Computer Architecture 2

Khaula Molapo

Year 2 Semester 2

Question 1.

Instruction Level Parallelism (ILP) is about executing multiple instructions at the same time to make processors faster. One example is superscalar execution. In a normal processor, instructions are executed one after another, but in superscalar processors, there are multiple execution units. This means the CPU can fetch, decode, and run several instructions in one cycle, as long as they are independent. For example, one instruction can add two numbers while another loads data from memory. By overlapping tasks, performance improves without increasing clock speed. Superscalar execution also depends on the compiler and processor to identify instructions that can run in parallel. However, data dependencies and branch instructions can limit how many instructions run at once. ILP makes modern processors efficient by using hardware like pipelines, multiple execution units, and advanced scheduling. Overall, ILP through superscalar execution increases throughput and keeps the CPU busy.

Question 2.

Branch prediction is an important part of Instruction Level Parallelism (ILP). When a program has conditional instructions, like an “if” statement, the CPU does not know which path will be taken. Without prediction, the CPU would stop and wait, which wastes cycles. Branch prediction guesses the most likely path and continues executing instructions ahead of time. If the guess is correct, performance increases because no time is lost. If the guess is wrong, the CPU discards the wrong path and reloads the correct one. This technique helps pipelines stay full and makes ILP work more effectively.

Question 3.

Simulating a pipeline in Logisim helped me understand how instructions move through different stages of a processor. I saw how fetch, decode, execute, and write-back happen in order, but also how multiple instructions can overlap in these stages. The simulation showed me how pipelining increases performance by keeping all parts of the CPU busy. At the same time, I noticed problems like hazards when two instructions depend on the same data. This reflection made me realize that pipelining is not just about speed but also about managing dependencies. Overall, the simulation made the concept clearer and practical.

Question 4.

The sketch I made in Canva explains the stages of an ILP pipeline. It starts with instruction fetch, where the CPU grabs the next instruction from memory. Then comes decode, where the instruction is understood. After that, execution happens in the arithmetic or logic unit. The next

stage is memory access if needed, followed by write-back, where the result is stored. By arranging these stages in parallel, multiple instructions can be processed at the same time. My sketch shows how overlapping stages improve efficiency. This makes the processor faster without increasing clock speed, showing how ILP pipelines improve performance.