

Preface

Compiler design is a subject which many believe to be fundamental and vital to computer science. It is a subject which has been studied intensively since the early 1950's and continues to be an important research field today. Compiler design is an important part of the undergraduate curriculum for many reasons: (1) It provides students with a better understanding of and appreciation for programming languages. (2) The techniques used in compilers can be used in other applications with command languages. (3) It provides motivation for the study of theoretic topics. (4) It is a good vehicle for an extended programming project.

There are several compiler design textbooks available today, but most have been written for graduate students. Here at Rowan University (formerly Glassboro State College), our students have had difficulty reading these books. However, I felt it was not the subject matter that was the problem, but the way it was presented. I was sure that if concepts were presented at a slower pace, with sample problems and diagrams to illustrate the concepts, that our students would be able to master the concepts. This is what I have attempted to do in writing this book.

This book is a revision of an earlier edition that was written for a Pascal based curriculum. As many computer science departments have moved to C++ as the primary language in the undergraduate curriculum, I have produced this edition to accommodate those departments. This book is not intended to be strictly an object-oriented approach to compiler design.

The most essential prerequisites for this book are courses in C or C++ programming, Data Structures, Assembly Language or Computer Architecture, and possibly Programming Languages. If the student has not studied formal languages and automata, this book includes introductory sections on these theoretic topics, but in this case it is not likely that all seven chapters will be covered in a one semester course. Students who have studied the theory will be able to skip the preliminary sections (2.0, 3.0, 4.0) without loss of continuity.

The concepts of compiler design are applied to a case study which is an implementation of a subset of C which I call MiniC. Chapters 2, 4, 5, and 6 include a section devoted to explaining how the relevant part of the MiniC compiler is designed. This public domain software is presented in full in the appendices and is available on the Internet. Students can benefit by enhancing or changing the MiniC compiler provided.

Chapters 6 and 7 focus on the back end of the compiler (code generation and optimization). Here I rely on a fictitious computer, called Mini, as the target machine. I use a fictitious machine for three reasons: (1) I can design it for simplicity so that the compiler design concepts are not obscured by architectural requirements, (2) It is available to anyone who has a C compiler (the Mini simulator, written in C, is available also), and (3) the teacher or student can modify the Mini machine to suit his/her tastes.

Chapter 7 includes only a brief description of optimization techniques since there is not enough time in a one semester course to delve into these topics, and because these are typically studied in more detail at the graduate level.

To use the software that accompanies this book, you will need access to the world wide web. The source files can be accessed at
<http://www.rowan.edu/~bergmann/books/minic>

Once you have stored the programs on your computer, the programs can be generated with the makefile on unix/linux:

```
> make
```

Additional description of these files can be found in Appendix B.

I wish to acknowledge the people who participated in the design of this book. The reviewers of the original Pascal version – James E. Miller of Transylvania University, Jeffrey C. Chang of Garner-Webb University, Stephen J. Allan of Utah State University, Karsten Henckell of the New College of USF, and Keith Olson of Montana Technical College – all took the time to read through various versions of the manuscript and provided many helpful suggestions. My students in the Compiler Design course here at Rowan University also played an important role in testing the original version and subsequent versions of this book. Support in the form of time and equipment was provided by the administration of Rowan University.

The pages of this book were composed entirely by me using Adobe Pagemaker, and diagrams were drawn with Microsoft Excel and Powerpoint.

Finally, I am most grateful to Sue, Aaron, and Sarah, for being so understanding during the time that I spent working on this project.

Seth D. Bergmann
bergmann@rowan.edu