ELSEVIER

COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

THIRD EDITION 3

DAVID A. PATTERSON
JOHN L. HENNESSY

MORGAN KAUFMANN

Senior Editor	Denise E. M. Penrose
Publishing Services Manager	Simon Crump
Editorial Assistant	Summer Block
Cover Design	Ross Caron Design
Cover and Chapter Illustration	Chris Asimoudis
Text Design	GGS Book Services
Composition	Nancy Logan and Dartmouth Publishing, Inc.
Technical Illustration	Dartmouth Publishing, Inc.
Copyeditor	Ken DellaPenta
Proofreader	Jacqui Brownstein
Indexer	Linda Buskus
Interior printer	Courier
Cover printer	Courier

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: 1-55860-604-1

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

Printed in the United States of America
04 05 06 07 08 5 4 3 2 1

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data

①

CORE INSTRUCTION SET

NAME	MNE-MON-IC	FOR-MAT	OPERATION (in Verilog)		OPCODE/ FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	$0 / 20_{hex}$
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1)(2)	8_{hex}
Add Imm. Unsigned	addiu	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		$0 / 21_{hex}$
And	and	R	R[rd] = R[rs] & R[rt]		$0 / 24_{hex}$
And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	j	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	jal	J	R[31]=PC+4;PC=JumpAddr	(5)	3_{hex}
Jump Register	jr	R	PC=R[rs]		$0 / 08_{hex}$
Load Byte Unsigned	lbu	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	$0 / 24_{hex}$
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	$0 / 25_{hex}$
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	$0 / 23_{hex}$
Nor	nor	R	R[rd] = ~ (R[rs] | R[rt])		$0 / 27_{hex}$
Or	or	R	R[rd] = R[rs] | R[rt]		$0 / 25_{hex}$
Or Immediate	ori	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		$0 / 2a_{hex}$
Set Less Than Imm.	slti	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	sltiu	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2)(6)	b_{hex}
Set Less Than Unsigned	sltu	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	$0 / 2b_{hex}$
Shift Left Logical	sll	R	R[rd] = R[rs] << shamt		$0 / 00_{hex}$
Shift Right Logical	srl	R	R[rd] = R[rs] >> shamt		$0 / 02_{hex}$
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Halfword	sh	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	$0 / 22_{hex}$
Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		$0 / 23_{hex}$

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1b'0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2 s comp.)

BASIC INSTRUCTION FORMATS

R	opcode	rs	rt	rd	shamt	funct
	31 26	25 21	20 16	15 11	10 6	5 0
I	opcode	rs	rt	immediate		
	31 26	25 21	20 16	15 0		
J	opcode	address				
	31 26	25 0				

ARITHMETIC CORE INSTRUCTION SET

②

NAME	MNE-MON-IC	FOR-MAT	OPERATION		OPCODE/ FMT / FT/ FUNCT (Hex)
Branch On FP True	bc1t	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	bc1f	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	div	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	add.s	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	add.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	c.*x*.s*	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/*y*
FP Compare Double	c.*x*.d*	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/*y*
* (*x* is eq, lt, or le) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)					
FP Divide Single	div.s	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	div.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	mul.s	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	mul.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	sub.s	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	sub.d	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	lwc1	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	ldc1	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	mfhi	R	R[rd] = Hi		0 /--/--/10
Move From Lo	mflo	R	R[rd] = Lo		0 /--/--/12
Move From Control	mfc0	R	R[rd] = CR[rs]		16 /0/--/0
Multiply	mult	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	multu	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Store FP Single	swc1	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	sdc1	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING POINT INSTRUCTION FORMATS

FR	opcode	fmt	ft	fs	fd	funct
	31 26	25 21	20 16	15 11	10 6	5 0
FI	opcode	fmt	ft	immediate		
	31 26	25 21	20 16	15 0		

PSEUDO INSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	bgt	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	ble	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	bge	if(R[rs]>=R[rt]) PC = Label
Load Immediate	li	R[rd] = immediate
Move	move	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes

