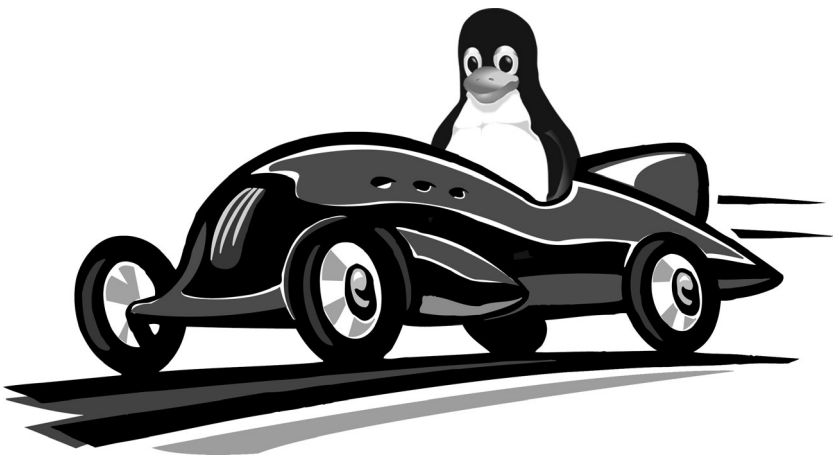
See MIPS® Run

Second Edition



See MIPS® Run

Second Edition

Dominic Sweetman

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Morgan Kaufmann Publishers is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

Publisher: Denise E. M. Penrose
Publishing Services Manager: George Morrison
Senior Project Manager: Brandy Lilly
Editorial Assistant: Kimberlee Honjo
Cover Design: Alisa Andreola and Hannus Design
Composition: diacriTech
Technical Illustration: diacriTech
Copyeditor: Denise Moore
Proofreader: Katherine Antonsen
Indexer: Steve Rath
Interior Printer: The Maple-Vail Book Manufacturing Group, Inc.
Cover Printer: Phoenix Color

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Application Submitted

ISBN 13: 978-0-12-088421-6
ISBN 10: 0-12-088421-6

For information on all Morgan Kaufmann publications,
visit our Web site at *www.mkp.com* or *www.books.elsevier.com*

Printed in the United States of America
06 07 08 09 10 5 4 3 2 1

Foreword

The MIPS architecture was born in the early 1980s from the work done by John Hennessy and his students at Stanford University. They were exploring the architectural concept of RISC (Reduced Instruction Set Computing), which theorized that relatively simple instructions, combined with excellent compilers and hardware that used pipelining to execute the instructions, could produce a faster processor with less die area. The concept was so successful that MIPS Computer Systems was formed in 1984 to commercialize the MIPS architecture.

Over the course of the next 14 years, the MIPS architecture evolved in a number of ways and its implementations were used very successfully in workstation and server systems. Over that time, the architecture and its implementations were enhanced to support 64-bit addressing and operations, support for complex memory-protected operating systems such as UNIX, and very high performance floating point. Also in that period, MIPS Computer Systems was acquired by Silicon Graphics and MIPS processors became the standard for Silicon Graphics computer systems. With 64-bit processors, high-performance floating point, and the Silicon Graphics heritage, MIPS processors became the solution of choice in high-volume gaming consoles.

In 1998, MIPS Technologies emerged from Silicon Graphics as a standalone company focused entirely on intellectual property for embedded markets. As a result, the pace of architecture development has increased to address the unique needs of these markets: high-performance computation, code compression, geometry processing for graphics, security, signal processing, and multithreading. Each architecture development has been matched by processor core implementations of the architecture, making MIPS-based processors the standard for high-performance, low-power applications.

The MIPS legacy in complex systems such as workstations and servers directly benefits today's embedded systems, which have, themselves, become very complex. A typical embedded system is composed of multiple processing elements, high-performance memory, and one or more operating systems.

When compared with other embedded architectures, which are just now learning what is required to build a complex system, the MIPS architecture provides a proven base on which to implement such systems.

In many ways, the first edition of *See MIPS Run* was a ground-breaking book on the MIPS architecture and its implementations. While other books covered similar material, *See MIPS Run* focused on what the programmer needed to understand of the architecture and the software environment in order to effectively program a MIPS chip.

Increasing complexity of embedded systems has been matched by enhancements to the MIPS architecture to address the needs of such systems. The second edition of this book is required reading for any current developer of MIPS-based embedded systems. It adds significant new material, including the architectural standardization of the MIPS32 and MIPS64 architectures, brand new application-specific extensions such as multithreading, and a very nice treatment of the implementation of the popular Linux operating system on the MIPS architecture. Short of the MIPS architecture specifications, the second edition of *See MIPS Run* is the most current description of the state of the art of the architecture and is, bar none, the most readable.

I hope that you will find this as worthwhile and as entertaining to read as I did.

