



计 算 机 科 学 丛 书

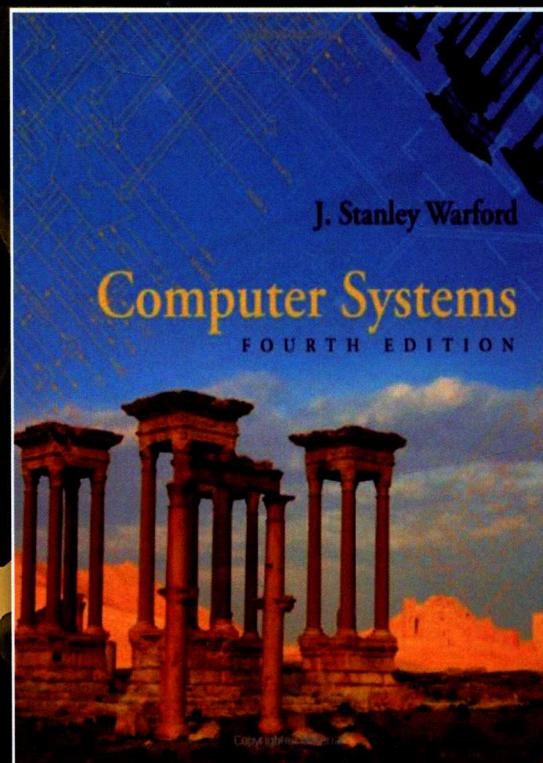
原书第4版

# 计算机系统

## 核心概念及软硬件实现

[美] J. 斯坦利·沃法德 (J. Stanley Warford) 著 龚奕利 译  
佩珀代因大学 武汉大学

Computer Systems  
Fourth Edition



机械工业出版社  
China Machine Press

计 算 机 科 学 丛 书

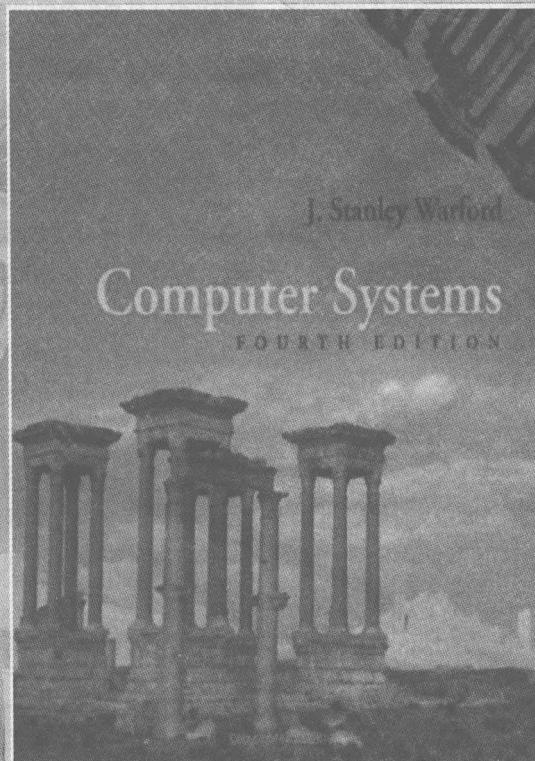
原书第4版

# 计算机系统

## 核心概念及软硬件实现

[美] J. 斯坦利·沃法德 (J. Stanley Warford) 著 龚奕利 译  
佩珀代因大学 武汉大学

Computer Systems  
Fourth Edition



机械工业出版社  
China Machine Press

## 图书在版编目 (CIP) 数据

计算机系统：核心概念及软硬件实现（原书第4版）/（美）沃法德（Warford, J. S.）著；龚奕利译。—北京：机械工业出版社，2015.7  
(计算机科学丛书)

书名原文：Computer Systems, Fourth Edition

ISBN 978-7-111-50783-3

I. 计… II. ① 沃… ② 龚… III. 计算机系统 IV. TP30

中国版本图书馆 CIP 数据核字（2015）第 150961 号

**本书版权登记号：图字：01-2013-5990**

Original English language edition published by Jones & Bartlett Learning, LLC, 5 Wall Street, Burlington, MA 01803.