OPCODES, BASE CONVERSION, ASCII SYMBOLS

3

MIPS opcode (31:26)	(1) MIPS funct (5:0)	(2) MIPS funct (5:0)	Binary	Decimal	Hexa-decimal	ASCII Character	Decimal	Hexa-decimal	ASCII Character
(1)	sll	add.*f*	00 0000	0	0	NUL	64	40	@
		sub.*f*	00 0001	1	1	SOH	65	41	A
j	srl	mul.*f*	00 0010	2	2	STX	66	42	B
jal	sra	div.*f*	00 0011	3	3	ETX	67	43	C
beq	sllv	sqrt.*f*	00 0100	4	4	EOT	68	44	D
bne		abs.*f*	00 0101	5	5	ENQ	69	45	E
blez	srlv	mov.*f*	00 0110	6	6	ACK	70	46	F
bgtz	srav	neg.*f*	00 0111	7	7	BEL	71	47	G
addi	jr		00 1000	8	8	BS	72	48	H
addiu	jalr		00 1001	9	9	HT	73	49	I
slti	movz		00 1010	10	a	LF	74	4a	J
sltiu	movn		00 1011	11	b	VT	75	4b	K
andi	syscall	round.w.*f*	00 1100	12	c	FF	76	4c	L
ori	break	trunc.w.*f*	00 1101	13	d	CR	77	4d	M
xori		ceil.w.*f*	00 1110	14	e	SO	78	4e	N
lui	sync	floor.w.*f*	00 1111	15	f	SI	79	4f	O
	mfhi		01 0000	16	10	DLE	80	50	P
(2)	mthi		01 0001	17	11	DC1	81	51	Q
	mflo	movz.*f*	01 0010	18	12	DC2	82	52	R
	mtlo	movn.*f*	01 0011	19	13	DC3	83	53	S
			01 0100	20	14	DC4	84	54	T
			01 0101	21	15	NAK	85	55	U
			01 0110	22	16	SYN	86	56	V
			01 0111	23	17	ETB	87	57	W
	mult		01 1000	24	18	CAN	88	58	X
	multu		01 1001	25	19	EM	89	59	Y
	div		01 1010	26	1a	SUB	90	5a	Z
	divu		01 1011	27	1b	ESC	91	5b	[
			01 1100	28	1c	FS	92	5c	\
			01 1101	29	1d	GS	93	5d	]
			01 1110	30	1e	RS	94	5e	^
			01 1111	31	1f	US	95	5f	_
lb	add	cvt.s.*f*	10 0000	32	20	Space	96	60	`
lh	addu	cvt.d.*f*	10 0001	33	21	!	97	61	a
lwl	sub		10 0010	34	22	"	98	62	b
lw	subu		10 0011	35	23	#	99	63	c
lbu	and	cvt.w.*f*	10 0100	36	24	$	100	64	d
lhu	or		10 0101	37	25	%	101	65	e
lwr	xor		10 0110	38	26	&	102	66	f
	nor		10 0111	39	27	'	103	67	g
sb			10 1000	40	28	(	104	68	h
sh			10 1001	41	29	)	105	69	i
swl	slt		10 1010	42	2a	*	106	6a	j
sw	sltu		10 1011	43	2b	+	107	6b	k
			10 1100	44	2c	,	108	6c	l
			10 1101	45	2d	-	109	6d	m
swr			10 1110	46	2e	.	110	6e	n
cache			10 1111	47	2f	/	111	6f	o
ll	tge	c.f.*f*	11 0000	48	30	0	112	70	p
lwc1	tgeu	c.un.*f*	11 0001	49	31	1	113	71	q
lwc2	tlt	c.eq.*f*	11 0010	50	32	2	114	72	r
pref	tltu	c.ueq.*f*	11 0011	51	33	3	115	73	s
	teq	c.olt.*f*	11 0100	52	34	4	116	74	t
ldc1		c.ult.*f*	11 0101	53	35	5	117	75	u
ldc2	tne	c.ole.*f*	11 0110	54	36	6	118	76	v
		c.ule.*f*	11 0111	55	37	7	119	77	w
sc		c.sf.*f*	11 1000	56	38	8	120	78	x
swc1		c.ngle.*f*	11 1001	57	39	9	121	79	y
swc2		c.seq.*f*	11 1010	58	3a	:	122	7a	z
		c.ngl.*f*	11 1011	59	3b	;	123	7b	{
		c.lt.*f*	11 1100	60	3c	<	124	7c	|
sdc1		c.nge.*f*	11 1101	61	3d	=	125	7d	}
sdc2		c.le.*f*	11 1110	62	3e	>	126	7e	~
		c.ngt.*f*	11 1111	63	3f	?	127	7f	DEL

(1) opcode(31:26) == 0

(2) opcode(31:26) == 17_{ten} (11_{hex}); if fmt(25:21)==16_{ten} (10_{hex}) *f* = s (single); if fmt(25:21)==17_{ten} (11_{hex}) *f* = d (double)

IEEE 754 FLOATING POINT STANDARD

4

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent - Bias})}$$

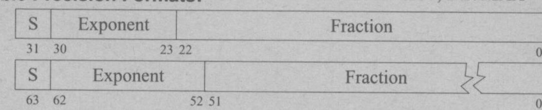
where Single Precision Bias = 127,
Double Precision Bias = 1023.

