two instructions write to the same register.
- **Control Dependency**: This is related to branch instructions where the flow of execution is determined by a condition, which can cause delays while waiting for the branch to be resolved.
- **Structural Dependency**: This happens when multiple instructions require the same functional unit simultaneously, such as two arithmetic instructions needing access to the same Arithmetic

Issuing multiple instructions per cycle can introduce new dependencies, including **false dependencies** (where data dependencies arise due to the out-of-order execution of instructions). These false dependencies must be managed to prevent incorrect execution. Techniques such as **register renaming** and **scheduling** are applied to handle these dependencies.

Renaming

Renaming is a technique used to eliminate false data dependencies, such as Write-After-Read (WAR) and Write-After-Write (WAW) hazards, which can occur in out-of-order processors. These false dependencies can arise when the same register is used by multiple instructions in ways that do not truly depend on each other. Renaming helps by dynamically assigning new registers to instructions, thus avoiding unnecessary serialization. By renaming registers, the processor can ensure that each instruction operates on a unique data set, eliminating the need to wait for previous instructions to complete. This increases parallelism and reduces the number of cycles wasted on false dependencies, ultimately improving processor performance.

Scheduling

Scheduling is another crucial technique used to manage structural hazards that arise when multiple instructions require the same functional unit. In out-of-order execution, instructions may not always arrive in the optimal order for execution, and as a result, there may be conflicts where multiple instructions attempt to use the same resource (such as the ALU or memory). To resolve this, dynamic scheduling is employed to rearrange the order in which instructions are issued. This ensures that no two instructions that depend on the same resource are issued simultaneously, thereby reducing the occurrence of structural hazards. By carefully controlling the issue of instructions, the processor can maximize resource utilization and minimize execution stalls, further enhancing performance.

Units in Out-of-Order Processors

In out-of-order processors, several key hardware units are employed to support dynamic instruction scheduling and manage dependencies efficiently. These units include the **Register Alias Table (RAT)**, **Reservation Stations**, and **Reorder Buffer (ROB)**.

- **Register Alias Table (RAT)**: The RAT is used for register renaming to eliminate false dependencies such as Write-After-Write (WAW) and Write-After-Read (WAR). It maps architectural registers to physical registers, allowing instructions to use physical registers without conflicting with previous instructions using the same architectural registers. This enables out-of-order execution by ensuring that each instruction works on unique data locations.
- **Reservation Stations**: These are temporary storage buffers that hold instructions waiting for operands before they can be executed. When an instruction is issued, it is placed in a reservation station where it waits for the necessary data to become available. Once all operands are ready, the instruction is sent to the appropriate functional unit for execution. This allows instructions to continue executing as soon as their operands are available, even if previous instructions have not yet completed.
- **Reorder Buffer (ROB)**: The ROB is used to maintain the original program order of instructions, ensuring that results are written back to registers in the correct sequence. While instructions may execute out of order, the ROB tracks their completion status and ensures that the final write-back happens in the correct order to maintain consistency in the program's state. It also helps in recovering from branch mispredictions by discarding incorrect speculative results.

These units, working together, allow an out-of-order processor to execute instructions more efficiently by optimizing resource usage and managing dependencies while maintaining the program's correct logical flow.