Structure And Interpretation Of Computer Programs, Second Edition
**Synopsis**

With an analytical and rigorous approach to problem solving and programming techniques, this book is oriented toward engineering. Structure and Interpretation of Computer Programs emphasizes the central role played by different approaches to dealing with time in computational models. Its unique approach makes it appropriate for an introduction to computer science courses, as well as programming languages and program design.

**Book Information**

Hardcover: 657 pages

Publisher: McGraw-Hill Science/Engineering/Math; 2 edition (August 1, 1996)

Language: English

ISBN-10: 0070004846


Product Dimensions: 6 x 1.4 x 9 inches

Shipping Weight: 2.3 pounds

Average Customer Review: 3.8 out of 5 stars – See all reviews (222 customer reviews)

Best Sellers Rank: #414,615 in Books (See Top 100 in Books) #23 in Books > Computers & Technology > Programming > Languages & Tools > Lisp #227 in Books > Computers & Technology > Hardware & DIY > Design & Architecture #522 in Books > Textbooks > Computer Science > Database Storage & Design

**Customer Reviews**

I think it's fascinating that there is such a split between those who love and hate this book. For most books, the review is a bell-shaped curve of star ratings; this one has a peak at 1, a peak at 5, and very little in between. How could this be? I think it is because SICP is a very personal message that works only if the reader is at heart a computer scientist (or willing to become one). So I agree that the book's odds of success are better if you read it after having some experience. To use an analogy, if SICP were about automobiles, it would be for the person who wants to know how cars work, how they are built, and how one might design fuel-efficient, safe, reliable vehicles for the 21st century. The people who hate SICP are the ones who just want to know how to drive their car on the highway, just like everyone else. Those who hate SICP think it doesn't deliver enough tips and tricks for the amount of time it takes to read. But if you're like me, you're not looking for one more trick, rather you're looking for a way of synthesizing what you already know, and building a rich framework onto which you can add new learning over a career. That's what SICP has done for me. I
read a draft version of the book around 1982, when I was in grad school, and it changed the way I think about my profession. If you're a thoughtful computer scientist (or want to be one), it will change your life too. Some of the reviewers complain that SICP doesn't teach the basics of OO design, and so on. In a sense they are right. The book doesn't directly tell you how to design and write an object-oriented program using the subset of object-oriented principles that show up in the syntax of Java or C++.