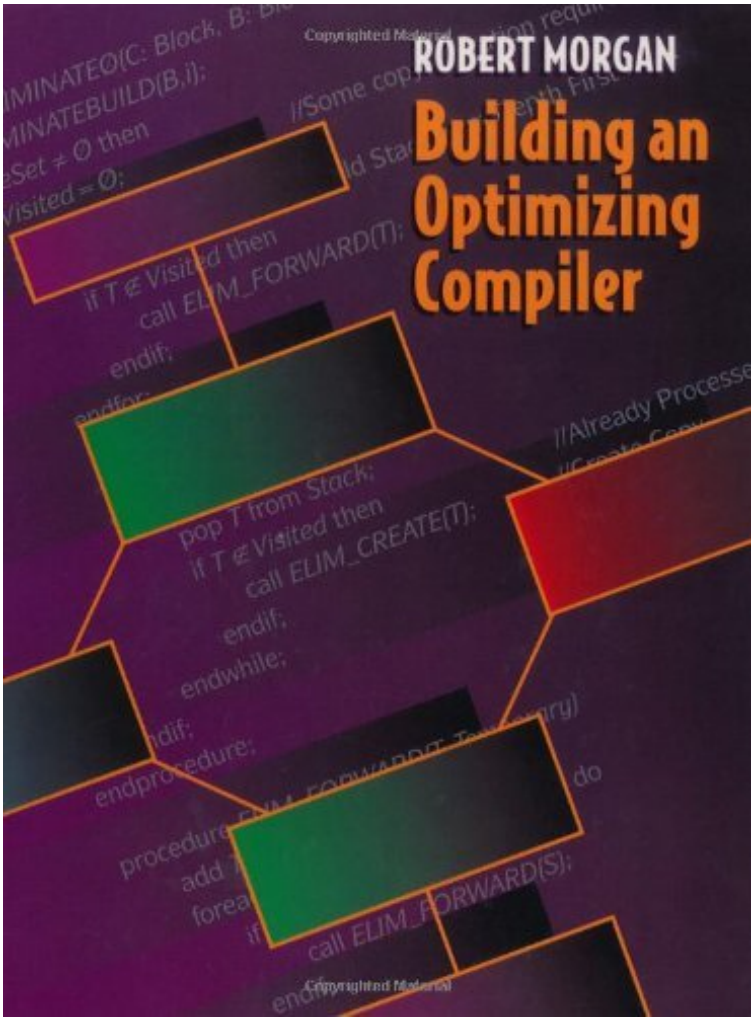
Building an Optimizing Compiler

Von Bob Morgan
*ePub | *DOC | audiobook | ebooks | Download PDF*



Produktinformation -Verkaufsrang: #937829 in eBooksVerffentlicht am: 1998-02-04Erscheinungsdatum: 1998-02-04File Name: B008COCE9G | File size: 57.Mb

Von Bob Morgan : Building an Optimizing Compiler before purchasing it in order to gage whether or not it would be worth my time, and all praised Building an Optimizing Compiler:

KundenrezensionenHilfreichste Kundenrezensionen1 von 1 Kunden fanden die folgende Rezension hilfreich. a code-free approach to *optimizing* compilersVon Ein KundeMost students who have taken a semester course in compiler would agree that a semester is too short even for just building a compiler, not to include the time used to learn the basics like parsing. This book is a good start at where your course left off. It covers many interesting and modern topics on building an _optimizing_ compiler, a compiler that generates compact and speedy code.As a comparison to the one Muchnick, this book is slightly less advanced but contains sufficient details to start one in this field. This book also excels in its clear and informative explanations.An interesting feature of this book (and also Muchnick) is that no

concrete code/implementation is included. The focus is on the concepts of building an optimizing compiler and the theory behind code optimization, not exactly on how to build one (from scratch) using whatever programming language. The reader must come up with the implementation side of the story if she wants to apply the techniques to her own compiler.As a final note, this is not an introductory book on compilers. If you want one, go find the classic Dragon book (by Aho, Sethi, Ullman) or the newly written work by Appel.

KurzbeschreibungCompiler courses have been an academic mainstay in both undergraduate and graduate computer science departments. These courses emphasize either the implementation of languages or parallel processing. Little analysis is presented for generating excellent code for a language on a single processor. Building an Optimizing Compiler addresses this gap. It provides a high level design for a thorough optimizer, code generator, scheduler and register allocator for a generic modern RISC processor. In the process it addresses the small issues that have a long impact on the implementation.The book approaches this subject from a practical viewpoint. Theory is introduced where intuitive arguments are insufficient, however the theory is described in practical terms. A single running example is used throughout the book to illustrate the compilation process. Provides a complete theory for Static Single Aassignment Methods and partial redundancy methods for code optimization Provides a new generatization of register allocation techniques Techniques described are applicable to most programming languages for PCs, workstations or serversKurzbeschreibungCompiler courses have been an academic mainstay in both undergraduate and graduate computer science departments. These courses emphasize either the implementation of languages or parallel processing. Little analysis is presented for generating excellent code for a language on a single processor. Building an Optimizing Compiler addresses this gap. It provides a high level design for a thorough optimizer, code generator, scheduler and register allocator for a generic modern RISC processor. In the process it addresses the small issues that have a long impact on the implementation.The book approaches this subject from a practical viewpoint. Theory is introduced where intuitive arguments are insufficient, however the theory is described in practical terms. A single running example is used throughout the book to illustrate the compilation process. Provides a complete theory for Static Single Aassignment Methods and partial redundancy methods for code optimization Provides a new generatization of register allocation techniques Techniques described are applicable to most programming languages for PCs, workstations or serversSynopsisProvides a high level design for an optimizer, code generator, scheduler and register allocator for a generic modern RISC processor. A single running example is used throughout the book to illustrate the compilation process.