Preface

This book is intended as a text for a one-term introductory course in compiler writing at a senior undergraduate level. It maintains a balance between a theoretical and practical approach to this subject. From a theoretical viewpoint, it introduces rudimental models underlying compilation and its essential phases. Based on these models, it demonstrates the concepts, methods, and techniques employed in compilers. It also sketches the mathematical foundations of compilation and related topics, including the theory of formal languages, automata, and transducers. Simultaneously, from a practical point of view, this book describes how the compiler techniques are implemented. Running throughout the book, a case study designs a new Pascal-like programming language and constructs its compiler; while discussing various methods concerning compilers, the case study illustrates their implementation. Additionally, many detailed examples and computer programs are presented to emphasize the actual applications of the compilation algorithms. Essential software tools are also covered. After studying this book, the student should be able to grasp the compilation process, write a simple real compiler, and follow advanced books on the subject.

From a logical standpoint, the book divides compilation into six cohesive phases. At the same time, it points out that a real compiler does not execute these phases in a strictly consecutive manner; instead, their execution somewhat overlaps to speed up and enhance the entire compilation process as much as possible. Accordingly, the book covers the compilation process phase by phase while simultaneously explaining how each phase is connected during compilation. It describes how this mutual connection is reflected in the compiler construction to achieve the most effective compilation as a whole.

On the part of the student, no previous knowledge concerning compilation is assumed. Although this book is self-contained, in the sense that no other sources are needed for understanding the material, a familiarity with an assembly language and a high-level language, such as Pascal or C, is helpful for quick comprehension. Every new concept or algorithm is preceded by an explanation of its purpose and followed by some examples, computer program passages, and comments to reinforce its understanding. Each complicated material is preceded by its intuitive explanation. All applications are given in a quite realistic way to clearly demonstrate a strong relation between the theoretical concepts and their uses.

In computer science, strictly speaking, every algorithm requires a verification that it terminates and works correctly. However, the termination of the algorithms given in this book is always so obvious that its verification is omitted throughout. The correctness of complicated algorithms is verified in detail. On the other hand, we most often give only the gist of the straightforward algorithms and leave their rigorous verification as an exercise. The text describes the algorithms in Pascal-like notation, which is so simple and intuitive that even the student unfamiliar with Pascal can immediately pick it up. In this description, a Pascal **repeat** loop is sometimes ended with **until no change**, meaning that the loop is repeated until no change can result from its further repetition. As the clear comprehensibility is a paramount importance in the book, the description of algorithms is often enriched by an explanation in words.

Algorithms, conventions, definitions, lemmas, and theorems are sequentially numbered within chapters and are ended with ■. Examples and figures are analogously organized. At the end of each chapter, a set of exercises is given to reinforce and augment the material covered. Selected exercises, denoted by ***Solved*** in the text, have their solutions at the chapter's conclusion. The appendix contains a C++ implementation of a substantial portion of a real compiler. Further backup materials, including lecture notes, teaching tips, homework assignments, errata, exams, solutions, programs, and implementation of compilers, are available at

http://www.fit.vutbr.cz/~meduna/books/eocd