IEEE 754 Symbols

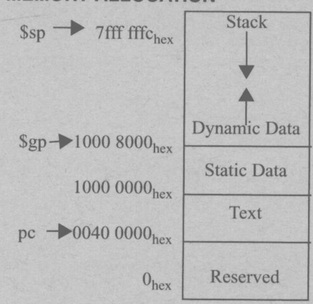
Exponent	Fraction	Object
0	0	± 0
0	≠0	± Denorm
1 to MAX - 1	anything	± Fl. Pt. Num.
MAX	0	±∞
MAX	≠0	NaN

S.P. MAX = 255, D.P. MAX = 2047

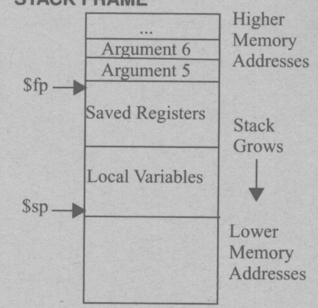
IEEE Single Precision and Double Precision Formats:



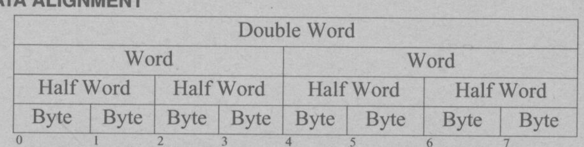
MEMORY ALLOCATION



STACK FRAME

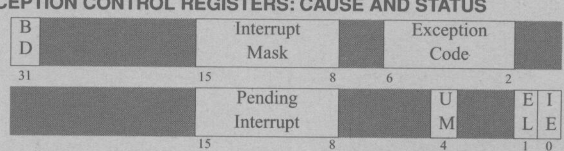


DATA ALIGNMENT



Value of three least significant bits of byte address (Big Endian)

EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS



BD = Branch Delay, UM = User Mode, EL = Exception Level,IE =Interrupt Enable

EXCEPTION CODES

Number	Name	Cause of Exception	Number	Name	Cause of Exception
0	Int	Interrupt (hardware)	9	Bp	Breakpoint Exception
4	AdEL	Address Error Exception (load or instruction fetch)	10	RI	Reserved Instruction Exception
5	AdES	Address Error Exception (store)	11	CpU	Coprocessor Unimplemented
6	IBE	Bus Error on Instruction Fetch	12	Ov	Arithmetic Overflow Exception
7	DBE	Bus Error on Load or Store	13	Tr	Trap
8	Sys	Syscall Exception	15	FPE	Floating Point Exception

SIZE PREFIXES (10^x for Disk, Communication; 2^x for Memory)

SIZE	PREFIX	SIZE	PREFIX	SIZE	PREFIX	SIZE	PREFIX
$10^3, 2^{10}$	Kilo-	$10^{15}, 2^{50}$	Peta-	10^{-3}	milli-	10^{-15}	femto-
$10^6, 2^{20}$	Mega-	$10^{18}, 2^{60}$	Exa-	10^{-6}	micro-	10^{-18}	atto-
$10^9, 2^{30}$	Giga-	$10^{21}, 2^{70}$	Zetta-	10^{-9}	nano-	10^{-21}	zepto-
$10^{12}, 2^{40}$	Tera-	$10^{24}, 2^{80}$	Yotta-	10^{-12}	pico-	10^{-24}	yocto-

The symbol for each prefix is just its first letter, except μ is used for micro.

MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

THIRD EDITION

Computer Organization Design

THE HARDWARE/SOFTWARE INTERFACE

ACKNOWLEDGEMENTS

Figures 1.9, 1.15 Courtesy of Intel.

Figure 1.11 Courtesy of Storage Technology Corp.

