# 4.15 Concluding remarks

> "Nine-tenths of wisdom consists of being wise in time."
> *American proverb.*

As we have seen in this chapter, both the datapath and control for a processor can be designed starting with the instruction set architecture and an understanding of the basic characteristics of the technology. In COD Section 4.3 (Building a datapath), we saw how the datapath for an LEGv8 processor could be constructed based on the architecture and the decision to build a single-cycle implementation. Of course, the underlying technology also affects many design decisions by dictating what components can be used in the datapath, as well as whether a single-cycle implementation even makes sense.


**Pipelining** improves throughput but not the inherent execution time, or *instruction latency*, of instructions; for some instructions, the latency is similar in length to the single-cycle approach. Multiple instruction issue adds additional datapath hardware to allow multiple instructions to begin every clock cycle, but at an increase in effective latency. Pipelining was presented as reducing the clock cycle time of the simple single-cycle datapath. Multiple instruction issue, in comparison, clearly focuses on reducing clock cycles per instruction (CPI).

***Instruction latency***: The inherent execution time for an instruction.


Pipelining and multiple issue both attempt to exploit instruction-level parallelism. The presence of data and control dependences, which can become hazards, are the primary limitations on how much parallelism can be exploited. Scheduling and speculation via **prediction**, both in hardware and in software, are the primary techniques used to reduce the performance impact of dependences.

We showed that unrolling the DGEMM loop four times exposed more instructions that could take advantage of the out-of-order execution engine of the Core i7 to more than double performance.

The switch to longer pipelines, multiple instruction issue, and dynamic scheduling in the mid-1990s helped sustain the 60% per year processor performance increase that started in the early 1980s. As mentioned in COD Chapter 1 (Computer Abstractions and Technology), these microprocessors preserved the sequential programming model, but they eventually ran into the power wall. Thus, the industry was forced to switch to multiprocessors, which exploit parallelism at much coarser levels (the subject of COD Chapter 6 (Parallel Processor from Client to Cloud)). This trend has also caused designers to reassess the energy-performance implications of some of the inventions since the mid-1990s, resulting in a simplification of pipelines in the more recent versions of microarchitectures.


To sustain the advances in processing performance via parallel processors, Amdahl's law suggests that another part of the system will become the bottleneck. That bottleneck is the topic of the next chapter: the **memory hierarchy**.

**PARTICIPATION ACTIVITY** 4.15.1: Processor design and technology.

| Speculation | Multiprocessors | Instruction latency |
|---|---|---|
| | | A form of prediction used to improve the performance of data and control dependencies. |
| | | An instruction's built-in execution time. |
| | | Multiple processors used to exploit instruction-level parallelism. |

Reset

Provide feedback on this section