HZ BOOKS
华章教育

计 算 机 科 学 丛 书

ELSEVIER

RISC-V版

计算机组成与设计

硬件/软件接口（原书第5版）

[美] 戴维·A. 帕特森（David A. Patterson）
约翰·L. 亨尼斯（John L. Hennessy） 著
易江芳 刘先华 等译

计算机体系结构新黄金时代必读之作，理解专用软硬件协同设计，共建开源生态

Computer Organization and Design
The Hardware/Software Interface, RISC-V Edition

机械工业出版社
China Machine Press

计算机组成与设计 硬件/软件接口 原书第5版·RISC-V版

Computer Organization and Design The Hardware/Software Interface, RISC-V Edition

在广大计算机程序员和工程师中，几乎没有人不知道Patterson和Hennessy的大作，而今RISC-V版的推出，再次点燃了大家的热情。RISC-V作为一种开源体系结构，从最初用于支持科研和教学，到现在已发展为产业标准的指令集。正在和即将阅读本书的年轻人，你们不仅能够从先行者的智慧中理解RISC-V的精髓，而且有望创建自己的RISC-V内核，为广阔的开源硬件和软件生态系统贡献力量。

—— Krste Asanovi，RISC-V基金会主席

教材的选择往往是一个令人沮丧的妥协过程——教学方法的适用度、知识点的覆盖范围、文辞的流畅性、内容的严谨度、成本的高低等都需要考虑。本书之所以是难得一见的好书，正是因为它能满足各个方面的要求，不再需要任何妥协。这不仅是一部关于计算机组成的教科书，也是所有计算机科学教科书的典范。

—— Michael Goldweber，Xavier University

无论是对于80后、90后还是00后，这都是一本应该珍藏在书架上（或iPad中）的计算机体系结构教材。这本书既古老又新颖，不仅介绍了那些伟大的原理——摩尔定律、抽象、加速经常性事件、冗余、存储层次、并行和流水线，而且使用现代设计对这些伟大原理进行了说明。

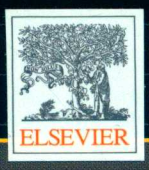
—— Mark D. Hill，University of Wisconsin-Madison

本书不仅讲解计算机体系结构，而且为读者准备了迎接新的变化与挑战的“锦囊”。目前，半导体工艺技术按比例缩小的困难使得所有系统功率受限，而移动系统和大数据处理的性能需求却仍在不断增长。在计算技术的新时代，必须进行软硬件协同设计，并且系统级体系结构优化与部件级优化一样重要。

—— Christos Kozyrakis，Stanford University

Patterson和Hennessy讨论了不断变化的计算机硬件体系结构中的重要议题，强调硬件和软件模块在不同抽象层次上的交互。书中涵盖各种硬件和软件机制，I/O和并行的概念贯穿其中，全景式呈现了后PC时代的计算机体系结构。无论是平板电脑硬件工程师还是云计算软件架构师，如果你正对能源效率和并行化问题一筹莫展，那么本书必将成为不二之选。

—— Jae C. Oh，Syracuse University



本书译自原版*Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition*
并由Elsevier授权出版

华章教育服务微信号

上架指导：计算机体系结构
ISBN 978-7-111-65214-4
9 787111 652144 >
定价：169.00元

投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com
客服电话：(010) 88361066 88379833 68326294

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

计 算 机 科 学 丛 书

RISC-V版

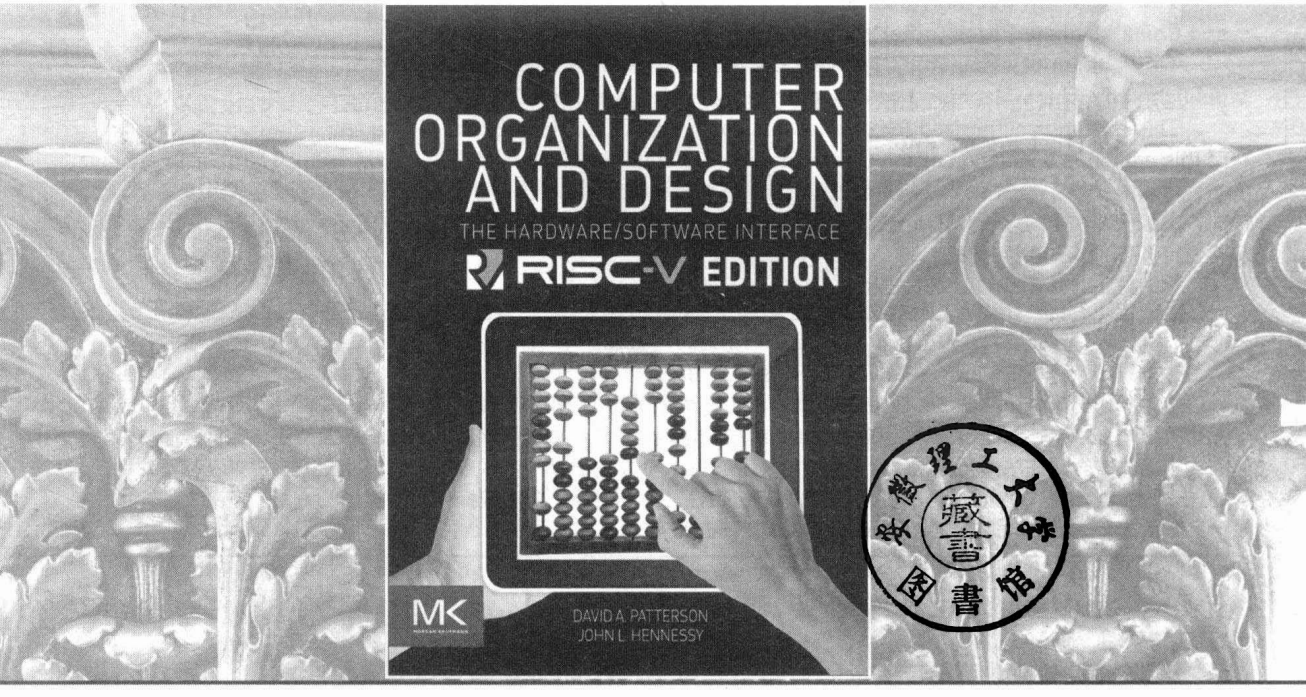
计算机组成与设计

硬件/软件接口（原书第5版）

[美] 戴维·A. 帕特森（David A. Patterson）
约翰·L. 亨尼斯（John L. Hennessy） 著
易江芳 刘先华 等译

Computer Organization and Design

The Hardware/Software Interface, RISC-V Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

计算机组成与设计：硬件 / 软件接口（原书第 5 版 · RISC-V 版）/（美）戴维 · A. 帕特森（David A. Patterson），（美）约翰 · L. 亨尼斯（John L. Hennessy）著；易江芳等译 . —北京：机械工业出版社，2020.4
（计算机科学丛书）
书名原文：Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition

