Preface

The objective of this book is to introduce assembly language programming. Assembly language is very closely linked to the underlying processor architecture and design. Popular processor designs can be broadly divided into two categories: Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC). The dominant processor in the PC market, Pentium, belongs to the CISC category. However, the recent design trend is to use the RISC designs. Some example RISC processors include the MIPS, SPARC, PowerPC, and ARM. Even Intel's 64-bit processor Itanium is a RISC processor. Thus, both types of processors are important candidates for our study.

This book covers assembly language programming of both CISC and RISC processors. We use the Intel Pentium processor as the representative of the CISC category. We have selected the Pentium processor because of its market dominance. To explore RISC assembly language, we selected the MIPS processor. The MIPS processor is appealing as it closely adheres to the RISC principles. Furthermore, the availability of the SPIM simulator allows us to use a Pentium-based PC to learn MIPS assembly language.

New in the Second Edition

The second edition has been substantially revised to reflect the changes that have taken place since the publication of the first edition. The major changes are listed below:

- We introduced RISC assembly language programming so that the reader can benefit from learning both CISC and RISC assembly languages. As mentioned before, Pentium and MIPS processors are used to cover CISC and RISC processors.

- The first edition used MASM/TASM assemblers. In this edition, we use the NASM assembler. The syntax of NASM is slightly different from that of MASM/TASM assemblers. The advantage is that NASM is free! Another advantage is that it works with both Microsoft Windows and Linux operating systems.
- Consistent with our shift to NASM, we moved away from DOS to Linux. Since NASM is available for Windows and Linux, most of the programs in this book can be used with either Windows or Linux. However, we clearly indicate our preference to Linux. This preference is exposed in chapters like “High-Level Language Interface” that deal with mixed-mode programming involving C and assembly language. For example, in Chapter 17, we use the GNU C compiler (`gcc`) rather than the Microsoft or Borland C compiler. Similarly, in Appendix C we use the GNU debugger (`gdb`) to explore the debugging process.
- The “Basic Computer Organization” chapter (Chapter 2) has been completely rewritten to give a general background on computer organization. The Pentium processor details are moved to a new chapter (Chapter 4).
- A completely new chapter has been added to discuss Pentium’s protected mode interrupt processing.
- We have added a new chapter on recursion. This chapter discusses how we can implement recursive procedures in the Pentium and MIPS assembly languages.
- We have augmented the Pentium assembly language programming by describing its floating-point instructions. This entire chapter is new in this edition.

In addition to these major changes, all chapters have gone through extensive revision. Some chapters have been reorganized to eliminate the duplication present in the first edition.

Intended Use

Assembly language programming is part of several undergraduate curricula in computer science, computer engineering, and electrical engineering departments. This book can be used as a text for those courses that teach assembly language.

It can also be used as a companion text in a computer organization course for teaching the assembly language. Because we cover both CISC and RISC processors, the instructor can select the assembly language that best fits her or his course.

In addition, it can be used as a text in vocational training courses offered by community colleges. Because of the teach-by-example style used in the book, it is also suitable for self-study by computer professionals and engineers.

Instructional Support

The book’s Web site (`www.scs.carleton.ca/~sivarama/asm_book`) has complete chapter-by-chapter PowerPoint slides for instructors. Instructors can use these slides directly in their classes or can modify them to suit their needs. In addition, instructors can obtain the solutions manual by contacting the publisher. For more up-to-date details, please see the book's Web site.

Prerequisites

The student is assumed to have had some experience in a structured, high-level language such as C. However, the book does not assume extensive knowledge of any high-level language—only the basics are needed. Furthermore, it is assumed that the student has a rudimentary background in the software development cycle, as is obtained in a typical high-level programming course.