Figures 1.7.1, 1.7.2, 6.13.2 Courtesy of the Charles Babbage Institute, University of Minnesota Libraries, Minneapolis.

Figures 1.7.3, 6.13.1, 6.13.3, 7.9.3, 8.11.2 Courtesy of IBM.

Figure 1.7.4 Courtesy of Cray Inc.

Figure 1.7.5 Courtesy of Apple Computer, Inc.

Figure 1.7.6 Courtesy of the Computer History Museum.

Figure 7.33 Courtesy of AMD.

Figures 7.9.1, 7.9.2 Courtesy of Museum of Science, Boston.

Figure 7.9.4 Courtesy of MIPS Technologies, Inc.

Figure 8.3 ©Peg Skorpinski.

Figure 8.11.1 Courtesy of the Computer Museum of America.

Figure 8.11.3 Courtesy of the Commercial Computing Museum.

Figures 9.11.2, 9.11.3 Courtesy of NASA Ames Research Center.

Figure 9.11.4 Courtesy of Lawrence Livermore National Laboratory.

Computers in the Real World:

Photo of "A Laotian villager," courtesy of David Sanger.

Photo of an "Indian villager," property of Encore Software, Ltd., India.

Photos of "Block and students" and "a pop-up archival satellite tag," courtesy of Professor Barbara Block. Photos by Scott Taylor.

Photos of "Professor Dawson and student" and "the Mica micromote," courtesy of AP/World Wide Photos.

Photos of "images of pottery fragments" and "a computer reconstruction," courtesy of Andrew Willis and David B. Cooper, Brown University, Division of Engineering.

Photo of "the Eurostar TGV train," by Jos van der Kolk.

Photo of "the interior of a Eurostar TGV cab," by Andy Veitch.

Photo of "firefighter Ken Whitten," courtesy of World Economic Forum.

Graphic of an "artificial retina," © The San Francisco Chronicle. Reprinted by permission.

Image of "A laser scan of Michelangelo's statue of David," courtesy of Marc Levoy and Dr. Franca Falletti, director of the Galleria dell'Accademia, Italy.

"An image from the Sistine Chapel," courtesy of Luca Pezzati. IR image recorded using the scanner for IR reflectography of the INOA (National Institute for Applied Optics, http://arte.ino.it) at the Opificio delle Pietre Dure in Florence.

Preface

The most beautiful thing we can experience is the mysterious.
It is the source of all true art and science.

Albert Einstein, *What I Believe*, 1930

About This Book

We believe that learning in computer science and engineering should reflect the current state of the field, as well as introduce the principles that are shaping computing. We also feel that readers in every specialty of computing need to appreciate the organizational paradigms that determine the capabilities, performance, and, ultimately, the success of computer systems.

Modern computer technology requires professionals of every computing specialty to understand both hardware and software. The interaction between hardware and software at a variety of levels also offers a framework for understanding the fundamentals of computing. Whether your primary interest is hardware or software, computer science or electrical engineering, the central ideas in computer organization and design are the same. Thus, our emphasis in this book is to show the relationship between hardware and software and to focus on the concepts that are the basis for current computers.

The audience for this book includes those with little experience in assembly language or logic design who need to understand basic computer organization as well as readers with backgrounds in assembly language and/or logic design who want to learn how to design a computer or understand how a system works and why it performs as it does.

About the Other Book

Some readers may be familiar with *Computer Architecture: A Quantitative Approach*, popularly known as Hennessy and Patterson. (This book in turn is called Patterson and Hennessy.) Our motivation in writing that book was to describe the principles of computer architecture using solid engineering fundamentals and quantitative cost/performance trade-offs. We used an approach that combined examples and measurements, based on commercial systems, to create realistic design experiences. Our goal was to demonstrate that computer architecture could be learned using quantitative methodologies instead of a descriptive approach. It is intended for the serious computing professional who wants a detailed understanding of computers.

A majority of the readers for this book do not plan to become computer architects. The performance of future software systems will be dramatically affected, however, by how well software designers understand the basic hardware techniques at work in a system. Thus, compiler writers, operating system designers, database programmers, and most other software engineers need a firm grounding in the principles presented in this book. Similarly, hardware designers must understand clearly the effects of their work on software applications.

Thus, we knew that this book had to be much more than a subset of the material in *Computer Architecture*, and the material was extensively revised to match the different audience. We were so happy with the result that the subsequent editions of *Computer Architecture* were revised to remove most of the introductory material; hence, there is much less overlap today than with the first editions of both books.

