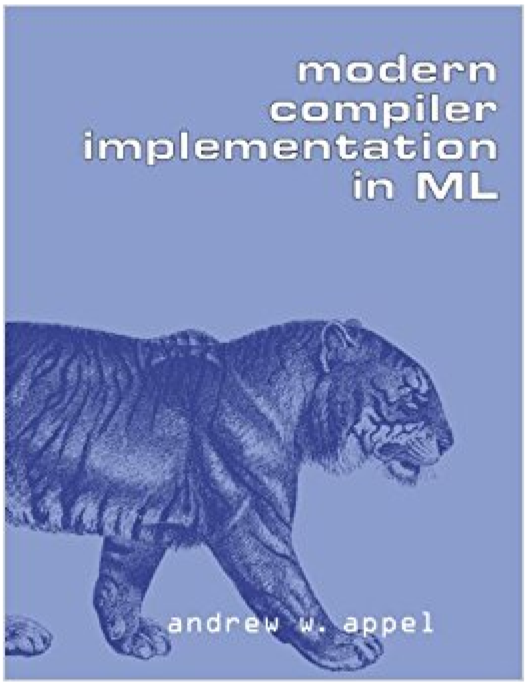
Modern Compiler Implementation In ML



Synopsis

This new, expanded textbook describes all phases of a modern compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as functional and object-oriented languages, that are missing from most books. In addition, more advanced chapters are now included so that it can be used as the basis for two-semester or graduate course. The most accepted and successful techniques are described in a concise way, rather than as an exhaustive catalog of every possible variant. Detailed descriptions of the interfaces between modules of a compiler are illustrated with actual C header files. The first part of the book, Fundamentals of Compilation, is suitable for a one-semester first course in compiler design. The second part, Advanced Topics, which includes the advanced chapters, covers the compilation of object-oriented and functional languages, garbage collection, loop optimizations, SSA form, loop scheduling, and optimization for cache-memory hierarchies.

Book Information

File Size: 18273 KB

Print Length: 548 pages

Simultaneous Device Usage: Up to 4 simultaneous devices, per publisher limits

Publisher: Cambridge University Press (December 13, 1997)

Publication Date: December 13, 1997

Sold by:Â Digital Services LLC

Language: English

ASIN: B00D2WQAE8

Text-to-Speech: Enabled

X-Ray: Not Enabled

Word Wise: Not Enabled

Lending: Not Enabled

Enhanced Typesetting: Not Enabled

Best Sellers Rank: #588,647 Paid in Kindle Store (See Top 100 Paid in Kindle Store)   #63 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compiler Design
#175 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compilers
#709 inÂ Kindle Store > Kindle eBooks > Computers & Technology > Programming > Software

Design > Software Development

Customer Reviews

This book covers a remarkably broad territory of compiler issues, with a good balance of clarity and depth. It spends long enough on each topic to work through examples of functioning code, but doesn’t dwell on any for so long that the reader loses interest. A decent set of variations on language design, runtime organization, and machine architecture are also discussed and implemented.An important caveat is that readers ought to know some dialect of ML. If not, this book does not teach it; there are alternative renditions of the same text in C and Java, but I have not read them. Apparently many reviewers find those confusing. The java rendition, for example, has many angry reviewers. Possibly disgruntled students who had a hard time with their coursework, possibly people with valid complaints about the text.All I can say is that I quite enjoyed it, and return to it regularly for its clear presentation, when working with related algorithms.

I love this book. Having experimented with compiler construction for some time (though never seriously), and being a long-time ML user, Modern Compiler Implementation in ML was the perfect companion to my compiler-construction project. I largely ignored the "Tiger" language described in the book - I had my own ideas about what I wanted to create. Relying on the book to give you code for a whole compiler is asking far too much. The general concepts are explained, and then reinforced with sample code. I suspect the people who have complained bitterly about this book are actually just looking for a source code print out.All in all, very worthwhile. Read a chapter, try to implement the concepts, and then either re-read the same chapter, or move on.

Appel explains the structure of a compiler in a very clear and precise fashion, including a lot of quite modern techniques. *However* it is only clear and precise if you have a rough high-level understanding of compilers and a good knowledge of ML beforehand. (If you understand one of the two, it is an excellent book for learning the other but you will need to do some thinking for yourself.)If you’re looking to learn about compilers from the very beginning, buy something else (I liked "Crafting A Compiler" as a starting point). If you’re wanting to get a lot more precise about the way you build compilers, buy this book.

This book aims at teaching the reader how to write compilers and interpreters. Compared to other books (most notably the Dragon book - Aho, Sethi & Ullman: Compilers - Principles, Techniques and Tools) the structure and organization of the compiler he teaches are better and more in sync with modern compiler research. So although ML is not a language that is used much in practice it is well choosen for teaching compilers.The first part of the book is intended for undergraduate Computer Science university student and the second part is for graduate students. I have only read the first part and browsed the second part. The first part is founded on the development of a compiler - the Tiger compiler - and that is the main flaw of the book. The Tiger compiler is fairly advanced and it is built up one phase at the time (see the chapter headings in the synopsis). A undergraduate reader who has never read anything about compiler design before will be very confused. The book is good in the way that it presents many good compiler design ideas and I learned some from it, but for me it was not the first book on compiler. I just think that the first part of the book has too steep a learning curve - even if you are proficient with ML, which is an absolute must. Recommended for serious readers who already know something about compiler construction before reading it. For other people I recommend the older but somewhat outdated book: Aho, Sethi & Ullman: Compilers - Principles, Techniques and Tools.

Appel very clearly describes, step by step, the implementation of a compiler, and explains as well the reasons why ML is a good language to use. If you don’t like ML (which is, incidentally, a "real language"), there are Java and C versions of the book.

I find this a nice textbook for a first compiler class. However, more than the book I find the author’s assignments, available on his course website on Princeton, very valuable. The assignments are well thought out, and well written. They help you pace the lectures very well. Search for "Princeton COS 320".

Appel’s stated goal in writing this book was, rather than to provide a laundry list of compiler algorithms that could potentially be used to implement a compiler, to use the algorithms that now dominate modern compilers. His thorough and authoritative approach starts with building a complete working compiler in the first half of the book before exploring deeper specialized topics in the last half of the book. Each chapter comes with example ML code from the chapter along with programming exercises which build off of the chapter’s code so that at the end you have built a working optimizing compiler.The book is an enjoyable read the whole way through. I particularly enjoyed the chapters on Instruction Selection, Register Allocation, Functional Languages, Loop Optimizations, and the Memory Hierarchy. This book gave me a new appreciation for how sophisticated a compiler's job is. If you're looking for an introduction or refresher on compilers, start with this book.