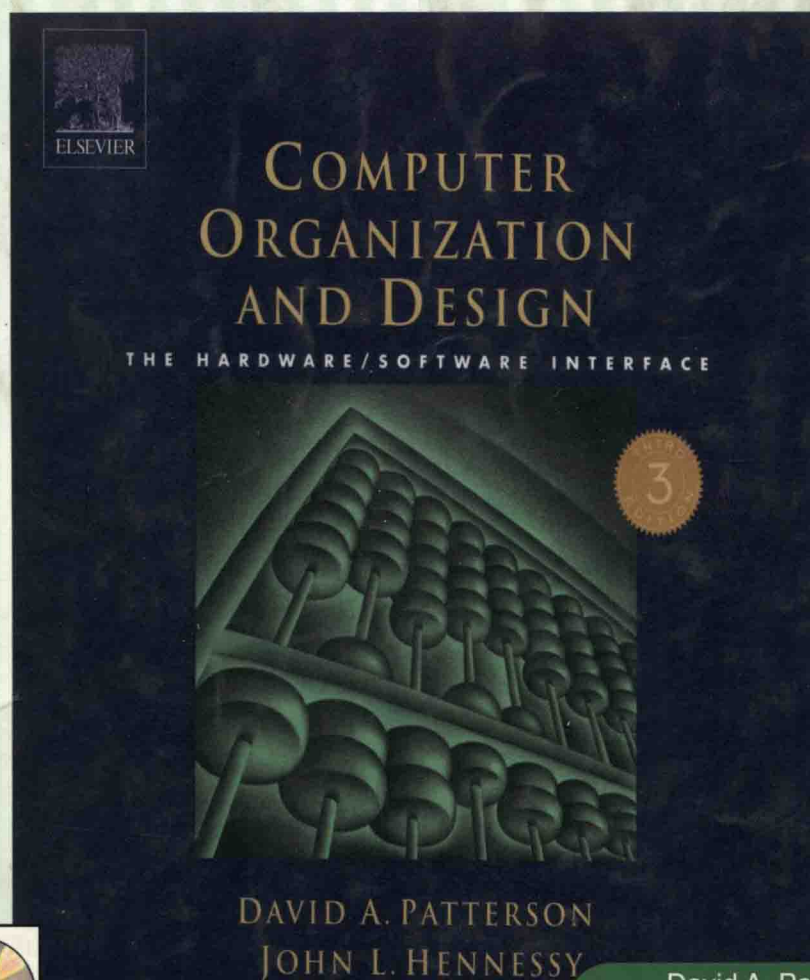


# 计算机组成与设计

## 硬件/软件接口

(英文版·第3版)



机械工业出版社

David A. Patterson  
(美) 加州大学伯克利分校  
John L. Hennessy  
斯坦福大学 著



机械工业出版社  
China Machine Press

# 计算机组成与设计 硬件/软件接口

(英文版·第3版)

Computer Organization and Design The Hardware/Software Interface

(Third Edition)

软件设计者对软件系统运行环境硬件技术是否了解、了解多少会很大程度地影响软件系统的性能,同样,硬件设计者也必须了解他们的设计决策将对软件产生怎样的影响。本书着眼于当前计算机设计中最基本的概念,展示了软硬件间的关系。无论上述的哪一类读者,本书的内容都会使他们对计算机有更深入的认识。

同以往版本一样,本书采用MIPS处理器作为展示计算机硬件技术基本功能的核心。书中逐条指令地列举了完整的MIPS指令集——汇编语言的核心、计算机算术运算、流水线、存储器层次结构以及I/O,并介绍了网络和多处理器结构的基本内容。

将CPU性能和程序性能紧密地联系起来是本版的一个新增内容。作者展示了软硬件部件(如算法、编程语言、编译器、指令集系统结构以及处理器的实现)如何影响程序的性能。另外,本版对软硬件的讨论更加深入,并在光盘中对侧重硬件和侧重软件的读者分别提供了相关资料。

随书光盘的内容非常丰富,不仅包括第9章、附录、本书网站内容、附加习题、术语表、参考文献、索引等,而且还提供了HDL模拟器、MIPS模拟器以及FPGA设计工具等软件。

