Computer Architecture week 5

Khaula Molapo

1.

A superscalar processor in a web server scenario can handle multiple instructions at the same time, significantly improving performance. Imagine a server processing requests from hundreds of users simultaneously. A traditional scalar processor would execute one instruction per cycle, creating a bottleneck. A superscalar processor, however, can fetch, decode, and dispatch several instructions in parallel if resources are available. For example, one execution unit can process an arithmetic calculation while another processes a memory access. This allows the server to deliver web pages faster, reducing response times and enhancing user experience. When multiple users request images, scripts, or HTML files, the processor's parallelism ensures that tasks like encryption, data retrieval, and rendering are handled concurrently. The operating system schedules tasks, while the superscalar architecture ensures that multiple instructions from different threads are executed efficiently. This scenario highlights why superscalar design is ideal for heavy workloads like web servers.

2.

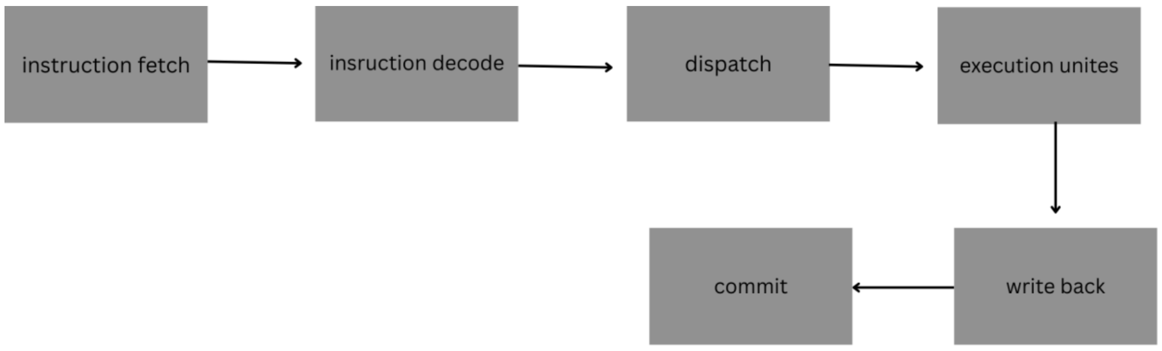
Dynamic scheduling is a technique in computer architecture that allows instructions to be executed out of order while maintaining correct results. Instead of stalling when a data dependency occurs, the processor uses hardware mechanisms like reservation stations and reorder buffers to rearrange instructions. This ensures that independent instructions can proceed while dependent ones wait for data. Dynamic scheduling improves CPU utilization and reduces idle cycles, which is especially important in superscalar processors where multiple instructions are issued per cycle. By allowing parallelism beyond compiler limitations, it adapts execution to runtime conditions, making programs run faster and more efficiently.

3.

For practice, I designed a simple superscalar pipeline with four main stages: fetch, decode, execute, and write-back. Unlike a scalar pipeline, this design allows two instructions to be fetched and decoded per cycle. In the execute stage, separate functional units handle arithmetic, logic, and memory operations simultaneously. Reflection: This exercise made me realize how important parallelism is to modern CPUs. A superscalar pipeline reduces latency and maximizes throughput by overlapping tasks. However, hazards like data dependencies and branch mispredictions remain challenges. Designing this simple model showed me why hardware techniques like dynamic scheduling are essential for maintaining efficiency.

4.

In Canva, I created a superscalar pipeline diagram. The diagram begins with an Instruction Fetch Unit, which pulls multiple instructions per cycle. These flow into the Instruction Decode Unit, where they are prepared for execution. Next, arrows branch into multiple Execution Units—an ALU, a Load/Store Unit, and a Floating-Point Unit—showing parallel execution paths. The outputs then converge at the Write-Back Stage, updating registers. The diagram visually demonstrates how superscalar processors exploit instruction-level parallelism. By showing multiple paths instead of one, the diagram highlights efficiency and throughput. It makes it clear that superscalar CPUs focus on parallel execution rather than sequential processing.


instruction fetch
insruction decode
dispatch
execution unites
write back
commit