ISBN 978-7-111-65214-4

I. 计… II. ① 戴… ② 约… ③ 易… III. ① 计算机体系结构 ②微型计算机 – 接口技术 IV. ① TP303 ② TP364.7

中国版本图书馆 CIP 数据核字（2020）第 052937 号

本书版权登记号：图字 01-2018-3144

出版发行：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：曲 熠 责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司 版 次：2020 年 5 月第 1 版第 1 次印刷
开 本：185mm × 260mm 1/16 印 张：30.5
书 号：ISBN 978-7-111-65214-4 定 价：169.00 元

客服电话：（010）88361066 88379833 68326294 投稿热线：（010）88379604
华章网站：www.hzbook.com 读者信箱：hzjsj@hzbook.com

作者简介

戴维·A. 帕特森（David A. Patterson）

Patterson与Hennessy共同荣获了2017年度“图灵奖”，以表彰他们在计算机体系结构领域的开创性贡献。Patterson现为Google杰出工程师，之前为加州大学伯克利分校教授。他曾任ACM主席一职，目前是ACM和IEEE会士，美国艺术与科学院和计算机历史博物馆院士，并入选了美国国家工程院、国家科学院和硅谷工程名人堂。他领导了RISC I的设计与实现工作，并且是RAID项目的领导者。

约翰·L. 亨尼斯（John L. Hennessy）

Hennessy与Patterson共同荣获了2017年度“图灵奖”。Hennessy现为Google母公司Alphabet的董事长，之前曾任斯坦福大学第十任校长。他是IEEE和ACM会士，美国国家工程院、国家科学院、美国哲学院以及美国艺术与科学院院士。他于1981年开始研究MIPS项目，之后创办MIPS Computer Systems公司，开发了最早的商用RISC微处理器之一。他还领导了DASH项目，设计了第一个可扩展cache一致性多处理器原型。

译者简介

易江芳

北京大学信息科学技术学院计算机系副教授，长期从事微处理器体系结构设计和性能优化科研及教学工作，近年来主持和参加了国家科技重大专项、北京市科委等十余项科研项目，参与了北京大学“众志”系列自主CPU系统芯片的设计、验证和流片工作，发表论文10余篇。曾获北大方正奖教金，主持了北京大学计算机体系结构实习课程的教改项目。

刘先华

北京大学信息科学技术学院计算机系副教授，长期从事计算机系统结构和编译优化科研及教学工作，近年来主持和参加了国家自然科学基金、国家科技重大专项等十余项科研项目，参与研发了北京大学“众志”系列自主CPU及配套软硬件系统并开展了相关产业化工作，发表论文20余篇。曾获首届“全国高校计算机专业优秀教师”奖励、宝钢奖教金、北京大学教学优秀奖等荣誉。

RISC-V版特色

- 紧跟后PC时代的发展浪潮，关注并行技术，新内容涵盖平板电脑、云基础设施、ARM（移动计算设备）以及x86（云计算）体系结构，新实例包括Intel Core i7、ARM Cortex-A53以及NVIDIA Fermi GPU。
- 全面切换至RISC-V（64位），精选并讲解其核心指令，同时结合“硬件/软件接口”和“详细阐述”等模块，为不同基础和侧重的读者铺设了深入研究的路径。
- 新增矩阵乘法实例，随着章节推进不断“加速”程序，分别采用子字并行、指令级并行、cache分块技术和线程级并行，仅增加24行代码便使性能翻了200倍，直观呈现出硬件对提高能效的重要性。
- 强调计算机体系结构中的8个伟大思想：并行，流水线，预测，摩尔定律，存储层次，抽象，加速经常性事件，可靠性。对这些思想的应用贯穿全书，引用总数约100次，反映了技术精髓所在。
- 本书网站booksite.elsevier.com/9780128122754提供更多学习资源，可下载RISC-V软件工具，还有用于教学的PPT、练习题答案和附录等。

封面设计　锡　彬

RISC-V Reference Data

RV64I BASE INTEGER INSTRUCTIONS, in alphabetical order

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)	NOTE
add,addw	R	ADD (Word)	R[rd] = R[rs1] + R[rs2]	1)
addi,addiw	I	ADD Immediate (Word)	R[rd] = R[rs1] + imm	1)
and	R	AND	R[rd] = R[rs1] & R[rs2]	
andi	I	AND Immediate	R[rd] = R[rs1] & imm	
auipc	U	Add Upper Immediate to PC	R[rd] = PC + {imm, 12'b0}	
beq	SB	Branch EQual	if(R[rs1]==R[rs2) PC=PC+{imm,1b'0}	
bge	SB	Branch Greater than or Equal	if(R[rs1]>=R[rs2) PC=PC+{imm,1b'0}	
bgeu	SB	Branch ≥ Unsigned	if(R[rs1]>=R[rs2) PC=PC+{imm,1b'0}	2)
blt	SB	Branch Less Than	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	
bltu	SB	Branch Less Than Unsigned	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	2)
bne	SB	Branch Not Equal	if(R[rs1]!=R[rs2) PC=PC+{imm,1b'0}	
csrrc	I	Cont./Stat.RegRead&Clear	R[rd] = CSR;CSR = CSR & ~R[rs1]	
csrrci	I	Cont./Stat.RegRead&Clear Imm	R[rd] = CSR;CSR = CSR & ~imm	
csrrs	I	Cont./Stat.RegRead&Set	R[rd] = CSR; CSR = CSR | R[rs1]	
csrrsi	I	Cont./Stat.RegRead&Set Imm	R[rd] = CSR; CSR = CSR | imm	
csrrw	I	Cont./Stat.RegRead&Write	R[rd] = CSR; CSR = R[rs1]	
csrrwi	I	Cont./Stat.Reg Read&Write Imm	R[rd] = CSR; CSR = imm	
ebreak	I	Environment BREAK	Transfer control to debugger	
ecall	I	Environment CALL	Transfer control to operating system	
fence	I	Synch thread	Synchronizes threads	
fence.i	I	Synch Instr & Data	Synchronizes writes to instruction stream	
jal	UJ	Jump & Link	R[rd] = PC+4; PC = PC + {imm,1b'0}	
jalr	I	Jump & Link Register	R[rd] = PC+4; PC = R[rs1]+imm	3)
lb	I	Load Byte	R[rd] = {56'bM[](7),M[R[rs1]+imm](7:0)}	4)
lbu	I	Load Byte Unsigned	R[rd] = {56'b0,M[R[rs1]+imm](7:0)}	
ld	I	Load Doubleword	R[rd] = M[R[rs1]+imm](63:0)	
lh	I	Load Halfword	R[rd] = {48'bM[](15),M[R[rs1]+imm](15:0)}	4)
lhu	I	Load Halfword Unsigned	R[rd] = {48'b0,M[R[rs1]+imm](15:0)}	
lui	U	Load Upper Immediate	R[rd] = {32b'imm<31>, imm, 12'b0}	
lw	I	Load Word	R[rd] = {32'bM[](31),M[R[rs1]+imm](31:0)}	4)
lwu	I	Load Word Unsigned	R[rd] = {32'b0,M[R[rs1]+imm](31:0)}	
or	R	OR	R[rd] = R[rs1] | R[rs2]	
ori	I	OR Immediate	R[rd] = R[rs1] | imm	
sb	S	Store Byte	M[R[rs1]+imm](7:0) = R[rs2](7:0)	
sd	S	Store Doubleword	M[R[rs1]+imm](63:0) = R[rs2](63:0)	
sh	S	Store Halfword	M[R[rs1]+imm](15:0) = R[rs2](15:0)	
sll,sllw	R	Shift Left (Word)	R[rd] = R[rs1] << R[rs2]	1)
slli,slliw	I	Shift Left Immediate (Word)	R[rd] = R[rs1] << imm	1)
slt	R	Set Less Than	R[rd] = (R[rs1] < R[rs2]) ? 1 : 0	
slti	I	Set Less Than Immediate	R[rd] = (R[rs1] < imm) ? 1 : 0	
sltiu	I	Set < Immediate Unsigned	R[rd] = (R[rs1] < imm) ? 1 : 0	2)
sltu	R	Set Less Than Unsigned	R[rd] = (R[rs1] < R[rs2]) ? 1 : 0	2)
sra,sraw	R	Shift Right Arithmetic (Word)	R[rd] = R[rs1] >> R[rs2]	1,5)
srai,sraiw	I	Shift Right Arith Imm (Word)	R[rd] = R[rs1] >> imm	1,5)
srl,srlw	R	Shift Right (Word)	R[rd] = R[rs1] >> R[rs2]	1)
srli,srliw	I	Shift Right Immediate (Word)	R[rd] = R[rs1] >> imm	1)
sub,subw	R	SUBtract (Word)	R[rd] = R[rs1] – R[rs2]	1)
sw	S	Store Word	M[R[rs1]+imm](31:0) = R[rs2](31:0)	
xor	R	XOR	R[rd] = R[rs1] ^ R[rs2]	
xori	I	XOR Immediate	R[rd] = R[rs1] ^ imm	