Michael Uhler,
Chief Technology Officer, MIPS Technologies, Inc.
Mountain View, CA
May 2006

Contents

Preface

This book is about MIPS, the cult hit from the mid-1980s' crop of RISC CPU designs. These days MIPS is not the highest-volume 32-bit architecture, but it is in a comfortable second place. Where it wins, hands down, is its range of applications. A piece of equipment built around a MIPS CPU might have cost you $35 for a wireless router or hundreds of thousands of dollars for an SGI supercomputer (though with SGI's insolvency, those have now reached the end of the line). Between those extremes are Sony and Nintendo games machines, many Cisco routers, TV set-top boxes, laser printers, and so on.

The first edition of this book has sold close to 10,000 English copies over the years and has been translated into Chinese. I'm pleased and surprised; I didn't know there were so many MIPS programmers out there.

This second edition is *See MIPS Run . . . Linux*. The first edition struggled to motivate some features of the MIPS architecture, because they don't make sense unless you can see how they help out inside an OS kernel. But now a lot of you have some sense of how Linux works, and I can quote its source code; more importantly, I can refer to it knowing that those of you who get interested can read the source code and find out how it's *really* done.

So this is a book about the MIPS architecture, but the last three chapters stroll through the Linux kernel and application-programming system to cast light on what those weird features do. I hope Linux experts will forgive my relative ignorance of Linux details, but the chance to go for a description of a real OS running on a real architecture was too good to pass up.

MIPS is a RISC: a useful acronym, well applied to the common features of a number of computer architectures invented in the 1980s, to realize efficient pipelined implementation. The acronym CISC is vaguer. I'll use it in a narrow sense, for the kind of features found in x86 and other pre-1982 architectures, designed with microcoded implementations in mind.

Some of you may be up in arms: He's confusing implementation with architecture! But while computer architecture is supposed to be a contract with the

programmer about what programs will run correctly, it's also an engineering design in its own right. A computer architecture is designed to make for good CPUs. As chip design becomes more sophisticated, the trade-offs change.

This book is for programmers, and that's the test we've used to decide what gets included—if a programmer might see it, or is likely to be interested, it's here. That means we don't get to discuss, for example, the strange system interfaces with which MIPS has tortured two generations of hardware design engineers. And your operating system may hide many of the details we talk about here; there is many an excellent programmer who thinks that C is quite low level enough, portability a blessing, and detailed knowledge of the architecture irrelevant. But sometimes you do need to get down to the nuts and bolts—and human beings are born curious as to how bits of the world work.

A result of this orientation is that we'll tend to be rather informal when describing things that may not be familiar to a software engineer—particularly the inner workings of the CPU—but we'll get much more terse and technical when we're dealing with the stuff programmers have met before, such as registers, instructions, and how data is stored in memory.

We'll assume some familiarity and comfort with the C language. Much of the reference material in the book uses C fragments as a way of compressing operation descriptions, particularly in the chapters on the details of the instruction set and assembly language.

Some parts of the book are targeted at readers who've seen some assembly language: the ingenuity and peculiarity of the MIPS architecture shows up best from that viewpoint. But if assembly is a closed book to you, that's probably not a disaster.

This book aims to tell you everything you need to know about programming generic MIPS CPUs. More precisely, it describes the architecture as it's defined by MIPS Technologies' MIPS32 and MIPS64—specifically, the second release of those specifications from 2003. We'll shorten that to "MIPS32/64." But this is not just a reference manual: To keep an architecture in your head means coming to understand it in the round. I also hope the book will interest students of programming (at college or enrolled in the school of life) who want to understand a modern CPU architecture all the way through.

If you plan to read this book straight through from front to back, you will expect to find a progression from overview to detail, and you won't be disappointed. But you'll also find some progression through history; the first time we talk about a concept we'll usually focus on its first version. Hennessy and Patterson call this "learning through evolution," and what's good enough for them is certainly good enough for me.

We start in Chapter 1 with some history and background, and set MIPS in context by discussing the technological concerns and ideas that were uppermost in the minds of its inventors. Then in Chapter 2 we discuss the characteristics of the MIPS machine language that follow from their approach.

To help you see the big picture, we leave out the details of processor control until Chapter 3, which introduces the ugly but eminently practical system that allows MIPS CPUs to deal with their caches, exceptions and startup, and memory management. Those last three topics, respectively, become the subjects of Chapters 4 through 6.

The MIPS architecture has been careful to separate out the part of the instruction set that deals with floating-point numbers. That separation allows MIPS CPUs to be built with various levels of floating-point support, from none at all through partial implementations to somewhere near the state of the art. So we have also separated out the floating-point functions, and we keep them back until Chapter 7.

Up to this point, the chapters follow a reasonable sequence for getting to know MIPS. The following chapters change gear and are more like reference manuals or example-based tutorials.

In Chapter 8, we go through the whole machine instruction set; the intention is to be precise but much more terse than the standard MIPS reference works—we cover in 10 pages what takes a hundred in other sources.[1] Chapter 9 is a brief introduction to reading and writing assembly, and falls far short of an assembly programming manual.

Chapter 10 is a checklist with helpful hints for those of you who have to port software between another CPU and a MIPS CPU. The longest section tackles the troublesome problem of endianness in CPUs, software, and systems.

Chapter 11 is a bare-bones summary of the software conventions (register use, argument passing, etc.) necessary to produce interworking software with different toolkits. Chapter 12 introduces the debug and profiling features standardized for MIPS CPUs.