J. Stanley Warford: Computer Systems, Fourth Edition (ISBN 978-0-7637-7144-7).

Copyright © 2010 by Jones & Bartlett Learning, LLC.

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2015 by China Machine Press.

本书中文简体字版由 Jones & Bartlett Learning, LLC 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

本书基于虚构的计算机 Pep/8，清晰、详细、循序渐进地介绍了计算机组成、汇编语言和计算机体系结构中的核心思想，围绕 7 个抽象层次组织内容，详细介绍了计算机系统的应用层、高级语言层、汇编层、操作系统层、指令集架构层、微代码层和逻辑门层。本书有完整的程序示例，理论和实践相结合，宽度和深度相结合，提供了对普适的冯·诺依曼机器架构的深入理解。

本书可作为高等院校计算机科学专业本科生的教材，也可作为相关专业人员学习计算机基础知识的参考书。

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：盛思源 曲 煜

责任校对：董纪丽

印 刷：三河市宏图印务有限公司

版 次：2015 年 7 月第 1 版第 1 次印刷

开 本：185mm×260mm 1/16

印 张：31

书 号：ISBN 978-7-111-50783-3

定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 88361066

投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259

读者信箱：hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本法律顾问：北京大成律师事务所 韩光 / 邹晓东

一个很自然的问题就是：为什么英文版出版多年之后本书的内容仍然没有过时呢？它现在没有过时，今后许多年也不会过时。原因有两点：

首先，本书讲授计算机系统的基础知识，这些基础知识数十年都没有改变过。所有的计算机文艺复兴以来，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，从他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, Alfred 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，有的还专门为本书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，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。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](http://www.hzbook.com)

电子邮件：[hzjsj@hzbook.com](mailto: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



华章教育

华章科技图书出版中心

# 中文版序

Computer Systems, Fourth Edition

## Computer Systems, Fourth Edition

It has been several years since the publication of the Fourth Edition of *Computer Systems*.

During that time computer technology has continued to advance. The natural question for this translation is, Why is *Computer Systems* still current after these years since its English publication? There are two reasons why it is current and will remain current for many years.

First, *Computer Systems* is still current because it teaches the fundamentals of computer systems that have been constant for decades. All computer systems consist of software and hardware, and all commercial hardware systems are based on the ubiquitous von Neumann cycle.

On the software side, all high-order languages manipulate data with iterative and recursive algorithms. All languages must be translated to lower level machine language to be executed or interpreted. On the hardware side, all physical machines are combinational and sequential circuits built from logic gates. No matter what the technology, the design principles behind these software and hardware systems do not change.

Second, *Computer Systems* is still current because it teaches the above design principles with the Pep/8 virtual machine. The Pep/8 machine will always be current because it is not subject to obsolescence, but rather is built to teach the fundamentals of computer systems that do not change. Specifically, it presents a computer system as seven levels of abstraction:

- Applications
- High-order languages
- Assembly
- Operating system
- Instruction set architecture
- Microcode
- Logic gate

The book illustrates these levels of abstraction using C++ as the high-order language and Pep/8 assembly language and machine language at the lower levels. The strength of this approach is that the central concepts of computer science are taught without getting entangled in the many irrelevant details that often accompany courses that focus on current technology rather than computing fundamentals. Students who learn the fundamentals are better equipped to master whatever new technology they will encounter in their future computing careers than they would be if they studied only the technology that is current when they are students.