Notes: 1) The Word version only operates on the rightmost 32 bits of a 64-bit regsiters
2) Operation assumes unsigned integers (instead of 2's complement)
3) The least significant bit of the branch address in jalr is set to 0
4) (signed) Load instructions extend the sign bit of data to fill the 64-bit register
5) Replicates the sign bit to fill in the leftmost bits of the result during right shift
6) Multiply with one operand signed and one unsigned
7) The Single version does a single-precision operation using the rightmost 32 bits of a 64-bit F regsiter
8) Classify writes a 10-bit mask to show which properties are true (e.g., –inf, -0, +0, +inf, denorm, ...)
The immediate field is sign-extended in RISC-V

ARITHMETIC CORE INSTRUCTION SET

RV64M Multiply Extension

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)	NOTE
mul,mulw	R	MULtiply (Word)	R[rd] = (R[rs1] * R[rs2])(63:0)	1)
mulh	R	MULtiply upper Half	R[rd] = (R[rs1] * R[rs2])(127:64)	
mulhsu	R	MULtiply upper Half Sign/Uns	R[rd] = (R[rs1] * R[rs2])(127:64)	6)
mulhu	R	MULtiply upper Half Unsigned	R[rd] = (R[rs1] * R[rs2])(127:64)	2)
div,divw	R	DIVide (Word)	R[rd] = (R[rs1] / R[rs2])	1)
divu	R	DIVide Unsigned	R[rd] = (R[rs1] / R[rs2])	2)
rem,remw	R	REMainder (Word)	R[rd] = (R[rs1] % R[rs2])	1)
remu,remuw	R	REMainder Unsigned (Word)	R[rd] = (R[rs1] % R[rs2])	1,2)

RV64F and RV64D Floating-Point Extensions

fld,flw	I	Load (Word)	F[rd] = M[R[rs1]+imm]	1)
fsd,fsw	S	Store (Word)	M[R[rs1]+imm] = F[rd]	1)
fadd.s,fadd.d	R	ADD	F[rd] = F[rs1] + F[rs2]	7)
fsub.s,fsub.d	R	SUBtract	F[rd] = F[rs1] – F[rs2]	7)
fmul.s,fmul.d	R	MULtiply	F[rd] = F[rs1] * F[rs2]	7)
fdiv.s,fdiv.d	R	DIVide	F[rd] = F[rs1] / F[rs2]	7)
fsqrt.s,fsqrt.d	R	SQuare RooT	F[rd] = sqrt(F[rs1])	7)
fmadd.s,fmadd.d	R	Multiply-ADD	F[rd] = F[rs1] * F[rs2] + F[rs3]	7)
fmsub.s,fmsub.d	R	Multiply-SUBtract	F[rd] = F[rs1] * F[rs2] - F[rs3]	7)
fmnsub.s,fmnsub.d	R	Negative Multiply-SUBtract	F[rd] = -(F[rs1] * F[rs2] – F[rs3])	7)
fmnadd.s,fmnadd.d	R	Negative Multiply-ADD	F[rd] = -(F[rs1] * F[rs2] + F[rs3])	7)
fsgnj.s,fsgnj.d	R	SiGN source	F[rd] = {F[rs2]<63>,F[rs1]<62:0>}	7)
fsgnjn.s,fsgnjn.d	R	Negative SiGN source	F[rd] = {~F[rs2]<63>, F[rs1]<62:0>}	7)
fsgnjx.s,fsgnjx.d	R	Xor SiGN source	F[rd] = {F[rs2]<63>^F[rs1]<63>, F[rs1]<62:0>}	7)
fmin.s,fmin.d	R	MINimum	F[rd] = (F[rs1] < F[rs2]) ? F[rs1] : F[rs2]	7)
fmax.s,fmax.d	R	MAXimum	F[rd] = (F[rs1] > F[rs2]) ? F[rs1] : F[rs2]	7)
feq.s,feq.d	R	Compare Float EQual	R[rd] = (F[rs1]== F[rs2]) ? 1 : 0	7)
flt.s,flt.d	R	Compare Float Less Than	R[rd] = (F[rs1]< F[rs2]) ? 1 : 0	7)
fle.s,fle.d	R	Compare Float Less than or =	R[rd] = (F[rs1]<= F[rs2]) ? 1 : 0	7)
fclass.s,fclass.d	R	Classify Type	R[rd] = class(F[rs1])	7,8)
fmv.s.x,fmv.d.x	R	Move from Integer	F[rd] = R[rs1]	7)
fmv.x.s,fmv.x.d	R	Move to Integer	R[rd] = F[rs1]	7)
fcvt.s.d	R	Convert from DP to SP	F[rd] = single(F[rs1])	
fcvt.d.s	R	Convert from SP to DP	F[rd] = double(F[rs1])	
fcvt.s.w,fcvt.d.w	R	Convert from 32b Integer	F[rd] = float(R[rs1](31:0))	7)
fcvt.s.l,fcvt.d.l	R	Convert from 64b Integer	F[rd] = float(R[rs1](63:0))	7)
fcvt.s.wu,fcvt.d.wu	R	Convert from 32b Int Unsigned	F[rd] = float(R[rs1](31:0))	2,7)
fcvt.s.lu,fcvt.d.lu	R	Convert from 64b Int Unsigned	F[rd] = float(R[rs1](63:0))	2,7)
fcvt.w.s,fcvt.w.d	R	Convert to 32b Integer	R[rd](31:0) = integer(F[rs1])	7)
fcvt.l.s,fcvt.l.d	R	Convert to 64b Integer	R[rd](63:0) = integer(F[rs1])	7)
fcvt.wu.s,fcvt.wu.d	R	Convert to 32b Int Unsigned	R[rd](31:0) = integer(F[rs1])	2,7)
fcvt.lu.s,fcvt.lu.d	R	Convert to 64b Int Unsigned	R[rd](63:0) = integer(F[rs1])	2,7)