## 本书主要特点

- 书中资料全部更新,以反映最新技术。
- 使用标准32位MIPS指令集作为教学指令集。
- 反映了体系结构的最新进展:
  - Intel IA-32
  - Power PC 604
  - Pentium P4
  - Google的PC集群
  - 处理器基准测试集SPEC CPU2000
  - Web基准测试集SPEC Web99
  - 嵌入式系统测试集EEMBC
  - AMD Opteron存储器层次结构
  - AMD与IA-64比较
  - Intrinsity FastMATH服务器处理器
- 硬件方面的新资料:
  - 使用逻辑设计约定
  - 用硬件描述语言设计
  - 高级流水线设计
  - 使用FPGA设计
  - HDL模拟器和使用说明
  - Xilinx CAD工具
- 软件方面的新资料:
  - 编译器如何工作
  - 如何优化编译器
  - 如何实现面向对象程序设计语言
  - 程序设计语言、编译器、操作系统以及数据库的历史

## 作者简介

**David A. Patterson** 加州大学伯克利分校计算机科学系教授,美国国家工程院院士,IEEE和ACM会士,曾因成功的启发式教育方法被IEEE授予James H. Mulligan, Jr.教育奖章。他因为对RISC技术的贡献而荣获1995年IEEE技术成就奖,而在RAID技术方面的成就为他赢得了1999年IEEE Reynold Johnson信息存储奖。2000年他和John L. Hennessy分享了John von Neumann奖。

**John L. Hennessy** 斯坦福大学校长,IEEE和ACM会士,美国国家工程院院士及美国科学艺术研究院院士。Hennessy教授因为在RISC技术方面做出了突出贡献而荣获2001年的Eckert-Mauchly奖章,他也是2001年Seymour Cray计算机工程奖得主,并且和本书另外一位作者David A. Patterson分享了2000年John von Neumann奖。



上架指导: 计算机/计算机原理

ISBN 7-111-19339-3



9 787111 193395



华章图书



华章网站 <http://www.hzbook.com>

网上购书: [www.china-pub.com](http://www.china-pub.com)

投稿热线: (010) 88379604

购书热线: (010) 68995259, 68995264

读者信箱: [hzjsj@hzbook.com](mailto:hzjsj@hzbook.com)

限中国大陆地区销售

ISBN 7-111-19339-3

定价: 85.00元 (附光盘)

HZ BOOKS



# 计算机组成与设计

硬件

软件接口

David A. Patterson  
John L. Hennessy

著

Computer Organization and Software Interface (Third Edition)

英文版  
第3版



清华大学出版社  
Tsinghua University Press



经典原版书库

# 计算机组成与设计

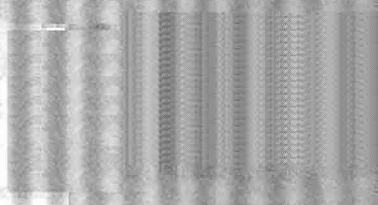
## 硬件/软件接口

(英文版·第3版)

Computer Organization and Design

The Hardware/Software Interface

(Third Edition)



(美) David A. Patterson  
加州大学伯克利分校  
John L. Hennessy  
斯坦福大学

著



机械工业出版社  
China Machine Press

David A. Patterson and John L. Hennessy: Computer Organization and Design: The Hardware/Software Interface, Third Edition (ISBN: 1-55860-604-1 ISBN-13: 978-1-55860-604-3).

Original English language edition copyright © 2005 by Elsevier Inc. All rights reserved.

Authorized English language reprint edition published by the Proprietor.

ISBN: 981-259-715-8 ISBN-13: 978-981-259-715-1

Copyright © 2006 by Elsevier (Singapore) Pte Ltd.

Printed in China by China Machine Press under special arrangement with Elsevier (Singapore) Pte Ltd. This edition is authorized for sale in China only, excluding Hong Kong SAR and Taiwan.

Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由Elsevier (Singapore) Pte Ltd.授权机械工业出版社在中国大陆境内独家发行。本版仅限在中国境内(不包括香港特别行政区及台湾地区)出版及标价销售。未经许可之出口,视为违反著作权法,将受法律之制裁。

版权所有,侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号: 图字: 01-2006-3122

### 图书在版编目(CIP)数据

计算机组成与设计: 硬件/软件接口(英文版·第3版)/(美)帕特森(Patterson, D. A.)等著. —北京: 机械工业出版社, 2006.7

(经典原版书库)

书名原文: Computer Organization and Design: The Hardware/Software Interface, Third Edition  
ISBN 7-111-19339-3

I. 计… II. 帕… III. ① 计算机体系结构—英文 ② 微型计算机—接口设备—英文  
IV. ① TP303 ② TP364

中国版本图书馆CIP数据核字(2006)第062551号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑: 迟振春