Changes for the Third Edition

We had six major goals for the third edition of *Computer Organization and Design:* make the book work equally well for readers with a software focus or with a hardware focus; improve pedagogy in general; enhance understanding of program performance; update the technical content to reflect changes in the industry since the publication of the second edition in 1998; tie the ideas from the book more closely to the real world *outside* the computing industry; and reduce the size of this book.

First, the table on the next page shows the hardware and software paths through the material. Chapters 1, 4, and 7 are found on both paths, no matter what the experience or the focus. Chapters 2 and 3 are likely to be review material for the hardware-oriented, but are essential reading for the software-oriented, especially for those readers interested in learning more about compilers and object-oriented programming languages. The first sections of Chapters 5 and 6 give overviews for those with a software focus. Those with a hardware focus, however, will find that these chapters present core material; they may also, depending on background, want to read Appendix B on logic design first and the sections on microprogramming and how to use hardware description languages to specify control. Chapter 8 on input/output is key to readers with a software focus and should be read if time permits by others. The last chapter on multiprocessors and clusters is again a question of time for the reader. Even the history sections show this balanced focus; they include short histories of programming languages, compilers, numerical software, operating systems, networking protocols, and databases.

Chapter or Appendix	Sections	Software Focus	Hardware Focus
1. Computer Abstractions and Technology	1.1 to 1.6	Read carefully	Read carefully
	1.7 (History)	Read for culture	Read for culture
2. Instructions: Language of the Computer	2.1 to 2.11	Read carefully	Review or read
	2.12 (Compilers)	Read if have time	
	2.13 (C sort)	Read carefully	Review or read
	2.14 (Java)	Read if have time	
	2.15 to 2.18	Read carefully	Review or read
	2.19 (History)	Read for culture	Read for culture
3. Arithmetic for Computers	3.1 to 3.11	Read carefully	Review or read
	3.12 (History)	Read for culture	Read for culture
D. RISC instruction set architectures	D.1 to D.19	Read if have time	
4. Assessing and Understanding Performance	4.1 to 4.6	Read carefully	Read carefully
	4.7 (History)	Read for culture	Read for culture
B. The Basics of Logic Design	B.1 to B.13		Review or read
5. The Processor: Datapath and Control	5.1 (Overview)	Read carefully	Read carefully
	5.2 to 5.7		Read carefully
	5.8 (Microcode)		Read if have time
	5.9 (Verilog)		Read carefully
	5.10 to 5.12	Read carefully	Read carefully
	5.13 (History)	Read for culture	Read for culture
C. Mapping Control to Hardware	C.1 to C.6		Read if have time
6. Enhancing Performance with Pipelining	6.1 (Overview)	Read carefully	Read carefully
	6.2 to 6.6		Read carefully
	6.7 (verilog)		Read carefully
	6.8 to 6.9		Read carefully
	6.10 to 6.12	Read carefully	Read carefully
	6.13 (History)	Read for culture	Read for culture
7. Large and Fast: Exploiting Memory Hierarchy	7.1 to 7.8	Read carefully	Read carefully
	7.9 (History)	Read for culture	Read for culture
8. *Storage, Networks, and Other Peripherals*	8.1 to 8.2	Read carefully	Read if have time
	8.3 (Networks)	Read if have time	Read if have time
	8.4 to 8.10	Read carefully	Read if have time
	8.13 (History)	Read for culture	Read for culture
9. Multiprocessors and Clusters	9.1 to 9.10	Read if have time	
	9.11 (History)	Read for culture	
A. Assemblers, Linkers, and the SPIM Simulator	A.1 to A.12	Reference	Reference
Computers in the Real World	Between Chapters	Read for culture	Read for culture

Read carefully · Review or read · Read if have time · Read for culture · Reference

The next goal was to improve the exposition of the ideas in the book, based on difficulties mentioned by readers of the second edition. We added five new book elements to help. To make the book work better as a reference, we placed definitions of new terms in the margins at their first occurrence. We hope this will help readers find the sections when they want to refer back to material they have already read. Another change was the insertion of the "Check Yourself" sections, which we added to help readers to check their comprehension of the material on the first time through it. A third change is that added extra exercises in the "For More Practice" section. Fourth, we added the answers to the "Check Yourself" sections and to the For More Practice exercises to help readers see for themselves if they understand the material by comparing their answers to the book. The final new book element was inspired by the "Green Card" of the IBM System/360. We believe that you will find that the MIPS Reference Data Card will be a handy reference when writing MIPS assembly language programs. Our idea is that you will remove the card from the front of the book, fold it in half, and keep it in your pocket, just as IBM S/360 programmers did in the 1960s.