CORE INSTRUCTION FORMATS

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	opcode
I	imm[11:0]			rs1	funct3	rd	opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

PSEUDO INSTRUCTIONS

MNEMONIC	NAME	DESCRIPTION	USES
beqz	Branch = zero	if(R[rs1]==0) PC=PC+{imm,1b'0}	beq
bnez	Branch ≠ zero	if(R[rs1]!=0) PC=PC+{imm,1b'0}	bne
fabs.s,fabs.d	Absolute Value	F[rd] = (F[rs1]< 0) ? –F[rs1] : F[rs1]	fsgnx
fmv.s,fmv.d	FP Move	F[rd] = F[rs1]	fsgnj
fneg.s,fneg.d	FP negate	F[rd] = –F[rs1]	fsgnjn
j	Jump	PC = {imm,1b'0}	jal
jr	Jump register	PC = R[rs1]	jalr
la	Load address	R[rd] = address	auipc
li	Load imm	R[rd] = imm	addi
mv	Move	R[rd] = R[rs1]	addi
neg	Negate	R[rd] = –R[rs1]	sub
nop	No operation	R[0] = R[0]	addi
not	Not	R[rd] = ~R[rs1]	xori
ret	Return	PC = R[1]	jalr
seqz	Set = zero	R[rd] = (R[rs1]== 0) ? 1 : 0	sltiu
snez	Set ≠ zero	R[rd] = (R[rs1]!= 0) ? 1 : 0	sltu

REGISTER NAME, USE, CALLING CONVENTION

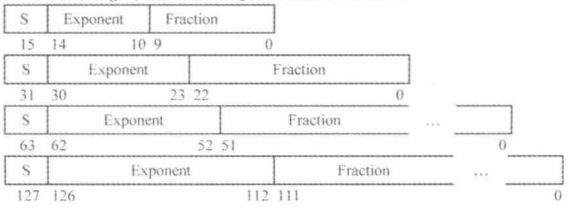
REGISTER	NAME	USE	SAVER
x0	zero	The constant value 0	N.A.
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	--
x4	tp	Thread pointer	--
x5-x7	t0-t2	Temporaries	Caller
x8	s0/fp	Saved register/Frame pointer	Callee
x9	s1	Saved register	Callee
x10-x11	a0-a1	Function arguments/Return values	Caller
x12-x17	a2-a7	Function arguments	Caller
x18-x27	s2-s11	Saved registers	Callee
x28-x31	t3-t6	Temporaries	Caller
f0-f7	ft0-ft7	FP Temporaries	Caller
f8-f9	fs0-fs1	FP Saved registers	Callee
f10-f11	fa0-fa1	FP Function arguments/Return values	Caller
f12-f17	fa2-fa7	FP Function arguments	Caller
f18-f27	fs2-fs11	FP Saved registers	Callee
f28-f31	ft8-ft11	R[rd] = R[rs1] + R[rs2]	Caller

IEEE 754 FLOATING-POINT STANDARD ④

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

where Half-Precision Bias = 15, Single-Precision Bias = 127, Double-Precision Bias = 1023, Quad-Precision Bias = 16383

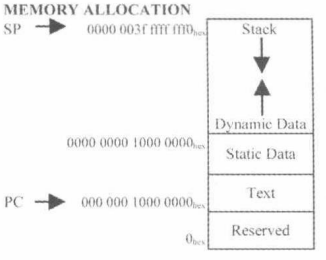
IEEE Half-, Single-, Double-, and Quad-Precision Formats:



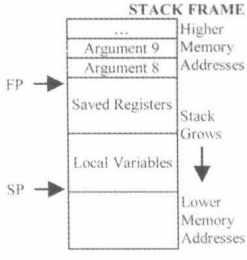
OPCODES IN NUMERICAL ORDER BY OPCODE ③

MNEMONIC	FMT	OPCODE	FUNCT3	FUNCT7 OR IMM	HEXADECIMAL
lb	I	0000011	000		03/0
lh	I	0000011	001		03/1
lw	I	0000011	010		03/2
ld	I	0000011	011		03/3
lbu	I	0000011	100		03/4
lhu	I	0000011	101		03/5
lwu	I	0000011	110		03/6
fence	I	0001111	000		0F/0
fence.i	I	0001111	001		0F/1
addi	I	0010011	000		13/0
slli	I	0010011	001	0000000	13/1/00
slti	I	0010011	010		13/2
sltiu	I	0010011	011		13/3
xori	I	0010011	100		13/4
srli	I	0010011	101	0000000	13/5/00
srai	I	0010011	101	0100000	13/5/20
ori	I	0010011	110		13/6
andi	I	0010011	111		13/7
auipc	U	0010111			17
addiw	I	0011011	000		1B/0
slliw	I	0011011	001	0000000	1B/1/00
srliw	I	0011011	101	0000000	1B/5/00
sraiw	I	0011011	101	0100000	1B/5/20
sb	S	0100011	000		23/0
sh	S	0100011	001		23/1
sw	S	0100011	010		23/2
sd	I	0100011	011		23/3
add	R	0110011	000	0000000	33/0/00
sub	R	0110011	000	0100000	33/0/20
sll	R	0110011	001	0000000	33/1/00
slt	R	0110011	010	0000000	33/2/00
sltu	R	0110011	011	0000000	33/3/00
xor	R	0110011	100	0000000	33/4/00
srl	R	0110011	101	0000000	33/5/00
sra	R	0110011	101	0100000	33/5/20
or	R	0110011	110	0000000	33/6/00
and	R	0110011	111	0000000	33/7/00
lui	U	0110111			37
addw	R	0111011	000	0000000	3B/0/00
subw	R	0111011	000	0100000	3B/0/20
sllw	R	0111011	001	0000000	3B/1/00
srlw	R	0111011	101	0000000	3B/5/00
sraw	R	0111011	101	0100000	3B/5/20
beq	SB	1100011	000		63/0
bne	SB	1100011	001		63/1
blt	SB	1100011	100		63/4
bge	SB	1100011	101		63/5
bltu	SB	1100011	110		63/6
bgeu	SB	1100011	111		63/7
jalr	I	1100111	000		67/0
jal	UJ	1101111			6F
ecall	I	1110011	000	000000000000	73/0/000
ebreak	I	1110011	000	000000000001	73/0/001
CSRRW	I	1110011	001		73/1
CSRRS	I	1110011	010		73/2
CSRRC	I	1110011	011		73/3
CSRRWI	I	1110011	101		73/5
CSRRSI	I	1110011	110		73/6
CSRRCI	I	1110011	111		73/7