Features

Here is a summary of the special features that sets this book apart:

- This is probably the only book to cover the assembly language programming of both CISC and RISC processors.
- This book uses NASM and Linux as opposed to scores of other books that use MASM and Windows.
- The book is self-contained and does not assume a background in computer organization. All necessary background material on computer organization is presented in the book.
- This book contains a methodical organization of chapters for a step-by-step introduction to the assembly language.
- Extensive examples are used in each chapter to illustrate the points discussed in the chapter. Our objective is not just to explain how an instruction works but also to provide the rationale as to why the instruction has been designed the way it is.
- Procedures are introduced early on to encourage modular programming in developing assembly language programs.
- A set of input and output routines is provided so that the student can focus on developing assembly language programs rather than spending time in understanding how the input and output are done using the basic I/O functions provided by the operating system.
- This book does not use fragments of code in examples. All examples are complete in the sense that they can be assembled and run, giving a better feeling as to how these programs work.
- All examples and other required software are available from the book's Web site (`www.scs.carleton.ca/~sivarama/asm_book`) to give opportunities for students to perform hands-on assembly programming.
- Most chapters are written in such a way that each chapter can be covered in two or three 60-minute lectures by giving proper reading assignments. Typically, important

concepts are emphasized in the lectures while leaving the other material as a reading assignment. Our emphasis on extensive examples facilitates this pedagogical approach.

- Interchapter dependencies are kept to a minimum to offer maximum flexibility to instructors in organizing the material. Each chapter clearly indicates the objectives and provides an overview at the beginning and a summary at the end.
- Each chapter contains two types of exercises—review and programming—to reinforce the concepts discussed in the chapter.
- The appendices provide special reference material that contains a thorough treatment of various topics.

Overview and Organization

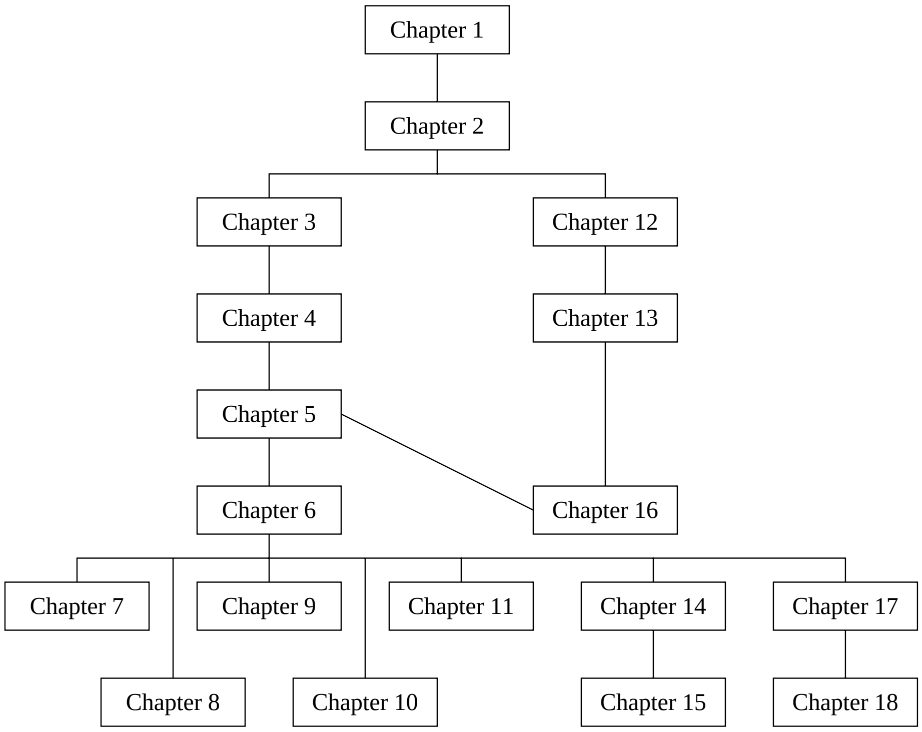
The 18 chapters in the book are divided into 6 parts (see the figure on the next page for chapter dependencies). Part I presents introductory topics and consists of the first two chapters. Chapter 1 provides introduction to the assembly language and gives reasons for programming in the assembly language. Chapter 2 presents the basics of computer organization with a focus on three system components: processor, memory, and I/O.