本书第4版完成已经有几年的时间了，这期间计算机技术持续发展。要出版中文版，一个很自然的问题就是：为什么英文版出版多年之后本书的内容仍然没有过时呢？它现在没有过时，今后许多年也不会过时，原因有两点。

首先，本书讲授计算机系统的基础知识，这些基础知识数十年都没有改变过。所有的计算机系统都是由软件和硬件组成的，所有的商用硬件系统都是基于普适的冯·诺依曼周期。

从软件方面来说，所有的高级语言都以迭代或递归算法来处理数据。所有的语言都必须翻译成更低级的机器语言才能执行或解释。从硬件方面来说，所有的物理机器都是由逻辑门构成的组合或时序电路。无论技术如何发展，这些软件和硬件系统背后的设计原理没有改变。

其次，本书以Pep/8虚拟机为例来讲授上述设计原理。Pep/8机器不会过时，因为它不受时间限制，设计它的初衷就是讲授计算机系统中不会变化的基本知识。具体来说，它展现的是计算机系统在7个抽象层次上的样子：

- 应用层
- 高级语言层
- 汇编层
- 操作系统层
- 指令集架构层
- 微代码层
- 逻辑门层

本书在描述各个抽象层次时使用C++作为高级语言，在较低层上使用Pep/8汇编语言和机器语言。这种方法的好处就是讲授计算机科学的核心概念，而又不会纠结于无关的细节，很多把重点放在当前技术而不是计算基础的课程通常都会有这种问题。比起只学习当前技术的学生，学习了基础知识的学生在以后的工作中能更好地掌握遇到的新技术。

J. Stanley Warford

**App7层** App7层是单独一章，介绍了应用程序。本章展示了抽象层的概念，建立本书剩下部分的框架。还介绍了一些关系数据库的概念，作为典型计算机应用的例子。同时，还假设学生对文字编辑器或文字处理器有一定经验。

**HOL6层** HOL6层也是一章，复习了C++编程语言。本章假定学生具有某种命令语言的经验，不一定是C++，可以是Java或C。书中避免了C++的高级特性，包括面向对象的概念。如果有必要，教师可以把C++例子翻译成其他HOL6层的语言。

本章着重介绍了C++内存模型，包括全局变量和局部变量、函数参数以及动态分配的变量。也介绍了递归的问题，因为它依赖于运行时栈上的内存分配机制。还相当详细地解释了函数调用的内存分配过程，因为本书后面还会在较低抽象层次上分析这个机制。

**ISA3层** ISA3层是指令集架构层，包括洞察。描述了一个用于说明计算机概念的虚构的Pep/8计算机。Pep/8是经典的冯·诺依曼机器。CPU包含一个累加器、一个地址寄存器、一个程序计数器、一个操作指针和一个指令寄存器。有8种寻址方式：立即数、直接、间接、栈相对、块相对而读、变址、块变址和栈变址间接。在模拟的只读存储器(ROM)中，Pep/8操作系统能从学生的文本文件中录入和执行十六进制格式的程序。学生可以在Pep/8

## 译者序

Computer Systems, Fourth Edition

本书第1版于2010年出版。在不断感叹计算机技术发展太快的时候，2010年的内容会不会太“老”太“旧”了呢？答案显然是否定的，尤其在读完了整本书之后。本书展示了计算机系统的7个抽象层次：应用层、高级语言层、汇编层、操作系统层、指令集架构层、微代码层和逻辑门层。本书的特色之一就是着眼于计算机软件和硬件系统背后的设计原理，而这些原理数年来都未曾改变过。而且，去除那些眼花缭乱的新技术的表象，能够更好地看清和理解系统的本质。

本书的另一个特色是建立了一个虚构的计算机系统Pep/8，借助于这个示例系统，能够让读者/学生更具体地了解计算机系统中的各个组成部分，进而了解核心概念，而不是通过抽象的描述学习抽象的概念；同时又不必拘泥于现实系统的实现细节。本书覆盖广泛，但又重点突出，强调了硬件及其相关软件的实现。本书文字简洁明了，是非常合适的计算机系统入门教材。

