© PROSPECT PRESS, FOR REVIEWING PURPOSES ONLY

Preface

Mission of the text

Assembly Programming and Computer Architecture for Software Engineers is an educational examination of Assembly programming and computer architecture. We approach these topics from a practical point of view, addressing *why* and *how* questions throughout the text. We begin by laying the foundation of computer language and computer architecture, and then we delve into Assembly programming as a mechanism for gaining a better understanding of computer architecture, and how Assembly can be used for software development.

Most of the existing books on computer architecture have one or more of the following disadvantages: (1) based on a non-mainstream architecture; (2) written for computer and electrical engineers as opposed to computer scientists and software engineers; (3) focused on a single platform-specific development environment; (4) over-priced; and (5) lacking in practical content. We wanted to give our students something better.

Audience

Assembly Programming and Computer Architecture for Software Engineers is primarily intended for undergraduate students in computer science and software engineering programs. Prerequisites for this book include introductory computing courses and a solid programming foundation up to and including data structures, preferably in C/C++. Working professionals are also likely to find this book helpful for independent learning and for writing both low-level and high-level code.

Book Development and Pedagogical Approach

The notion to write a book arises every so often in an academic career, with one driver for book ideas being the courses we teach. Both of us teach a course on computer architecture, which has been a staple of computer science and software engineering programs for decades. We also teach in programs where applied and practical skills are foremost.

During our doctoral studies, we made passing comments about writing a book together and at some point we decided that doing so was a real possibility. Writing a book takes a lot of effort, thus our collaboration was essential. So after several years teaching computer architecture and discussing ideas with our wonderful Prospect Press partner, Beth Golub, we arrived at a basic but unique concept. We would write a book for teachers, students, and professionals seeking educational content for computer architecture that overcomes the previously stated disadvantages. Our book is based on mainstream architectures, written for computer scientists and software engineers, written for multiple development environments, well priced, and loaded with practical content.

Many hours, days, and weeks have been poured into this text and, in particular, the programs. Writing Assembly code can be infuriating, compelling, and fun, as you are about to discover. Writing about such a complex topic in a clear and efficient manner was also a particular challenge, as was choosing what aspects to cover and when—a classic issue with computing content.

For many computing students, learning about computer architecture via Assembly programming is an advantageous approach. While learning about architecture, useful programming skills are fostered. We knew that a book based on such an approach would help in our courses and was likely to provide a much-needed resource for similar institutions, educators, and students. We did our best and we hope you enjoy learning about Assembly programming and computer architecture the way we enjoy teaching the topics.

Organization and Objectives

Chapters 1 and 2 discuss computer language and computer architecture fundamentals.
Chapters 3 through 5 introduce x86/x86_64 Assembly syntax and a variety of instructions.
Chapters 6 through 8 cover the more complex topics of functions, structures, and floating-point operations.

© PROSPECT PRESS, FOR REVIEWING PURPOSES ONLY

Chapters 9 and 10 show advanced ways to use Assembly with high-level languages and system software, as well as introduce other advanced aspects of computer and system architecture.
Chapter 11 explores architectures other than x86/x86_64
Chapter 12 introduces basic hardware principles and components.

Chapter Objectives

Chapter 1: Describe computer language translation
Chapter 2: Identify computer and processor components
Chapter 3: Distinguish between Assembly syntaxes
Chapter 4: Perform basic arithmetic
Chapter 5: Control program flow
Chapter 6: Follow function calling conventions
Chapter 7: Use strings and structures
Chapter 8: Execute floating-point operations
Chapter 9: Integrate low-level and high-level code
Chapter 10: Issue system calls
Chapter 11: Compare computer architectures
Chapter 12: Build simple circuits and devices

Chapter Supplements

Chapters 1 and 2, Chapter 6, Chapter 8, and Chapter 10 have additional supplements that provide programs and content for the respective chapters.

Appendices

The **Introduction to the Appendices** and **Appendices A through I** provide practical information such as resources, translating between Assembly syntaxes, development environment setup, disassembling, debugging, linking Assembly and C++, following calling conventions, using CPUID, performing Decimal and ASCII arithmetic, and using intrinsics.

What is New in Edition 2.0?

We have made additions, modifications, and corrections to the text.

General Updates

- Moved from a 32-bit first approach to a 64-bit first approach. The primary programs and examples in most cases are x86_64.
- Added programs on GitHub, which now offers three variants per assembler where appropriate: x86, x86/64, and x86_64. The x86 versions are the 32-bit programs from Edition 1. The x86/64 versions mix 32-bit and 64-bit code and register usage and are direct translations of the x86 versions to be x86_64 compatible. The x86_64 versions mainly use x86_64 code and registers.
- Added an assignment for most chapters. All chapters now contain four assignments. Note that some assignment numbering has changed.
- Updated chapter slides and solutions for instructors.
- Updated links for web resources.

Chapter Specific Updates Worth Noting

- Chapters 3-5: x86_64 is the focus of the syntax and instruction fundamentals.
- Chapter 6: The function walkthrough follows the x86_64 calling convention, while the x86 conventions are provided for context.
- Chapter 9: Incorporates compiler intrinsics. Covering Appendix I: Intrinsics prior to Chapter 9 is helpful.

Resources

- Book website: https://www.prospectpressvt.com/textbooks/hall-assembly-programming-2-0
- Book repository: https://github.com/brianrhall/Assembly

© PROSPECT PRESS, FOR REVIEWING PURPOSES ONLY

- Instructor resources: https://www.prospectpressvt.com/textbooks/hall-assembly-programming-2-0/instructor-resources-hall/
- Student resources: https://www.prospectpressvt.com/textbooks/hall-assembly-programming-2-0
- Brian's website: http://www.brianrhall.com
- Kevin's website: http://www.kevinslonka.com
- For content-specific resources see the **Appendices** (particularly the **Introduction to the Appendices**) and the **Web Resources** links at the beginning of each chapter.

Acknowledgements

We owe a debt of thanks to the people who have been supportive of our efforts in writing this book. Thank you to the *teachers* who have led us to where we are in our careers. Thank you to *Beth Golub* for giving us the opportunity to publish with Prospect Press and for her wonderful guidance throughout the writing process. Thank you to our *reviewers* for taking time to provide valuable feedback. They include:

John Doyle, Indiana University Southeast
Kent Einspahr, Concordia University
Hai Jiang, Arkansas State University
Saad Khattak, Tuque Games
Susan Lincke, University of Wisconsin - Parkside
Amanda Smith, Arizona Western College
Ray Toal, Loyola Marymount University
David Topham, Ohlone College
Renaat Verbruggen, Dublin City University & Champlain College
John Zamora, Modesto Junior College

Thank you to our *colleagues* and *students* for putting up with us clamoring so much about the book over the past several years. And most of all, thank you to our *families*. Without your tolerance and willingness to sacrifice precious time together, this work would have not been possible. Thank you.

Brian and Kevin

© PROSPECT PRESS, FOR REVIEWING PURPOSES ONLY