Part II is dedicated to Pentium assembly language programming. It consists of nine chapters—Chapters 3 through 11. This part begins with a description of the Pentium processor organization (Chapter 3). In particular, this chapter gives sufficient details on the 16- and 32-bit Intel processors so that the student can effectively program in the assembly language. Chapter 4 gives an overview of the assembly language. After covering these two chapters, one can write simple standalone assembly language programs.

To emphasize the importance of modular programming, procedures are introduced early on (in Chapter 5). The other chapters in this part expand on the overview given in Chapter 4. Chapter 6 presents the addressing modes supported by the Intel 16- and 32-bit processors. This chapter also contains a detailed discussion on the motivation for providing the various addressing modes. Addressing modes are one of the differentiating characteristics of CISC processors. Chapter 7 discusses the arithmetic instructions and the use of the flags register. Chapters 8 and 9 present conditional and bit manipulation instructions. A feature of these two chapters is that they relate how high-level language statements can be implemented using the instructions discussed in these two chapters. Chapter 10 discusses the string processing instructions in detail. ASCII and BCD arithmetic instructions are presented in Chapter 11.

The first four chapters of this part—Chapters 3 to 6—should be covered in some detail for proper grounding in assembly language programming. However, the remaining five chapters can be studied in any order. In addition, the depth at which these five chapters are covered can be varied without sacrificing the effectiveness, depending on the time available and importance to the course objective.

Part III is dedicated to the MIPS assembly language programming. Chapter 12 describes the RISC design principles; it also covers MIPS processor details. The MIPS assembly language is presented in Chapter 13. This chapter also gives details on the SPIM simulator. All



the programming examples given in this chapter can be run on a Pentium-based PC using the SPIM simulator. The SPIM simulator details are given in Appendix D.

Part IV focuses on Pentium's interrupt processing mechanism. We cover both protected-mode and real-mode interrupt processing. Chapter 14 gives details on protected-mode interrupt processing. This chapter uses Linux system calls to facilitate our discussion of software interrupts. The next chapter discusses the real-mode interrupt processing. This is the only chapter that uses DOS to explore how programmed I/O and interrupt-driven I/O are done.

The remaining 3 of the 18 chapters constitute Part V. These chapters deal with advanced topics. Chapter 16 focuses on how recursive procedures are implemented in Pentium and MIPS assembly languages. The next chapter deals with the high-level language interface, which allows mixed-mode programming. We use C and assembly language to cover the principles involved in mixed-mode programming. The last chapter discusses Pentium's floating-point instructions. To follow the programming examples of this chapter, you need to understand the high-level language interface details presented in Chapter 17.

The seven appendices provide a wealth of reference material the student needs. Appendix A primarily discusses the number systems and their internal representation. Appendix B gives information on the use of I/O routines provided with this book and the assembler software. The debugging aspect of assembly language programming is discussed in Appendix C. The SPIM simulator details are given in Appendix D. Selected Pentium and MIPS instructions are given in Appendices E and F, respectively. Finally, Appendix G gives the standard ASCII table.

Acknowledgments

Several people have contributed, either directly or indirectly, in writing this book. First and foremost, I would like to thank my family for enduring my preoccupation with this project. My heartfelt thanks to Sobha and Veda for their understanding and patience!

I want to thank Ann Kostant, Executive Editor at Springer, for her positive feedback on the proposal for the revision. A very special thanks to Wayne Wheeler, Associate Editor, for handling various aspects of the project in a timely manner. I would also like to express my appreciation to the staff at the Springer production department for converting my camera-ready copy into the book in front of you.

I also express my appreciation to the School of Computer Science at Carleton University for providing a great atmosphere to complete this book.

Feedback

Works of this nature are never error-free, despite the best efforts of the authors, editors, and others involved in the project. I welcome your comments, suggestions, and corrections by electronic mail.

Ottawa, Canada
January 2004

Sivarama Dandamudi
`sivarama@scs.carleton.ca`
`http://www.scs.carleton.ca/~sivarama`