October 1st Lecture Summary

2023-12675 박지호

The first topic covered in today's lecture was an overview of simplest CPU circuit, which was also the final project in the Logic Design class. The CPU consists of many components, including but not limited to instruction memory, registers, data memory, and the computing unit, and the control unit. We learnt how these components interact with each other to perform various operations that we learnt about, like r-format instructions, load/store instruction, branch instruction. Furthermore, we went through the control signals from the control unit, and what values these control signal should have for specific instructions.

In the process, we were also re-introduced to the basics of circuit design, including concepts like binary encoding of data, difference between combinational and sequential logic, and how rising or falling edge of the clock signal enables sequential logic circuit. Likewise, these topics were mostly covered in the Logic Design class, and hence was kept brief.

However, the CPU circuit may have several performance problem. Since accessing the memory takes significantly longer time than other computations, the load instruction acts as a bottleneck, lengthening the clock period.

Then, we were introduced to pipelining as a solution to this problem. Pipelining refers to a strategy where multiple tasks are ran in parallel, drastically reducing the overall execution time. Such pipeline can be achieved by dividing the operation into steps which do not require overlapping resources. Notably, the time required must be balanced throughout the steps in order for the pipelining to be effective.

In RISC-V, a task is broken down into five stages: instruction fetch (IF), instruction decode and register read (ID), execution operation or calculate address (EX), access memory operand (MEM), and write back (WB). RISC-V is also designed to be pipeline friendly, as set length, regularized instruction format allows for easier instruction fetching and decoding.

Although pipelining can bring about performance improvement, it is not without downsides. These are called hazards, and there are three types of hazards: structure, data, and control hazard. Structure hazard wasn't explained thoroughly, but concisely speaking it refers to a resource conflict.

Data hazard refers to a dependency that a later instruction may have to a data created by previous instruction. This can be problematic as in this scenario the evaluation of later instruction cannot start before the previous instruction is fully complete, making the pipelining pointless. In order to counteract this, the result of previous instruction can be sent directly to a specific step that requires a data. This technique is called forwarding. However, forwarding requires additional datapath, and not every data hazard can be resolved by forwarding. Also, data hazard can also be avoided by clever compilation.

Meanwhile, control hazard refers to an issue caused by branch instructions. When evaluating a branch instruction, the next instruction cannot be fetched until the branch instruction is complete, as it depends on the result of the branch instruction. Longer pipeline causes this to be more significant. Branch prediction, static and dynamic, is used to counteract this. The specifics of these prediction methods were not discussed.