MEMORY ALLOCATION



STACK FRAME



文艺复兴以来，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取得了垄断性的优势；也正是这样的优势，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，不仅擘划了研究的范畴，还揭示了学术的源变，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和发展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到积极的推动作用，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公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson、McGraw-Hill、Elsevier、MIT、John Wiley & Sons、Cengage 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 Andrew S. Tanenbaum、Bjarne Stroustrup、Brian W. Kernighan、Dennis Ritchie、Jim Gray、Afred V. Aho、John E. Hopcroft、Jeffrey D. Ullman、Abraham Silberschatz、William Stallings、Donald E. Knuth、John L. Hennessy、Larry L. Peterson 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、研究及珍藏。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相助，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，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近 500 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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。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，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

赞 誉

Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition

教材的选择往往是一个令人沮丧的妥协过程——教学方法的适用度、知识点的覆盖范围、文辞的流畅性、内容的严谨度、成本的高低等都需要考虑。本书之所以是难得一见的好书，正是因为它能满足各个方面的要求，不再需要任何妥协。这不仅是一部关于计算机组成的教科书，也是所有计算机科学教科书的典范。

——Michael Goldweber, Xavier University

我从第 1 版开始就在使用本书，到现在已经很多年了。这次的版本是对已有经典内容的又一次完美升级。从桌面计算到移动计算再到大数据、云计算，技术的发展为嵌入式处理器（如 ARM）开拓了新的应用领域，为通过软硬件交互来提高性能带来了新素材。所有这些都离不开基本的组成原理。

——Ed Harcourt, St. Lawrence University

无论是对于 80 后、90 后还是 00 后，这都是一本应该珍藏在书架上（或 iPad 中）的计算机体系结构教材。这本书既古老又新颖，不仅介绍了那些伟大的原理——摩尔定律、抽象、加速经常性事件、冗余、存储层次、并行和流水线，而且使用现代设计对这些伟大原理进行了说明。

——Mark D. Hill, University of Wisconsin-Madison

本书的新版本与新兴的嵌入式和众核（GPU）系统的发展保持同步，当前的发展趋势使得平板电脑和智能手机很快变成新的桌面电脑。本书接纳了这些变化，但仍介绍了大量计算机组成与设计的基本原理，这对于新设备和新系统的软硬件设计人员来说非常有用。

——Dave Kaeli, Northeastern University

本书不仅讲解计算机体系结构，而且为读者准备了迎接新的变化与挑战的“锦囊”。目前，半导体工艺技术按比例缩小的困难使得所有系统功率受限，而移动系统和大数据处理的性能需求却仍在不断增长。在计算技术的新时代，必须进行软硬件协同设计，并且系统级体系结构优化与部件级优化一样重要。

——Christos Kozyrakis, Stanford University

Patterson 和 Hennessy 讨论了不断变化的计算机硬件体系结构中的重要议题，强调硬件和软件模块在不同抽象层次上的交互。书中涵盖各种硬件和软件机制，I/O 和并行的概念贯穿其中，全景式呈现了后 PC 时代的计算机体系结构。无论是平板电脑硬件工程师还是云计算软件架构师，如果你正对能源效率和并行化问题一筹莫展，那么本书必将成为不二之选。

——Jae C. Oh, Syracuse University

D. Patterson 和 J. Hennessy 是计算机领域的知名学者，他们为计算机体系结构设计和评估以及产业发展做出了巨大贡献，并产生了持久影响。两位学者因此获得了 2017 年图灵奖，这是当时计算机体系结构领域的一大盛事。两位教授合著的 *Computer Organization and Design: The Hardware/Software Interface* 及其姊妹篇 *Computer Architecture: A Quantitative Approach* 堪称计算机体系结构学科的“圣经”，是计算机系统设计从业者的必读经典。自 20 世纪 90 年代至今，北京大学在国内率先使用这两本著作作为本科及研究生计算机体系结构课程的教材，译者以学生、助教和教师的不同身份经历了本书的多个版本变迁，深感受益匪浅。

D. Patterson 及其 UC Berkeley 的团队提出了 RISC（Reduced Instruction Set Computer，精简指令系统计算机），并于 1982 年推出了 RISC-I 处理器。J. Hennessy 于 1984 年联合创立了 MIPS 公司，而 UC Berkeley 团队的研究成果则通过 Sun 公司的 SPARC 系列处理器在产业界发挥作用。这些都对后续诸多处理器的设计产生了极为深远的影响。2014 年，在 D. Patterson 教授的领导下，UC Berkeley 设计并推出了 RISC-V 开放指令系统。该指令系统继续秉承前四代指令系统简单规整的特点，同时又尽可能摒弃之前指令系统的各种缺陷，具有短小精悍、便于扩展、易于实现等新特点。值得一提的是，该指令系统完全开源，这大大地推动了硬件开源设计的发展。

相应地，本书第 5 版在 MIPS 和 ARM 版的基础上，特别推出了 RISC-V 版，这满足了广大读者学习和了解新技术及其发展的需要。该版本使用 RISC-V 指令系统作为实例，抽丝剥茧般呈现了设计一套新指令系统所需的技术考虑及其与微体系结构之间的密切联系，真正做到了“知其然，知其所以然”，这正是国外优秀的计算机体系结构研究者的“底蕴”。作为教育界同行，我们非常希望本书能够帮助国内的读者积累这样的底蕴。D. Patterson 教授曾经表示过，RISC-V 的未来在中国。我们也特别希望这部经典著作之 RISC-V 版中译本的出版，能够对我国的软硬件生态建设和发展有所贡献，能够对计算机体系结构领域的教学和科研有所帮助。

感谢机械工业出版社华章分社一直关注本书的引进和中译本的出版工作，感谢温莉芳总编辑为我们提供了这次宝贵的机会，感谢曲熠等编辑仔细阅读译稿，为中译本的翻译工作提出了大量有建设性的意见。

