

Guide to RISC Processors

Sivarama P. Dandamudi

Guide to RISC Processors

for Programmers and Engineers

Sivarama P. Dandamudi
School of Computer Science
Carleton University
Ottawa, ON K1S 5B6
Canada
sivarama@scs.carleton.ca

Library of Congress Cataloging-in-Publication Data

Dandamudi, Sivarama P., 1955–

Guide to RISC processors / Sivarama P. Dandamudi.

p. cm.

Includes bibliographical references and index.

ISBN 0-387-21017-2 (alk. paper)

1. Reduced instruction set computers. 2. Computer architecture. 3. Assembler language
(Computer program language) 4. Microprocessors—Programming. I. Title.

QA76.5.D2515 2004

004.3—dc22

2004051373

ISBN 0-387-21017-2

Printed on acid-free paper.

© 2005 Springer Science+Business Media, Inc.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, Inc., 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed in the United States of America. (HAM)

9 8 7 6 5 4 3 2 1

SPIN 10984949

springeronline.com

To
my parents, **Subba Rao** and **Prameela Rani**,
my wife, **Sobha**,
and
my daughter, **Veda**

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>

Contents

Preface	vii
PART I: Overview	1
1 Introduction	3
Processor Architecture	3
RISC Versus CISC	5
What Is Assembly Language?	7
Advantages of High-Level Languages	9
Why Program in Assembly Language?	10
Summary	11
2 Processor Design Issues	13
Introduction	13
Number of Addresses	14
The Load/Store Architecture	20
Processor Registers	22
Flow of Control	22
Procedure Calls	26
Handling Branches	28
Instruction Set Design Issues	32
Summary	36
3 RISC Principles	39
Introduction	39
Evolution of CISC Processors	40
Why RISC?	41
RISC Design Principles	43
Summary	44

PART II: Architectures	45
4 MIPS Architecture	47
Introduction	47
Registers	48
Register Usage Convention	48
Addressing Modes	50
Instruction Format	51
Memory Usage	52
Summary	53
5 SPARC Architecture	55
Introduction	55
Registers	56
Addressing Modes	58
Instruction Format	59
Instruction Set	59
Procedures and Parameter Passing	69
Summary	76
6 PowerPC Architecture	79
Introduction	79
Register Set	81
Addressing Modes	83
Instruction Format	84
Instruction Set	86
Summary	96
7 Itanium Architecture	97
Introduction	97
Registers	98
Addressing Modes	100
Procedure Calls	101
Instruction Format	102
Instruction-Level Parallelism	105
Instruction Set	106
Handling Branches	112
Speculative Execution	114
Branch Prediction Hints	119
Summary	119

8	ARM Architecture	121
	Introduction	121
	Registers	123
	Addressing Modes	125
	Instruction Format	128
	Instruction Set	131
	Summary	145
	PART III: MIPS Assembly Language	147
9	SPIM Simulator and Debugger	149
	Introduction	149
	Simulator Settings	152
	Running a Program	153
	Debugging	154
	Summary	157
10	Assembly Language Overview	159
	Introduction	159
	Assembly Language Statements	160
	SPIM System Calls	161
	SPIM Assembler Directives	162
	MIPS Program Template	165
	Data Movement Instructions	165
	Load Instructions	166
	Store Instructions	167
	Addressing Modes	167
	Sample Instructions	168
	Our First Program	172
	Illustrative Examples	174
	Summary	182
11	Procedures and the Stack	183
	Introduction	183
	Procedure Invocation	186
	Returning from a Procedure	188
	Parameter Passing	189
	Our First Program	189
	Stack Implementation in MIPS	192
	Parameter Passing via the Stack	196
	Illustrative Examples	200
	Passing Variable Number of Parameters	207
	Summary	210

12 Addressing Modes	211
Introduction	211
Addressing Modes	212
Processing Arrays	214
Our First Program	217
Illustrative Examples	219
Summary	224
13 Arithmetic Instructions	225
Introduction	225
Addition	226
Subtraction	226
Multiplication	228
Division	229
Our First Program	230
Illustrative Examples	232
Summary	242
14 Conditional Execution	243
Introduction	243
Comparison Instructions	244
Unconditional Branch Instructions	246
Conditional Branch Instructions	248
Our First Program	249
Illustrative Examples	252
Indirect Jumps	259
Indirect Procedures	262
Summary	267
15 Logical and Shift Operations	269
Introduction	269
Logical Instructions	270
Shift Instructions	276
Rotate Instructions	280
Our First Program	281
Illustrative Examples	284
Summary	290
16 Recursion	291
Introduction	291
Our First Program	292
Illustrative Examples	295

Recursion Versus Iteration	303
Summary	304
17 Floating-Point Operations	305
Introduction	305
FPU Registers	306
Floating-Point Instructions	307
Our First Program	312
Illustrative Examples	314
Summary	322
Appendices	323
A Number Systems	325
Positional Number Systems	325
Conversion to Decimal	327
Conversion from Decimal	328
Binary/Octal/Hexadecimal Conversion	329
Unsigned Integers	330
Signed Integers	331
Floating-Point Representation	334
Summary	336
B Character Representation	339
Character Representation	339
ASCII Character Set	340
C MIPS Instruction Set Summary	343
D Programming Exercises	365
Bibliography	375
Index	379

PART I

Overview