Chapter 1 is a very brief introduction to Boolean algebra and it's relationship with logical circuits. This is done using the example of a simple machine designed to play tic-tac-toe which the author apparently built when they were very young. This is a useful example, I think, as it demonstrates how a seemingly complex task can be built up out of basic logic functions. The rest of the chapter is spent exploring, through a number of examples, the idea that these logic systems—and indeed, therefore, computers—need not be built from electronic circuits embedded in silicon. Rather, one could build a functioning computer out of any number of things, namely sticks and ropes, water in pipes, or even Tinker Toys. We get some ideas about hierarchies of abstraction toward the end of the chapter.

Chapter 2 mostly extends the ideas in the previous chapter, giving a more formal overview of logic as it might actually be conceptualized and implemented in a real processor. This includes logic gates and truth tables. We then get an informal introduction to finite state machines, using analogies to real-world systems.

The example of the tic-tac-toe machine in chapter 1 was interesting to me, as it's a problem I've come across before, albeit at a much higher level of abstraction in, software. I've been a TA for CSCI 200 since it's inception (and the consequent death of CSCI 261) here at Mines, and implementing a game of tic-tac-toe against the computer has always been a popular final project. While students don't always design their program to employ any real strategy, I'm always curious when they do, as the myriad ways to over-engineer the solution to such a simple problem are fascinating. However, I don't think I've ever realized just how simple a solution one could really come up with until I read Hillis' description of doing it with rudimentary hardware. Perhaps some ideas for future office hours, then.

Something else I haven't given much thought before is the low-level hardware implementation of a cache. As computer scientists, we rather like to leave such details to the electrical engineers and physicists, but it's always interesting to get a peek behind the curtain. I like that the book gives a reason, however frustratingly lacking in detail, that a cache loses its memory when power is interrupted. My interest is piqued, and you're likely to find me knee-deep in Wikipedia articles about memory hierarchy and cache design at some point soon.