与所有计算机书籍的翻译一样，翻译过程中充满了艰难的术语选择，因为我们越来越习惯于在日常技术工作中使用英文术语，有时候使用它的中文翻译反而显得有些陌生和别扭。比如token，中文一般译作“语言符号”，但是实际上它有终结符字符串的含义，如果翻译成中文，很容易失去这些意义。所以我选择保留英文术语，相信不会影响读者阅读或增加阅读难度。

在此感谢王文杰帮助我一起讨论翻译中遇到的问题，还要感谢机械工业出版社华章公司的编辑们给了我很多理解和支持，使得本书得以完成。

在翻译过程中，我尽量做到认真仔细，但还是难以避免出现错误和不尽如人意的地方。在此欢迎广大读者批评指正，我也会把勘误表及时在网上更新，便于大家阅读。

龚奕利

2015年5月于安娜堡

# 前言

在课程中的使用。本书是第四版，与前一版相比，内容更新，但结构和组织方式保持不变。本书的目的是帮助学生理解计算机系统的组成和工作原理，同时提供一些实际的应用示例。通过阅读本书，读者将能够掌握计算机系统的基本概念，并能够将其应用于实际的项目中。

本书清晰、详细、循序渐进地展示了计算机组成、汇编语言和计算机体系结构中的核心思想。本书的很大一部分是建立在一个虚构的计算机 Pep/8 基础上的，用它来讲解经典的冯·诺依曼机器的基本概念。这种方法的好处是能够讲解计算机科学的核心概念，而又不必拘泥于此类课程中常见的许多不相关的细节。这种方法还能鼓励学生思考计算机科学底层的原理。本书的范围也比较广泛，重点强调了与硬件及其相关软件的处理有关而少有提及的计算机科学主题。

## 内容摘要

计算机运行在一些抽象层上，在高级抽象层上编程只是一部分。基于图 1 的层次结构，本书展示了计算机系统的一个统一的概念。

对应于图 1 的 7 层，本书也分为 7 个部分：

- App7 层 应用层
- HOL6 层 高级语言层
- ISA3 层 指令集架构层
- Asmb5 层 汇编层
- OS4 层 操作系统层
- LG1 层 逻辑门层
- Mc2 层 微代码层

本书主要是按照从上到下、从最高层到最低层的顺序来书写。ISA3 层在 Asmb5 层之前以及 LG1 层在 Mc2 层之前讲解是出于教学的目的。在这两种情况下，暂时用相反的从下至上的方法来讲解更自然，有了低层的构造模块就很容易完成上层的构建。

**App7 层** App7 层是单独一章，介绍了应用程序。本章展示了抽象层次的概念，建立本书剩下部分的框架。还介绍了一些关系数据库的概念，作为典型计算机应用的例子。同时，还假设学生对文字编辑器或文字处理器有一定的经验。

**HOL6 层** HOL6 层也是一章，复习了 C++ 编程语言。本章假设学生具有某种命令语言的经验，不一定是 C++，可以是 Java 或 C。书中避免了 C++ 的高级特性，包括面向对象的概念。如果有必要，教师可以把 C++ 例子翻译成其他 HOL6 层的语言。

本章着重介绍了 C++ 内存模型，包括全局变量和局部变量、函数参数以及动态分配的变量。也介绍了递归的问题，因为它依赖于运行时栈上的内存分配机制。还相当详细地解释了函数调用的内存分配过程，因为本书后面还会在较低抽象层次上分析这个机制。

**ISA3 层** ISA3 层是指令集架构层，包括两章，描述了一个用于说明计算机概念的虚构的 Pep/8 计算机。Pep/8 是经典的冯·诺依曼机器。CPU 包含一个累加器、一个变址寄存器、一个程序计数器、一个栈指针和一个指令寄存器。有 8 种寻址方式：立即数、直接、间接、栈相对、栈相对间接、变址、变址和栈变址间接。在模拟的只读存储器 (ROM) 中，Pep/8 操作系统能从学生的文本文件中装入和执行十六进制格式的程序。学生可以在 Pep/8