Then we're on to seeing how MIPS runs GNU/Linux. We describe relationship between the Linux kernel and a computer architecture in Chapter 13; then Chapters 14 and 15 dig down into some of the detail as to how the MIPS architecture does what the Linux kernel needs. Chapter 16 gives you a quick look at the dynamic linking magic that makes GNU/Linux applications work.

Appendix A covers the MIPS MT (multithreading) extension, probably the most important addition to the architecture in many years. And Appendix B describes the more important add-ons: MIPS16, the new MIPS DSP extensions, and MDMX.

You will also find at the end of this book a glossary of terms—a good place to look for specialized or unfamiliar usage and acronyms—and a list of books, papers, and online references for further reading.

---

1. I have taken considerable care in the generation of these tables, and they are mostly right. But if your system depends on it, be sure to cross-check this information. An excellent source of fairly reliable information can be found in the behavior and source code of the GNU tool collection—but I referred to that too, so it's not completely independent.

Style and Limits

Every book reflects its author, so we'd better make a virtue of it.

Since some of you will be students, I wondered whether I should distinguish general use from MIPS use. I decided not to; I aim to be specific except where it costs the reader nothing to be general. I also try to be concrete rather than abstract. I don't worry overmuch about whatever meaning terms like "TLB" have in the wider industry, but I explain them in a MIPS context. Human beings are great generalizers, and this is unlikely to damage your learning much.

It's 20 years since I started working with MIPS CPUs in the fall of 1986. Some of the material in this book goes back as far as 1988, when I started giving training courses on MIPS architecture. In 1993, I gathered them together to make a software manual focused on IDT's R3051 family CPUs. It took quite a lot of extra material to create the first edition, published in 1999.

A lot has happened since 1999. MIPS is now at the very end of its life in servers with SGI but has carved out a significant niche in embedded systems. Linux has emerged as the most-used OS for embedded MIPS, but there's still a lot of diversity in the embedded market. The MIPS specifications have been reorganized around MIPS32 and MIPS64 (which this edition regards as the baseline). This second edition has been in the works for about three years.

The MIPS story continues; if it did not, we'd only be writing this book for historians, and Morgan Kaufmann wouldn't be very interested in publishing it. MIPS developments that weren't announced by the end of 2005 are much too late for this edition.

Conventions

A quick note on the typographical conventions used in this book:

- Type in this font (Minion) is running text.
- Type in this font (Futura) is a sidebar.
- **Type in this font (Courier bold) is used for assembly code and MIPS register names.**
- `Type in this font (Courier) is used for C code and hexadecimals.`
- *Type in this font (Minion italic, small) is used for hardware signal names.*

Acknowledgments

The themes in this book have followed me through my computing career. Mike Cole got me excited about computing, and I've been trying to emulate his skill in picking out good ideas ever since. In the brief but exciting life of Whitechapel Workstations (1983–1988), many colleagues taught me something about computer architecture and about how to design hardware—Bob Newman and Rick Filipkiewicz probably the most. I also have to thank Whitechapel's salesperson, Dave Gravell, for originally turning me on to MIPS. My fellow engineers during the lifetime of Algorithmics Ltd. (Chris Dearman, Rick Filipkiewicz, Gerald Onions, Nigel Stephens, and Chris Shaw) have to be doubly thanked, both for all I've learned through innumerable discussions, arguments, and designs and for putting up with the book's competition for my time.

Many thanks are due to the reviewers who've read chapters over a long period of time: Phil Bourekas of Integrated Device Technology, Inc.; Thomas Daniel of the LSI Logic Corporation; Mike Murphy of Silicon Graphics, Inc.; and David Nagle of Carnegie Mellon University.

On the second edition: I've known Paul Cobb for a long time, as we both worked around MIPS companies. Paul contributed material updating the historical survey of MIPS CPUs and the programming chapter and cleaning up the references. In all cases, though, I've had a final edit—so any errors are mine.

During the preparation of this edition I've been employed by MIPS Technologies Inc. It's dangerous to pick out some colleagues and not others, but I'll do it anyway.

Ralf Baechle runs the `www.linux-mips.org` site, which coordinates MIPS work on the Linux kernel. He's been very helpful in dispelling some of the illusions I'd formed about Linux: I started off thinking it was like other operating systems . . . (Robert Love's *Linux Kernel Development* book helped too; I warmly recommend it to anyone who wants a more educated guidebook to the kernel). Thanks to MIPS Technologies and my various managers for being flexible about my time, and to many colleagues at MIPS Technologies (too many to name) who have read and commented on drafts.

Todd Bezenek has been my most persistent colleague/reviewer of this edition. Reviewers outside MIPS Technologies did it for their love and respect for the field: notable contributers were Steven Hill (Reality Diluted, Inc.), Jun Sun (DoCoMo USA Labs), Eric DeVolder, and Sophie Wilson.

Denise Penrose is easily the Best Editor Ever. Not many people in Finsbury Park (my home in North London) can say they're just flying to San Francisco for brunch with their publisher.

Last but not least, thanks to Carol O'Brien, who was rash enough to marry me in the middle of this rewrite.