同时我们还要感谢清华大学郑纬民教授对前 3 版中译本所做的工作，感谢西北工业大学康继昌教授、樊晓桠教授和安建峰副教授对第 4 版中译本所做的工作，感谢西北工业大学王党辉副教授、国防科技大学陈微教授分别翻译了第 5 版的 MIPS 及 ARM 版，是他们的工作为本书提供了良好的参考范本，并为其在国内拥有更广泛的读者群奠定了基础。

北京大学信息科学技术学院计算机系的周叔欣、周昱晨、戚妙、张馨月、张炜奇、赵宏烨等也参加了本书的翻译和校对工作。

由于译者水平有限，书中难免存在一些翻译不当或理解欠妥的地方，恳请读者批评指正。

译者

2020 年 4 月于燕园

前　言

Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition

我们能体验的最美好的事情莫过于神秘，它是所有真实的艺术和科学的源泉。

——阿尔伯特·爱因斯坦，《我的信仰》，1930 年

关于本书

我们认为，在学习计算机科学与工程时，除了掌握计算的基本原理外，还应该了解该领域的最新进展。同时，我们还认为，各种计算领域中的读者都应学习计算机系统的组成理论，因为这是决定计算机系统的功能、性能、能耗甚至最终成功与否的关键。

现代计算机技术需要各个计算领域的专业人员对计算机软件和硬件都有所了解。软硬件在不同层次上的相互影响，恰好也提供了一个理解计算基础的框架。不管你的关注点是硬件还是软件，专业是计算机科学还是电子工程，计算机组成和设计的核心思想都是相同的。因此，本书的重点是展示硬件和软件之间的关系，并重点关注现代计算机的基本概念。

本书从第 1 版起就提出了以上观点，最近从单处理器向多核微处理器的转变再一次见证了这个颇有远见的观点。然而，编程人员无视我们的忠告，不想改造程序，只想依赖计算机体系结构设计者、编译器设计者或者芯片设计者来让自己的程序运行得更快、能效性更好——这样的时代已经一去不复返了。为了运行得更快，需要把程序改造成并行的。让程序员尽可能不知道它们正在使用的底层硬件的并行属性，这是许多研究者的目标，但这需要花费很长时间才能实现。我们的观点是，至少在接下来的十年里，如果想让程序在并行计算机上运行得更为高效，大多数编程人员还是需要了解硬件 / 软件接口的。

本书的读者包括不了解汇编语言或者逻辑设计，但需要了解计算机基本组成的人；同时也包括拥有汇编语言或者逻辑设计背景，但想学习如何设计计算机或者想搞清楚系统的工作原理及原因的人。

关于另一本书

有些读者可能已经熟悉我们的另一本书——*Computer Architecture: A Quantitative Approach*（《计算机体系结构：量化研究方法》，后文简称为《量化研究》）。该书已广为流传，经常以作者姓名命名，称为 Hennessy 和 Patterson。（本书则常被称为 Patterson 和 Hennessy。）我们写《量化研究》的动机是，希望能够使用坚实的工程基础、成本 / 性能之间的量化分析和折中来描述计算机体系结构的基本原则。我们使用的方法是，基于商业系统，将实例与评估相结合，建立真实的设计体验。我们的目标是证实可以使用量化分析的方法而不是描述性方法来学习计算机体系结构。希望这一方法有助于培养能精准理解计算机的专业人才。

本书的大多数读者并不一定要成为计算机体系结构设计者。但是，未来软件系统的性能和能效性，很大程度上取决于软件设计者对所使用系统的基本硬件技术的了解程度。因此，编译器设计者、操作系统设计者、数据库编程人员以及大多数其他软件设计人员需要对本书中提到的基本原则有深入的理解。同样，硬件设计人员也需要清楚地知道自己的设计对软件应用程序的影响。

因此，本书不仅仅是《量化研究》一书的子集，我们已经对本书进行了大幅修订以满足不同读者的需要。我们非常高兴地看到《量化研究》的后续版本也在不断修订，删除了大量的介绍性材料。相比第 1 版，此后两本书之间的内容重叠会越来越少。

关于 RISC-V 版本

选择合适的指令系统对于计算机体系结构教材来说至关重要。不管是否为主流指令系统，我们都不希望介绍那些具有不必要的新奇特性的指令系统。理想情况是，你学习的第一个指令系统应该是一个典范，就像你的初恋一样。令人惊讶的是，你学习的第一个指令系统和你的初恋都会令你分外怀念。

由于当时有太多选择，所以在《量化研究》的第 1 版中我们提出了自己的 RISC 风格指令系统。之后，MIPS 指令系统因其简洁的风格而日益受到关注，我们在本书第 1 版时选择了它，并且《量化研究》的后续版本也是如此。MIPS 一直为我们和读者提供了很好的服务。

20 年来，使用 MIPS 指令系统的芯片成千上万，并且还在不断生产出来，它们一般用于嵌入式设备，而该领域的指令系统几乎不可见，因此，目前很难找到一台真实的计算机，让读者能够下载并运行 MIPS 程序。

好消息是，最近一个开放的 RISC 指令系统首次亮相，并快速获得了不少追随者。它就是由加州大学伯克利分校（UC Berkeley）开发的 RISC-V 指令系统，它不仅消除了 MIPS 指令系统的弊病，而且还具备指令系统应有的简洁、优雅和现代的特点。

RISC-V 指令系统不是闭源的，它提供了一套开源的模拟器、编译器、调试器等，这些都很容易获得。它甚至还提供开源的使用硬件描述语言编写的 RISC-V 处理器实现。除此之外，很快还会提供低成本的硬件平台，供运行 RISC-V 程序使用。读者不仅可以学习这些设计，还能修改它们并贯穿整个实现流程，以充分了解这些修改对性能、晶片面积和能耗方面的影响。

这对于计算产业和教育行业来说是一个令人激动的机会。截止到写这篇前言之时，已经有 40 多家公司加入到 RISC-V 基金会中，赞助商名单几乎囊括了除 ARM 和 Intel 以外的所有主要厂商，包括 AMD、Google、HP、IBM、Microsoft、NVIDIA、Oracle 和 Qualcomm 公司。

正是因为这些，我们为本书撰写了 RISC-V 版本，同样，《量化研究》也有对应版本。

RISC-V 同时提供 32 位和 64 位指令系统，它们的指令类型基本相同。我们可以切换指令系统，并保持地址宽度为 32 位。我们的出版商调查了本书的读者群，发现 75% 的读者首选大型或中型地址空间，因此我们选用 64 位地址空间，这比 32 位地址空间更有意义。