北京中兴印刷有限公司印刷 · 新华书店北京发行所发行

2006年7月第1版第1次印刷

186mm × 240mm · 41.5印张

定价: 85.00元(附光盘)

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换  
本社购书热线 (010) 68326294

# MIPS Reference Data

## 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 <sub>hex</sub>
Add Immediate	addi	I	$R[rt] = R[rs] + \text{SignExtImm}$	(1)(2) 8 <sub>hex</sub>
Add Imm. Unsigned	addiu	I	$R[rt] = R[rs] + \text{SignExtImm}$	(2) 9 <sub>hex</sub>
Add Unsigned	addu	R	$R[rd] = R[rs] + R[rt]$	0/21 <sub>hex</sub>
And	and	R	$R[rd] = R[rs] \& R[rt]$	0/24 <sub>hex</sub>
And Immediate	andi	I	$R[rt] = R[rs] \& \text{ZeroExtImm}$	(3) 0 <sub>hex</sub>
Branch On Equal	beq	I	if $R[rs] == R[rt]$ $PC = PC + 4 + \text{BranchAddr}$	(4) 4 <sub>hex</sub>
Branch On Not Equal	bne	I	if $R[rs] != R[rt]$ $PC = PC + 4 + \text{BranchAddr}$	(4) 5 <sub>hex</sub>
Jump	j	J	$PC = \text{JumpAddr}$	(5) 2 <sub>hex</sub>
Jump And Link	jal	J	$R[31] = PC + 4; PC = \text{JumpAddr}$	(5) 3 <sub>hex</sub>
Jump Register	jr	R	$PC = R[rs]$	0/08 <sub>hex</sub>
Load Byte Unsigned	lbu	I	$R[rt] = (24'b0, M[R[rs]] + \text{SignExtImm})(7:0)$	(2) 0/24 <sub>hex</sub>
Load Halfword Unsigned	lhu	I	$R[rt] = (16'b0, M[R[rs]] + \text{SignExtImm})(15:0)$	(2) 0/25 <sub>hex</sub>
Load Upper Imm.	lui	I	$R[rt] = (\text{imm}, 16'b0)$	0 <sub>hex</sub>
Load Word	lw	I	$R[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 0/23 <sub>hex</sub>
Nor	nor	R	$R[rd] = \sim (R[rs]   R[rt])$	0/27 <sub>hex</sub>
Or	or	R	$R[rd] = R[rs]   R[rt]$	0/25 <sub>hex</sub>
Or Immediate	ori	I	$R[rt] = R[rs]   \text{ZeroExtImm}$	(3) 0 <sub>hex</sub>
Set Less Than	slt	R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	0/28 <sub>hex</sub>
Set Less Than Imm.	slti	I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	(2) a <sub>hex</sub>
Set Less Than Imm. Unsigned	sltiu	I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	(2)(6) b <sub>hex</sub>
Set Less Than Unsigned	sltu	R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	(6) 0/2b <sub>hex</sub>
Shift Left Logical	sll	R	$R[rd] = R[rs] \ll \text{shamt}$	0/00 <sub>hex</sub>
Shift Right Logical	srl	R	$R[rd] = R[rs] \gg \text{shamt}$	0/02 <sub>hex</sub>
Store Byte	sb	I	$M[R[rs] + \text{SignExtImm}](7:0) = R[rt](7:0)$	(2) 28 <sub>hex</sub>
Store Halfword	sh	I	$M[R[rs] + \text{SignExtImm}](15:0) = R[rt](15:0)$	(2) 29 <sub>hex</sub>
Store Word	sw	I	$M[R[rs] + \text{SignExtImm}] = R[rt]$	(2) 2b <sub>hex</sub>
Subtract	sub	R	$R[rd] = R[rs] - R[rt]$	(1) 0/22 <sub>hex</sub>
Subtract Unsigned	subu	R	$R[rd] = R[rs] - R[rt]$	0/23 <sub>hex</sub>

- (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
I	opcode	rs	rt	immediate		
	31	26 25	21 20	16 15		
J	opcode	address				
	31	26 25				