Third, computers are so complex today that understanding the performance of a program involves understanding a good deal about the underlying principles and the organization of a given computer. Our goal is that readers of this book should be able to understand the performance of their progams and how to improve it. To aid in that goal, we added a new book element called "Understanding Program Performance" in several chapters. These sections often give concrete examples of how ideas in the chapter affect performance of real programs.

Fourth, in the interval since the second edition of this book, Moore's law has marched onward so that we now have processors with 200 million transistors, DRAM chips with a billion transistors, and clock rates of multiple gigahertz. The "Real Stuff" examples have been updated to describe such chips. This edition also includes AMD64/IA-32e, the 64-bit address version of the long-lived 80x86 architecture, which appears to be the nemesis of the more recent IA-64. It also reflects the transition from parallel buses to serial networks and switches. Later chapters describe Google, which was born after the second edition, in terms of its cluster technology and in novel uses of search.

Fifth, although many computer science and engineering students enjoy information technology for technology's sake, some have more altruistic interests. This latter group tends to have more women and underrepresented minorities. Consequently, we have added a new book element, "Computers in the Real World," two-page layouts found between each chapter. Our perspective is that information technology is more valuable for humanity than most other topics you could study—whether it is preserving our art heritage, helping the Third World, saving our environment, or even changing political systems—and so we demonstrate our view with concrete examples of nontraditional applications. We think readers of these segments will have a greater appreciation of the computing culture beyond the inherently interesting technology, much like those who read the history sections at the end of each chapter

Finally, books are like people: they usually get larger as they get older. By using technology, we have managed to do all the above and yet shrink the page count by hundreds of pages. As the table illustrates, the core portion of the book for hardware and software readers is on paper, but sections that some readers would value more than others are found on the companion CD. This technology also allows your authors to provide longer histories and more extensive exercises without concerns about lengthening the book. Once we added the CD to the book, we could then include a great deal of free software and tutorials that many instructors have told us they would like to use in their courses. This hybrid paper-CD publication weighs about 30% less than it did six years ago—an impressive goal for books as well as for people.

Instructor Support

We have collected a great deal of material to help instructors teach courses using this book. Solutions to exercises, figures from the book, lecture notes, lecture slides, and other materials are available to adopters from the publisher. Check the publisher's Web site for more information:

`www.mkp.com/companions/1558606041`

Concluding Remarks

If you read the following acknowledgments section, you will see that we went to great lengths to correct mistakes. Since a book goes through many printings, we have the opportunity to make even more corrections. If you uncover any remaining, resilient bugs, please contact the publisher by electronic mail at *cod3bugs@mkp.com* or by low-tech mail using the address found on the copyright page. The first person to report a technical error will be awarded a $1.00 bounty upon its implementation in future printings of the book!

This book is truly collaborative, despite one of us running a major university. Together we brainstormed about the ideas and method of presentation, then individually wrote about one-half of the chapters and acted as reviewer for every draft of the other half. The page count suggests we again wrote almost exactly the same number of pages. Thus, we equally share the blame for what you are about to read.

Acknowledgments for the Third Edition

We'd like to again express our appreciation to **Jim Larus** for his willingness in contributing his expertise on assembly language programming, as well as for welcoming readers of this book to use the simulator he developed and maintains. Our

exercise editor **Dan Sorin** took on the Herculean task of adding new exercises and answers. **Peter Ashenden** worked similarly hard to collect and organize the companion CD.