相比 MIPS 版本，RISC-V 版本唯一的修改就是那些与指令系统相关的描述，主要影响第 2、3 和 5 章中的虚拟存储部分，以及第 6 章中的 VMIPS 示例。在第 4 章中，我们改用 RISC-V 指令，修改了相关的图表，添加了一些“详细阐述”模块，这些变化没有我们想象中那么复杂。第 1 章和其余的附录几乎没有变化。由于存在大量的在线文档，并且与 RISC-V 相关的修改过多，这使得 MIPS 版本中的附录 A 很难被替换（附录 A 是指“汇编器、链接器和 SPIM 模拟器”，详见 MIPS 第 5 版）。另外，第 2、3 和 5 章中包含上百条 RISC-V 指令的快速概览，这些指令都不在本书详细介绍的 RISC-V 核心指令范围内。

请注意，我们并没有正式地、永久地切换到 RISC-V 指令系统。比如，除了新出的 RISC-V 版本，目前还可以购买本书的 ARMv8 版和 MIPS 版。未来存在如下可能性：读者需

要所有不同指令系统的版本，或可能只需要一个版本。船到桥头自然直，目前我们期待你对此次修改的反馈。

第 5 版中的改变

在本书的第 5 版中我们有 6 个主要的修订目标：

- 使用实际运行的实例来证实理解硬件的重要性。
- 对于第 1 章中提到的 8 个伟大思想，在后面每次应用这些思想时予以突出显示。
- 更新实例以反映从 PC 时代到后 PC 时代的转变。
- 将和 I/O 有关的资料分布到书中各章节，而不是将其作为单独的一章。
- 更新技术内容，以反映自 2009 年第 4 版出版以来工业界的变化。
- 将附录和可选章节放到网上，而不是放到 CD 中，这样可降低成本，并使得本书和电子书一样方便。

在详细讨论这些修订目标之前，我们先看一看下表。它展示的是分别为软硬件人员设计的导读方案。不论自身经验和关注点如何，第 1、4、5 和 6 章都出现在每一种导读方案中。第 1 章讨论了能耗的重要性，并介绍它是如何促使微处理器从单核向多核转变的，同时还会介绍计算机体系结构的 8 个重要思想。第 2 章对于硬件读者来说可能是复习性材料，但对于软件读者来说则是必读材料，特别是那些对学习编译器和面向对象编程语言感兴趣的读者。第 3 章适合那些对数据通路或者浮点运算感兴趣的读者，有些读者可能会跳过第 3 章。不过，我们将在这一章介绍关于矩阵乘法的运行实例，展示如何采用子字并行的方法将性能提高 4 倍，因此不要跳过 3.6 ～ 3.8 节。第 4 章介绍了流水线处理器。4.1 节、4.5 节和 4.10 节给出了概述，4.12 节为那些关心软件的读者提供了进一步改善矩阵乘法性能的方法。对于硬件人员来说，第 4 章是核心内容。另外，读者可以根据知识背景的不同，选择是否首先阅读附录 A 中的逻辑设计部分。最后一章讨论多核、多处理器和集群系统，这一章是全新的内容，因此所有读者都应该阅读。本版中进行了重大组织结构调整，这使得许多思想的引入更加自然，阐述更加深入，例如 GPU、仓储级计算机以及集群系统中的关键——网卡的软硬件接口设计。

章或附录	节	软件人员	硬件人员
第 1 章 计算机抽象及相关技术	1.1 ～ 1.11	👓	👓
	1.12（历史）	👓	👓
第 2 章 指令：计算机的语言	2.1 ～ 2.14	👓	👓
	2.15（编译器和 Java）	👓	
	2.16 ～ 2.20	👓	👓
	2.21（历史）	👓	👓
附录 D 精简指令系统体系结构计算机	D.1 ～ D.17	👓	
第 3 章 计算机的算术运算	3.1 ～ 3.5	👓	👓
	3.6 ～ 3.8（子字并行）	👓	👓
	3.9 ～ 3.10（谬误）	👓	👓
	3.11（历史）	👓	👓
附录 A 逻辑设计基础	A.1 ～ A.13		👓

（续）

章或附录	节	软件人员	硬件人员
第 4 章 处理器	4.1（概述）		
	4.2（逻辑设计基础）		
	4.3 ～ 4.4（简单实现）		
	4.5（流水线概述）		
	4.6（流水线数据通路）		
	4.7 ～ 4.9（冒险、例外）		
	4.10 ～ 4.12（并行、实例）		
	4.13（Verilog 描述的流水线控制）		
	4.14 ～ 4.15（谬误）		
	4.16（历史）		
附录 C 将控制映射至硬件	C.1 ～ C.6		
第 5 章 大而快：层次化存储	5.1 ～ 5.10		
	5.11（廉价磁盘冗余阵列）		
	5.12（Verilog 描述的缓存控制器）		
	5.13 ～ 5.17		
	5.18（历史）		
第 6 章 并行处理器：从客户端到云	6.1 ～ 6.8		
	6.9（网络）		
	6.10 ～ 6.14		
	6.15（历史）		
附录 B 图形处理单元	B.1 ～ B.13		

仔细阅读  　闲时阅读  　参考阅读 

回顾阅读 　背景阅读

本书第 5 版的 6 大修订目标中，首要目标是使用实例证实理解现代硬件对于提高性能和能效的重要性。正如之前提到的，为将矩阵乘法的性能提高 4 倍，我们在第 3 章中介绍了子字并行技术。在第 4 章中，我们通过循环展开将性能又提高 2 倍，这说明了指令级并行技术的有效性。在第 5 章中，通过使用 cache 块优化技术，我们将矩阵乘法的性能再次成倍提升。最后，在第 6 章中，我们使用 16 个处理器通过线程级并行技术获得了 14 倍的性能加速。上述 4 项优化加在一起，只在原始矩阵乘法示例中添加了 24 行 C 代码。

第二个修订目标是，指出计算机体系结构中的 8 个重要思想，并在其出现时予以突出显示，以此来帮助读者从树木中找出森林，尽早建立对计算机系统的整体认知。本书中有将近 100 次提及这 8 个重要思想，每一章中对重要思想的举例都不少于 7 个，每一个重要思想都至少被提及 5 次。通过并行、流水线和预测技术改善性能是三种最流行的重要设计思想，另一个重要思想就是摩尔定律。在第 4 章中，与处理器相关的例子是最多的。这并不奇怪，因为计算机体系结构设计者最关注的可能就是处理器。在每个章节都可以发现的一个重要思想就是通过并行改善性能。这是个令人愉快的现象，因为最近在本领域和本书的这个版本中都在强调并行性。

本书的第三个修订目标是，通过我们的实例和材料使读者认识到计算技术正从 PC 时代向后 PC 时代转变。因此，第 1 章深入到平板电脑而不是个人电脑内部，第 6 章描述了云计

算基础设施。我们还介绍了 ARM，这是后 PC 时代个人移动设备的指令系统选择，与之对应的是，x86 是 PC 时代和云计算时代的主流指令系统。

第四个修订目标是将 I/O 相关内容分散到整本书中，而不是集中在独立的一章。这和我们在第 4 版中将并行相关内容分散到整本书中一样。本书的 I/O 相关内容可以在 1.4 节、4.9 节、5.2 节、5.5 节、5.11 节和 6.9 节中找到。这样做的初衷是，如果 I/O 没有独立成章，那么读者（和教师）可能更容易阅读和使用这些内容。

