Khaula Molapo

Computer Architecture week 6

1.

In an embedded system, such as a smart thermostat, the Instruction Set Architecture (ISA) plays a critical role in defining how the processor executes commands. The ISA acts as a contract between hardware and software, specifying instructions, registers, and data formats the CPU understands. For a thermostat, energy efficiency is crucial, so the processor uses a compact ISA optimized for low power. The software controlling the thermostat uses instructions defined by this ISA to read temperature sensors, process user input, and control heating or cooling systems. Because the ISA determines how efficiently these tasks run, it directly impacts performance and battery life. For example, a reduced instruction set allows faster execution of critical operations. This makes the system more responsive while conserving power. In embedded devices where processing power and energy are limited, selecting the right ISA is essential for balancing speed, efficiency, and complexity.

2.

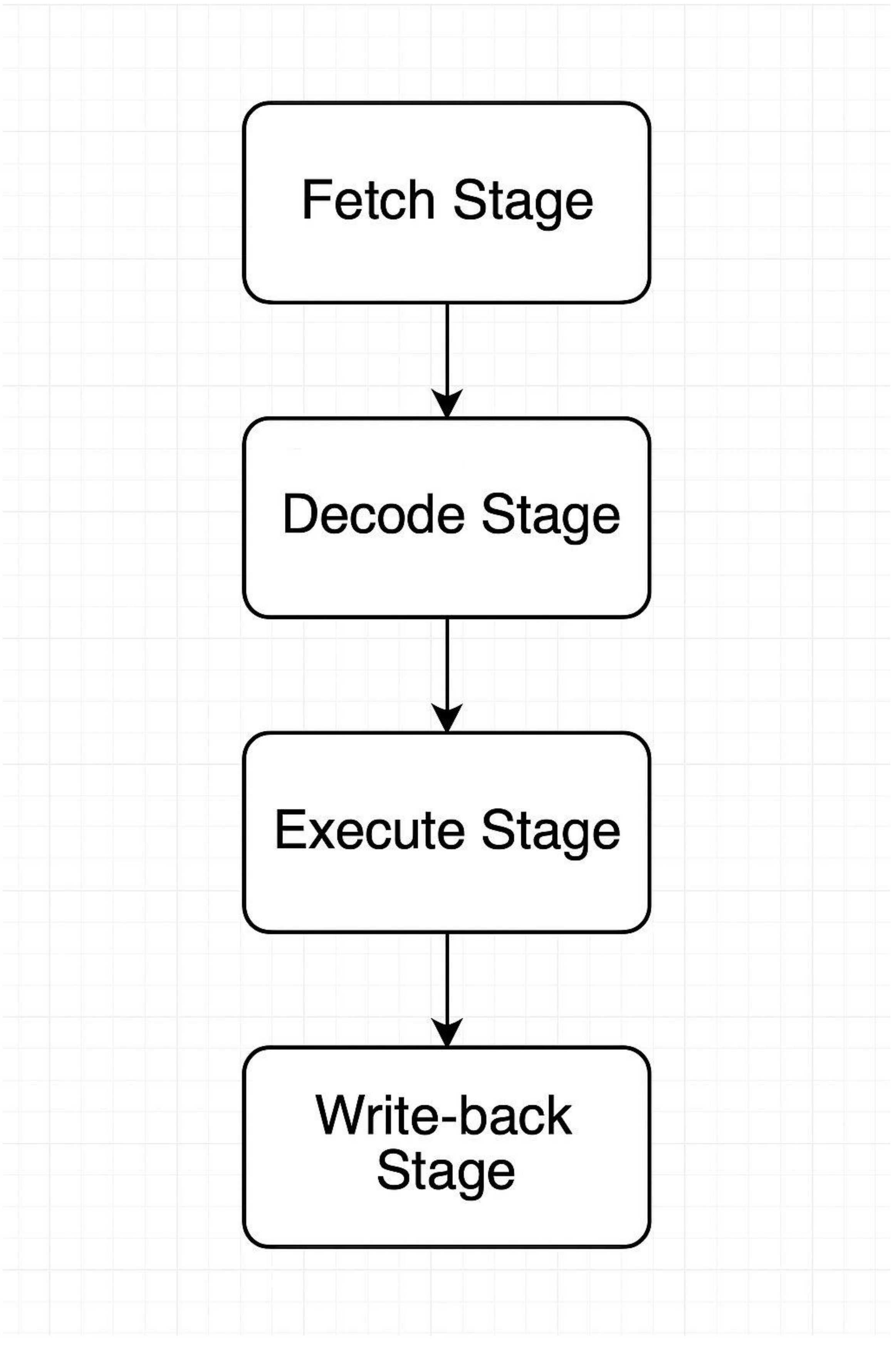
RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) are two CPU design philosophies. RISC uses a small, optimized set of instructions, making execution faster and simpler, often requiring fewer cycles per instruction. It emphasizes efficiency and predictability, beneficial for embedded systems and high-performance applications. CISC uses a larger set of more complex instructions, allowing more work per instruction but often requiring multiple cycles. It reduces program size and simplifies coding. The choice depends on application needs: RISC for speed and simplicity, CISC for flexibility and compact code. Both approaches shape modern processor design and performance.

3.

In Logisim, I simulated a basic instruction for a CPU design. I created a simple ALU setup, connected control signals, and defined the instruction format. I chose an ADD instruction to test register values, sending them to the ALU. After running the simulation, I observed how control lines and data paths worked together to execute the instruction. The result matched expectations, confirming correct simulation. This exercise showed how each instruction follows a defined path through hardware components, highlighting the importance of the ISA. It also reinforced my understanding of CPU internals and how Logisim can model them for design and testing purposes.

4.

In Draw.io, I created a flowchart for a CPU execution cycle. It begins with the Fetch Stage, where the CPU retrieves an instruction from memory. Next is the Decode Stage, where the instruction is interpreted and control signals are generated. This is followed by the Execute Stage, where the CPU performs the operation using the ALU. Finally, the Write-back Stage stores results back into registers or memory. This flowchart shows how each instruction progresses through the CPU, emphasizing the structured process behind execution. It clarifies how the ISA governs every step in this cycle for efficiency and accuracy.


Fetch Stage
Decode Stage
Execute Stage
Write-back Stage