We are grateful to the many instructors who answered the publisher's surveys, reviewed our proposals, and attended focus groups to analyze and respond to our plans for this edition. They include the following individuals: Michael Anderson (University of Hartford), David Bader (University of New Mexico), Rusty Baldwin (Air Force Institute of Technology), John Barr (Ithaca College), Jack Briner (Charleston Southern University), Mats Brorsson (KTH, Sweden), Colin Brown (Franklin University), Lori Carter (Point Loma Nazarene University), John Casey (Northeastern University), Gene Chase (Messiah College), George Cheney (University of Massachusetts, Lowell), Daniel Citron (Jerusalem College of Technology, Israel), Albert Cohen (INRIA, France), Lloyd Dickman (PathScale), Jose Duato (Universidad Politécnica de Valencia, Spain), Ben Dugan (University of Washington), Derek Eager (University of Saskatchewan, Canada), Magnus Ekman (Chalmers University of Technology, Sweden), Ata Elahi (Southern Connecticut State University), Soundararajan Ezekiel (Indiana University of Pennsylvania), Ernest Ferguson (Northwest Missouri State University), Michael Fry (Lebanon Valley College, Pennsylvania), R. Gaede (University of Arkansas at Little Rock), Jean-Luc Gaudiot (University of California, Irvine), Thomas Gendreau (University of Wisconsin, La Crosse), George Georgiou (California State University, San Bernardino), Paul Gillard (Memorial University of Newfoundland, Canada), Joe Grimes (California Polytechnic State University, SLO), Max Hailperin (Gustavus Adolphus College), Jayantha Herath (St. Cloud State University, Minnesota), Mark Hill (University of Wisconsin, Madison), Michael Hsaio (Virginia Tech), Richard Hughey (University of California, Santa Cruz), Tony Jebara (Columbia University), Elizabeth Johnson (Xavier University), Peter Kogge (University of Notre Dame), Morris Lancaster (BAH), Doug Lawrence (University of Montana), David Lilja (University of Minnesota), Nam Ling (Santa Clara University, California), Paul Lum (Agilent Technologies), Stephen Mann (University of Waterloo, Canada), Diana Marculescu (Carnegie Mellon University), Margaret McMahon (U.S. Naval Academy Computer Science), Uwe Meyer-Baese (Florida State University), Chris Milner (University of Virginia), Tom Pittman (Southwest Baptist University), Jalel Rejeb (San Jose State University, California), Bill Siever (University of Missouri, Rolla), Kevin Skadron (University of Virginia), Pam Smallwood (Regis University, Colorado), K. Stuart Smith (Rocky Mountain College), William J. Taffe (Plymouth State University), Michael E. Thomodakis (Texas A&M University), Ruppa K. Thulasiram (University of Manitoba, Canada), Ye Tung (University of South Alabama), Steve VanderLeest (Calvin College), Neal R. Wagner (University of Texas at San Antonio), and Kent Wilken (University of California, Davis).

We are grateful too to those who carefully read our draft manuscripts; some read successive drafts to help ensure new errors didn't creep in as we revised. They include Krste Asanovic (Massachusetts Institute of Technology), Jean-Loup Baer (University of Washington), David Brooks (Harvard University), Doug Clark (Princeton University), Dan Connors (University of Colorado at Boulder), Matt Farrens (University of California, Davis), Manoj Franklin (University of Maryland College Park), John Greiner (Rice University), David Harris (Harvey Mudd College), Paul Hilfinger (University of California, Berkeley), Norm Jouppi (Hewlett-Packard), David Kaeli (Northeastern University), David Oppenheimer (University of California, Berkeley), Timothy Pinkston (University of Southern California), Mark Smotherman (Clemson University), and David Wood (University of Wisconsin, Madison).

To help us meet our goal of creating 70% new exercises and solutions for this edition, we recruited several graduate students recommended to us by their professors. We are grateful for their creativity and persistence: Michael Black (University of Maryland), Lei Chen (University of Rochester), Nirav Dave (Massachusetts Institute of Technology), Wael El Essawy (University of Rochester), Nikil Mehta (Brown University), Nicholas Nelson (University of Rochester), Aaron Smith (University of Texas, Austin), and Charlie Wang (Duke University).

We would like to especially thank **Mark Smotherman** for making a careful final pass to find technical and writing glitches that significantly improved the quality of this edition.

We wish to thank the extended Morgan Kaufmann family for agreeing to publish this book again under the able leadership of **Denise Penrose**. She developed the vision of the hybrid paper-CD book and recruited the many people above who played important roles in developing the book.

Simon Crump managed the book production process, and **Summer Block** coordinated the surveying of users and their responses. We thank also the many freelance vendors who contributed to this volume, especially **Nancy Logan** and **Dartmouth Publishing, Inc.**, our compositors.

The contributions of the nearly 100 people we mentioned here have made this third edition our best book yet. Enjoy!