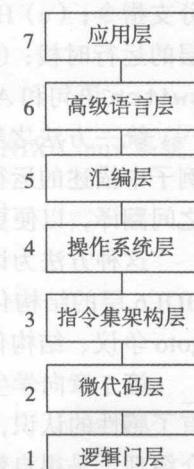


图 1 典型计算机系统的层次结构

模拟器上运行小程序，学习不会改变内存值的 ROM 存储指令。

学生能学习到位层的信息表示和计算机组成的知识。因为本书的中心主题是层次之间的关系，所以有关 Pep/8 的章节展示了 ASCII 表示（ISA3 层）和类型为 `char` 的 C++ 变量（HOL6 层）之间的关系。还展示了补码表示（ISA3 层）和类型为 `int` 的 C++ 变量（HOL6）之间的关系。

**Asmb5 层** Asmb5 层是汇编层，书中介绍了汇编器的概念（汇编器是汇编层和机器层之间的翻译器），还介绍了 Asmb5 层的符号和符号表。

这里是统一的方法派上用场的地方。第 5 章和第 6 章中的编译器是高级语言到汇编语言的翻译器。前面，学生学习了一种具体的 HOL6 层语言 C++ 和一种具体的冯·诺依曼机器 Pep/8。接下来的章节将继续介绍层次之间的关系，讲述下面这样一些对应关系：(a) HOL6 层的赋值语句和 Asmb5 层的装入 / 存储指令；(b) HOL6 层的循环和 `if` 语句与 Asmb5 层的分支指令；(c) HOL6 层的数组和 Asmb5 层的变址寻址；(d) HOL6 层的过程调用和 Asmb5 层的运行时栈；(e) HOL6 层的函数和过程参数与 Asmb5 层的栈相对寻址；(f) HOL6 层的 `switch` 语句和 Asmb5 层的转移表；(g) HOL6 层的指针和 Asmb5 层的地址。

统一方法之美就在于可以在较低层次上实现 C++ 章节中的例子。例如，第 2 章的递归例子中描述的运行时栈直接对应于 Pep/8 主存中的硬件栈。学生可以用手动方式直接在两层之间翻译，以便更好地理解编译的过程。

这种方法为讨论计算机科学中的核心问题提供了一种很自然的环境。例如，本书介绍了 HOL6 层的结构化编程，可以和 Asmb5 层的非结构化编程的可能性进行对比。书中讨论了 `goto` 争议、结构化编程 / 效率之间的折中，给出了两个层次上语言的实际例子。

第 7 章向学生介绍了计算机科学理论。既然学生对如何把高级语言翻译成汇编语言已经有了感性的认识，那么我们就提出所有计算中最基本的问题：什么是能够被自动化的？这里介绍理论是很自然的，因为学生现在知道了编译器（自动化翻译器）必须做什么。他们通过识别 C++ 和 Pep/8 汇编语言的语言符号来学习语法分析和有限状态机——确定性的和非确定性的。本章包括两种小语言之间的自动翻译器，说明了词法分析、语法分析和代码生成。词法分析器是有限状态机的实现。还有什么比这样更自然的介绍理论的方法呢？

**OS4 层** OS4 层讲述操作系统，分为两章。第 8 章讲述进程管理，包括两节，一节讲装载器，一节讲陷阱处理程序，说明了 Pep/8 操作系统的概念。有 5 条指令具有产生软件陷阱的未实现操作码。操作系统把用户正在运行的进程的进程控制块存储在系统栈上，中断服务例程解释该指令。通过具体实现一个挂起进程来强化操作系统中运行和等待进程的经典状态转移图。本章结尾描述了并发进程和死锁。第 9 章描述存储管理，包括主存和磁盘存储器。