## ARITHMETIC CORE INSTRUCTION SET

NAME	MNE- MON- FOR- IC	MAT	OPERATION	OPCODE/ FUNCT (Hex)
Branch On FP True	bclt	FI	if(FPcond) $PC = PC + 4 + \text{BranchAddr}$	(4) 11/8/1/-
Branch On FP False	bclfi	FI	if(!FPcond) $PC = PC + 4 + \text{BranchAddr}$	(4) 11/8/0/-
Divide	div	R	$Lo = R[rs]/R[rt]; Hi = R[rs]\%R[rt]$	(6) 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	$FPcond = (F[fs] \text{ op } F[ft]) ? 1 : 0$	11/10/-/y
FP Compare Double	c.x.d*	FR	$FPcond = ((F[fs], F[fs+1]) \text{ op } (F[ft], F[ft+1])) ? 1 : 0$	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	$F[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 31/-/-/-
Load FP Double	ldc1	I	$F[rt], F[rt+1] = M[R[rs] + \text{SignExtImm}];$	(2) 35/-/-/-
Move From Hi	mghi	R	$R[rd] = Hi$	0/-/-/10
Move From Lo	mflr	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	$M[R[rs] + \text{SignExtImm}] = F[rt]$	(2) 39/-/-/-
Store FP Double	sdc1	I	$M[R[rs] + \text{SignExtImm}] = F[rt];$ $M[R[rs] + \text{SignExtImm} + 4] = F[rt+1]$	(2) 3d/-/-/-

## FLOATING POINT INSTRUCTION FORMATS

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

## PSEUDO INSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	if $R[rs] < R[rt]$ $PC = \text{Label}$
Branch Greater Than	bgt	if $R[rs] > R[rt]$ $PC = \text{Label}$
Branch Less Than or Equal	b1e	if $R[rs] <= R[rt]$ $PC = \text{Label}$
Branch Greater Than or Equal	bge	if $R[rs] >= R[rt]$ $PC = \text{Label}$
Load Immediate	li	$R[rd] = \text{immediate}$
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

P90 Language

# OPCODES, BASE CONVERSION, ASCII SYMBOLS

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)	all	add	00 0000	0	0	NUL	64	40	@
		sub	00 0001	1	1	SOH	65	41	A
j	srl	mul	00 0010	2	2	STX	66	42	B
jal	sra	div	00 0011	3	3	ETX	67	43	C
beq	sliv	sqr	00 0100	4	4	EOT	68	44	D
bne		abs	00 0101	5	5	ENQ	69	45	E
blez	srlv	mov	00 0110	6	6	ACK	70	46	F
bgz	sra	neg	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	00 1100	12	c	FF	76	4c	L
ori	break	trunc.w	00 1101	13	d	CR	77	4d	M
xori		ceil.w	00 1110	14	e	SO	78	4e	N
lui	sync	floor.w	00 1111	15	f	SI	79	4f	O
(2)			01 0000	16	10	DLE	80	50	P
			01 0001	17	11	DC1	81	51	Q
			01 0010	18	12	DC2	82	52	R
			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
			01 1000	24	18	CAN	88	58	X
			01 1001	25	19	EM	89	59	Y
			01 1010	26	1a	SUB	90	5a	Z
			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
lwr	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
ab			10 1000	40	28	(	104	68	h
ah			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
			10 1110	46	2e	,	110	6e	n
swr			10 1111	47	2f	/	111	6f	o
cache									
l1	tge	c.f.f	11 0000	48	30	0	112	70	p
lwc1	tgeu	c.unf	11 0001	49	31	1	113	71	q
lwc2	tlr	c.eq.f	11 0010	50	32	2	114	72	r
pref	tlru	c.usq.f	11 0011	51	33	3	115	73	s
	teq	c.o.f	11 0100	52	34	4	116	74	t
ldc1		c.o.f	11 0101	53	35	5	117	75	u
ldc2	tne	c.o.f	11 0110	54	36	6	118	76	v
		c.o.f	11 0111	55	37	7	119	77	w
ac		c.s.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.ngf.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

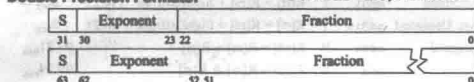
(2) opcode(31:26) == 17<sub>hex</sub> (11<sub>hex</sub>); if fnt(25:21) == 16<sub>hex</sub> (10<sub>hex</sub>) f = s (single); if fnt(25:21) == 17<sub>hex</sub> (11<sub>hex</sub>) f = d (double)

## IEEE 754 FLOATING POINT STANDARD

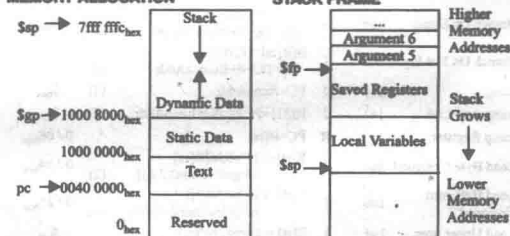
$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

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

IEEE Single Precision and Double Precision Formats:



### MEMORY ALLOCATION

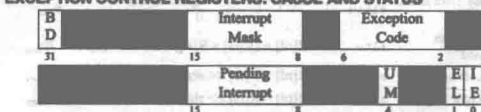


### DATA ALIGNMENT

Double Word							
Word				Word			
Half Word		Half Word		Half Word		Half Word	
Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte
0	1	2	3	4	5	6	7

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

Num ber	Name	Cause of Exception	Num ber	Name	Cause of Exception
0	Int	Interrupt (hardware)	9	Bp	Breakpoint Exception
4	AdE	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<sup>3</sup> for Disk, Communication; 2<sup>3</sup> for Memory)

SIZE	PRE-FIX	SIZE	PRE-FIX	SIZE	PRE-FIX	SIZE	PRE-FIX
10 <sup>3</sup> , 2 <sup>10</sup>	Kilo-	10 <sup>15</sup> , 2 <sup>50</sup>	Peta-	10 <sup>-3</sup>	milli-	10 <sup>-15</sup>	femto-
10 <sup>6</sup> , 2 <sup>20</sup>	Mega-	10 <sup>18</sup> , 2 <sup>60</sup>	Exa-	10 <sup>-6</sup>	micro-	10 <sup>-18</sup>	atto-
10 <sup>9</sup> , 2 <sup>30</sup>	Giga-	10 <sup>21</sup> , 2 <sup>70</sup>	Zetta-	10 <sup>-9</sup>	nano-	10 <sup>-21</sup>	zepto-
10 <sup>12</sup> , 2 <sup>40</sup>	Tera-	10 <sup>24</sup> , 2 <sup>80</sup>	Yotta-	10 <sup>-12</sup>	pico-	10 <sup>-24</sup>	yocto-

The symbol for each prefix is just its first letter, except  $\mu$  is used for micro.

## 出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域中取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅肇划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战，而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall, Addison-Wesley, McGraw-Hill, Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum, Stroustrup, Kernighan, Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及收藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国



家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T., Stanford, U.C. Berkeley, C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

电子邮件：hzjsj@hzbook.com

联系电话：(010) 68995264

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

## 专家指导委员会

(按姓氏笔画顺序)

尤晋元  
石教英  
张立昂  
邵维忠  
周克定  
郑国梁  
高传善  
裘宗燕

王 珊  
吕 建  
李伟琴  
陆丽娜  
周傲英  
施伯乐  
梅 宏  
戴 葵

冯博琴  
孙玉芳  
李师贤  
陆鑫达  
孟小峰  
钟玉琢  
程 旭

史忠植  
吴世忠  
李建中  
陈向群  
岳丽华  
唐世渭  
程时端

史美林  
吴时霖  
杨冬青  
周伯生  
范 明  
袁崇义  
谢希仁



# 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 funda-



mentals 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		
	■ 1.7 (History)		
	2.1 to 2.11		
2. Instructions: Language of the Computer	■ 2.12 (Compilers)		
	2.13 (C sort)		
	■ 2.14 (Java)		
	2.15 to 2.18		
	■ 2.19 (History)		
3. Arithmetic for Computers	3.1 to 3.9		
	■ 3.10 (History)		
D. RISC instruction set architectures	■ D.1 to D.19		
4. Assessing and Understanding Performance	4.1 to 4.6		
	■ 4.7 (History)		
B. The Basics of Logic Design	■ B.1 to B.13		
5. The Processor: Datapath and Control	5.1 (Overview)		
	5.2 to 5.7		
	■ 5.8 (Microcode)		
	■ 5.9 (Verilog)		
	5.10 to 5.12		
C. Mapping Control to Hardware	■ 5.13 (History)		
	■ C.1 to C.6		
	6.1 (Overview)		
6. Enhancing Performance with Pipelining	6.2 to 6.6		
	■ 6.7 (verilog)		
	6.8 to 6.9		
	6.10 to 6.12		
	■ 6.13 (History)		
7. Large and Fast: Exploiting Memory Hierarchy	7.1 to 7.8		
	■ 7.9 (History)		
8. Storage, Networks, and Other Peripherals	8.1 to 8.2		
	■ 8.3 (Networks)		
	8.4 to 8.10		
	■ 8.13 (History)		
9. Multiprocessors and Clusters	■ 9.1 to 9.10		
	■ 9.11 (History)		
A. Assemblers, Linkers, and the SPIM Simulator	■ A.1 to A.12		
Computers in the Real World	Between Chapters		

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 programs 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](http://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](mailto: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).