CISC and RISC Architectures: An Overview

CHAPTER OVERVIEW

This chapter discusses the rise of the CISC architecture and the RISC architecture; it then discusses modern hybrid processor architectures.

This chapter includes:

▶ A discussion of RISC and CISC processors;
▶ A brief comparison of RISC and CISC;
▶ Modern hybrid processors.

19.1 CISC Processors

As I mentioned in chapter 17, our simple processor is an example of a CISC, Complex Instruction Set Computer, processor. CISC processors have been with us since the dawn of the electronic computer, even if the term CISC hasn’t.

Generally, as they were initially built using vacuum tube components that often failed, CISC processors began life with a very limited number of registers, and a simple hardwired control unit. Over time, newer more reliable technologies were used, leading to more complex processor designs. To give these newer complex processors greater flexibility the control unit was switched from
a hardware design to a microcoded control unit similar to what we saw in chapter 17, making it possible to update instruction sets using firmware. Instruction sets grew more complex to better utilize these new processors.

However, as technologies changed, philosophies didn’t. The trend was to build more and more complex processors, with very complicated instruction sets, and so it was time for a paradigm shift.

### 19.2 RISC Processors

A group of individuals around the globe began to see this level of complexity in processors as self-defeating. It was leading to instruction sets that were difficult to understand and use, which in turn lead to difficulties in writing compilers for such an architecture, which lead to less efficient programs.

By the 1980s, a back-to-basics approach was mooted, whereby a simpler processor design could actually lead to less power hungry and higher performance processors.

This new architecture included as its key features:

- More general purpose registers, typically 32;
- A limited, uniform, and faster instruction set;
- Very limited addressing modes;
- Hardware compiler support;
- A hardwired control unit.

This approach was termed the RISC, Reduced Instruction Set Computer, architecture by one of the leading RISC pioneers, David Patterson. He also coined the term CISC, as a thinly veiled jibe against the complex architectures of the time.

The RISC philosophy is best explained by referring to the four design principles described by Hennessy and Patterson in [PH98]:

- *Simplicity favours regularity.* If we were to look at the instruction set of our simple processor, we would find instructions that took zero, one, two or three operands. It is even possible to design instructions that make use of even more operands, and this is relatively easy to do given that our processor has a microcoded control unit. If we were to use the RISC philosophy it makes sense to use the simplest hardwired control unit possible, and this leads us into needing a very regular instruction set. Most RISC instruction sets have this level of regularity. For example, rather than having `ADD` instructions that can have different numbers of operands (depending on the addressing modes), all `ADD` instructions on a RISC processor will always take the same number of operands;