Preface

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 trend is to use the RISC designs. Even Intel has moved from CISC to RISC design for their 64-bit processor. The main objective of this book is to provide a guide to the architecture and assembly language of the popular RISC processors. In all, we cover five RISC designs in a comprehensive manner.

To explore RISC assembly language, we selected the MIPS processor, which is pedagogically appealing as it closely adheres to the RISC principles. Furthermore, the availability of the SPIM simulator allows us to use a PC to learn the MIPS assembly language.

Intended Use

This book is intended for computer professionals and university students. Anyone who is interested in learning about RISC processors will benefit from this book, which has been structured so that it can be used for self-study. The reader 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.

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 companion text in those courses that teach assembly language.

Features

Here is a summary of the special features that set this book apart.

- This probably is the only book on the market to cover five popular RISC architectures: MIPS, SPARC, PowerPC, Itanium, and ARM.
- There is a methodical organization of chapters for a step-by-step introduction to the MIPS assembly language.
- 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.
- Source code for the MIPS assembly language program examples is available from the book's Web site (`www.scs.carleton.ca/˜sivarama/risc_book`).
- The book is self-contained and does not assume a background in computer organization. All necessary background material is presented in the book.
- Interchapter dependencies are kept to a minimum to offer maximum flexibility to instructors in organizing the material. Each chapter provides an overview at the beginning and a summary at the end.
- An extensive set of programming exercises is provided to reinforce the MIPS assembly language concepts discussed in Part III of the book.

Overview and Organization

We divide the book into four parts. Part I presents introductory topics and consists of the first three chapters. Chapter 1 provides an introduction to CISC and RISC architectures. In addition, it introduces assembly language and gives reasons for programming in assembly language. The next chapter discusses processor design issues including the number of addresses used in processor instructions, how flow control is altered by branches and procedure calls, and other instruction set design issues. Chapter 3 presents the RISC design principles.

The second part describes several RISC architectures. In all, we cover five architectures: MIPS, PowerPC, SPARC, Itanium, and ARM. For each architecture, we provide many details on its instruction set. Our discussion of MIPS in this part is rather brief because we devote the entire Part III to its assembly language.

The third part, which consists of nine chapters, covers the MIPS assembly language. This part allows you to get hands-on experience in writing the MIPS assembly language programs. You don't need a MIPS-based system to do this! You can run these programs on your PC using the SPIM simulator. Our thanks go to Professor James Larus for writing the simulator, for which we provide details on installation and use.

The last part consists of several appendices. These appendices give reference information on various number systems, character representation, and the MIPS instruction set. In addition, we also give several programming exercises so that you can practice writing MIPS assembly language programs.

Acknowledgments

Several people have contributed, either directly or indirectly, to the writing of this book. First and foremost, I would like to thank Sobha and Veda for their understanding and patience!

I want to thank Ann Kostant, Executive Editor and Wayne Wheeler, Associate Editor, both at Springer, for their enthusiastic support for the project. 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, 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.

Carleton University
Ottawa, Canada
April 2004

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