David A. Patterson **John L. Hennessy**

Computers in the Real World

Saving Lives through Better Diagnosis

Problem: Find a way to examine internal organs to diagnose psychological problems without the use of invasive surgery or harmful radiation.

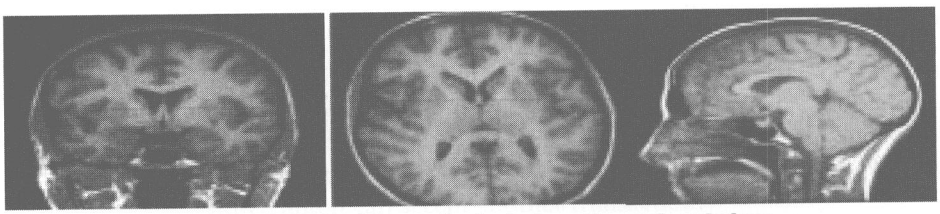
Solution: The development of magnetic resonance imaging (MRI), a three-dimensional scanning technology, has been one of the most important breakthroughs in modern medical technology. MRI uses a combination of radio-frequency pulses and magnetic fields to scan tissue. The organ to be imaged is scanned in a series of two-dimensional slices, which are then composed to create a three-dimensional image.

In addition to this computationally intensive task of composing the slices to create a volumetric image, extensive computation is used to extract the initial two-dimensional images, since the signal-to-noise ratio is often low. The development of MRI has allowed the scanning of soft tissues, such as the brain, for which X-rays are not as effective and exploratory surgery is dangerous. Without a cost-effective computing capability, MRI would remain slow and expensive.

The two illustrations shows a series of MRI images of the human brain; the images below represent two-dimensional slices, while those on the facing page show a three-dimensional reconstruction. Once an image is in digital form, a physician can manipulate the image, removing outer layers, examining the image from different viewpoints, or looking at the three-dimensional structure to help in diagnosis.

The major benefits of MRI are twofold:

- It can reduce the need for unnecessary exploratory surgery. A physician may be able to determine whether a patient ex-



MRI images of a human brain, in two-dimensional view

periencing headaches has a brain tumor, which requires surgery, or simply needs medication for a headache.

- By providing a surgeon with an accurate three-dimensional image, MRI can improve the surgical planning process and hence the outcome. For example, in operating on the brain to remove a tumor without accurate images of the tumor, the surgeon likely would have to enter the brain and then create a plan on the fly depending on the size and exact placement of the tumor. Furthermore, minimally invasive techniques (e.g. endoscopic surgery), which have become quite effective, would be impossible without accurate images.

There are many new interesting uses of MRI technology, which rely on faster and more cost effective computing. Some of the most promising are

- real-time imaging of the heart and blood vessels to enhance diagnosis of cardiac and cardiovascular disease;
- Combining real-time images and MRI images during surgery to help surgeons accurately perform surgery, particularly when using minimally invasive techniques.
- Functional MRI (FMRI): a new type of application that uses MRI to examine brain function, primarily by analyzing blood flow in various portions of the brain. FMRI is being used for a number of applications, including exploring the physiological bases for cognitive problems such as dyslexia, pain management, planning for neurosurgery, and understanding neurological disorders.

To learn more see these references on the library

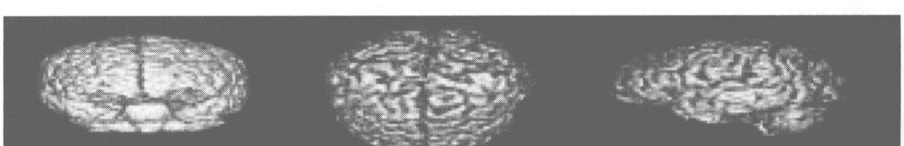
MRI scans from the National Institutes of Health's Visible Human project

Principles of MRI and its application to medical imaging (long and reasonably detailed, but only a little mathematics)

Using MRI to do real-time cardiac imaging and angiography (imaging of blood vessels)

Functional MRI, www.fmri.org/fmri.htm

Visualization and imaging (including MRI and CT images): high-performance computing for complex images



MRI images of a human brain in three dimensions

Index

CD information is listed by chapter and section number followed by page ranges (CD9.1:1-2). In More Depth references are listed by chapter number followed by page ranges (IMD4:5-6). Page references preceded by a single letter refer to appendices.

A