计算机体系结构是一个快速发展的领域，相关书籍的重要目标之一就是更新技术内容。新版本的书也需要如此，这就是我们的第五个目标。本书选择的运行实例是对后 PC 时代产生重要影响的 ARM Cortex A53 和 Intel Core i7。其他重要内容包括一份关于 GPU 的教程，它解释了 GPU 中独有的术语；此外，我们还更为深入地介绍了构成云的仓储级计算机和万兆字节以太网卡。

最后一个目标是，为保持本书主体部分简短，并与电子书兼容，我们将可选阅读材料作为附录放在网上，而不是像之前那样保存在附带的 CD 上。

最后，我们还更新了本书的所有练习。

虽然发生了不少变化，但我们依然保留了旧版本中那些有用的元素。为使本书更适合作为参考书籍，我们仍在新术语第一次出现时在页边给出它们的定义。书中名为“理解程序性能”的模块是用来帮助读者理解如何改善程序性能的，同样，“硬件 / 软件接口”模块是用来帮助读者了解该接口并进行软硬件划分权衡的。书中仍保留“重点”模块，帮助读者从一片树木中看出森林。“自我检测”部分在每一章的结尾都附有答案，可以帮助读者第一时间确认自己对书中内容的理解。本版本也包括了 RISC-V 的参考数据卡[⊖]，这是受到了 IBM 360 系统“绿卡”的启发。该卡片已更新，适于在书写 RISC-V 汇编程序时作为参考资料使用。

教学支持[⊖]

我们收集了大量的资料，帮助教师使用本书进行授课。只要教师在出版商处进行注册，就能获得练习答案、书中的图表、教学讲义以及其他资料。除此之外，配套网站还提供了开源 RISC-V 软件的下载链接。如需更多信息，可访问出版商网站：booksite.elsevier.com/9780128122754。

结束语

从后续的致谢一节可以看到，我们花了大量的时间来进行修订。这本书多次再版，经过了多次印刷，我们也有机会不断完善。如果你在此版中发现任何一处错误或漏洞，可通过电子邮件 codRISCVbugs@mkp.com 联系出版商。

本书的这一版本是 Hennessy 和 Patterson 自 1989 年长期合作以来的第三次突破。由于要管理一所世界一流大学，这意味着 Hennessy 校长无法再对编写一个新版本做出实质性的承诺。这使得另一位作者再次感到自己像一个没有安全措施就走上钢丝的人。因此，伯克利的同事和致谢中提到的人在本书的内容方面发挥了更大的作用。无论如何，这一次只有一位作者会对你所阅读的新内容负责。

⊖ 参考数据卡见本书封面和封底的背面。——编辑注

⊖ 关于本书教辅资源，只有使用本书作为教材的教师才可以申请，需要的教师请访问爱思唯尔的教材网站 https://textbooks.elsevier.com/ 进行申请。——编辑注

致谢

对于本书的每一个版本，我们都很幸运地得到了许多读者、审稿人和其他贡献者的帮助。这些人的帮助使这本书变得更好。

我们非常感谢 Khaled Benkrid 以及他在 ARM 公司的同事，他们非常认真地检查了 ARM 的相关材料，并提供了有益的反馈。

我们对第 6 章进行了大幅修订，对其中的观点和内容分别进行了审查，针对每一位审稿人的意见都进行了修改。感谢 Christos Kozyrakis（Stanford University），是他建议我们使用集群系统的网络接口作为实例来说明 I/O 的软硬件接口划分，并对该章的内容组织给出了意见。感谢 Mario Flagsilk（Stanford University），是他提供了和 NetFPGA NIC 平台相关的技术细节、图表和性能评估数据。感谢以下人员对第 6 章提出了修改建议：David Kaeli（Northeastern University）、Partha Ranganathan（HP Labs）、David Wood（University of Wisconsin），以及我在 Berkeley 的同事 Siamak Faridani、Shoaib Kamil、Yunsup Lee、Zhangxi Tan 和 Andrew Waterman。

特别感谢 Rimas Avizenis（UC Berkeley），是他改写了各种版本的矩阵乘法并提供了性能数据。当我还是 UCLA 的研究生时，我和他的父亲一起工作，能与 Rimas 在 UC Berkeley 共事是一件美好的事情。

同时，还要感谢我的长期合作者——Randy Katz（UC Berkeley），他帮助我提出了“计算机体系结构中的重要思想”的概念，并将其作为我们一起合作开设的本科生课程的教学内容。

我还要感谢 David Kirk 和 John Nickolls 以及他们在 NVIDIA 的同事（Michael Garland、John Montrym、Doug Voorhies、Lars Nyland、Erik Lindholm、Paulius Micikevicius、Massimiliano Fatica、Stuart Oberman 和 Vasily Volkov），他们编写了第一个深入阐述 GPU 的附录。我希望能再次表达我对 Jim Larus 的感激之情，最近他刚被任命为 EPFL 的计算机和通信科学学院院长。感谢他愿意在汇编语言编程方面贡献自己的专业知识，并欢迎本书的读者使用他开发及维护的模拟器。

还要感谢 Zachary Kurmas（Grand Valley State University），是他在原始版本的基础上更新并重新设计了新的章末练习。原始版本由 Perry Alexander（University of Kansas）、Jason Bakos（University of South Carolina）、Javier Bruguera（Universidade de Santiago de Compostela）、Matthew Farrens（University of California，Davis）、David Kaeli（Northeastern University）、Nicole Kaiyan（University of Adelaide）、John Oliver（Cal Poly, San Luis Obispo）、Milos Prvulovic（Georgia Tech）、Jichuan Chang（Google）、Jacob Leverich（Stanford）、Kevin Lim（Hewlett-Packard）和 Partha Ranganathan（Google）开发。

特别的感谢送给 Peter Ashenden，他帮我们更新了课程讲义。

感谢下面这些老师，他们回复了出版商的调查问卷，审阅了我们的提议，还参加了各种专题小组。专题小组人员如下：Bruce Barton（Suffolk County Community College），Jeff Braun（Montana Tech），Ed Gehringer（North Carolina State），Michael Goldweber（Xavier University），Ed Harcourt（St. Lawrence University），Mark Hill（University of Wisconsin，Madison），Patrick Homer（University of Arizona），Norm Jouppi（HP Labs），Dave Kaeli（Northeastern University），Christos Kozyrakis（Stanford University），Jae C. Oh（Syracuse University），Lu Peng（LSU），Milos Prvulovic（Georgia Tech），Partha Ranganathan（HP Labs），David Wood（University of Wisconsin），Craig Zilles（University of Illinois at Urbana-