**LG1 层** LG1 层用两章来介绍组合电路和时序电路。从布尔代数的定理开始，第 10 章重点介绍计算机科学的数学基础的重要性，展示布尔代数和逻辑门之间的关系，然后介绍一些常见的 SSI 和 MSI 逻辑设备，包括 Pep/8 ALU 的完整的逻辑设计。第 11 章通过介绍时序电路的状态转移图，描述有限状态机的基本概念。最后描述常见的计算机子系统，比如双向总线、内存芯片和双端口存储器体。

**Mc2 层** 第 12 章描述 Pep/8 CPU 的微程序设计控制区，给出了一些示例指令和寻址方式的控制序列，还提供了有关其他指令和寻址方式的大量练习。本章还介绍了装入 / 存储体系结构的概念，对比了 MIPS 的 RISC 机器和 Pep/8 CISC 机器。最后描述了高速缓存、流水线、动态分支预测和超标量机器，介绍了一些性能问题。

## 在课程中的使用

本书覆盖的内容广泛，教师在设计课程时可以省略一些内容。第 1~5 章可以看作核心，第 6~12 章可以有所取舍。

本书第 1~5 章必须顺序讲授，第 6 章和第 7 章可以按任意顺序讲授。我通常会省略第 6 章而直接讲第 7 章，开始一个大的软件项目——为 Pep/8 汇编语言的一个子集写一个汇编器，这样学生在一学期中有足够的时间完成它。第 11 章显然依赖于第 10 章，但是它们都不依赖于第 9 章，所以第 9 章可以省略。图 2 是一个章节依赖图，图中总结了可以省略哪些章节。

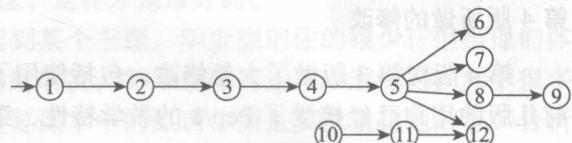


图 2 章节依赖图

## 辅助资料

下面列出的辅助资料可以从出版社网站获得：

<http://www.jbpub.com/catalog/9780763771447/>

**Pep/8 汇编器和模拟器** Pep/8 机器在 MS Windows、Mac OS X 和 UNIX/Linux 系统上都有。汇编器的特性包括：

- 集成的文字编辑器；
- 在源代码中发现错误的地方插入红色字体的错误消息；
- 对学生友好的、十六进制格式的机器语言目标代码；
- 能够直接以机器语言编写代码，跳过汇编器；
- 能够重定义触发同步陷阱的未实现操作码的助记符。

模拟器的特性包括：

- 模拟的 ROM，装入指令不能修改；
- 在模拟的 ROM 中烧入了一个小的操作系统，包括一个装载器和一个陷阱处理程序；
- 一个集成的调试器，允许设置断点、单步执行、CPU 跟踪和内存跟踪；
- 用户定义的、从无限循环恢复的语句执行计数的上限；
- 能够通过为未实现操作码设计新的陷阱处理程序来修改操作系统。

**Pep/8 CPU 模拟器** CPU 模拟器，有 MS Windows、Mac OS X 和 UNIX/Linux 系统版本，可以用在计算机组成课程中。CPU 模拟器的特性包括：

- 颜色编码的展示通路，根据控制信号跟踪复用器的数据流；
- 操作的单周期模式，用 GUI 输入每个控制信号，立即展示信号的效果；
- 操作的多周期模式，学生可以在集成的文字编辑器中编写 Mc2 微代码序列并执行它们以便实现 ISA3 指令。

**课程课件** 每章有 50~125 页的课程幻灯片，有 Keynote 和 PDF 格式。幻灯片包括课本中所有的图和总结信息，通常以标号的形式给出。不过其中没有太多的例子，给教师展示示例和教师指导讨论留出了空间。

**考试题目** 提供有一组考试题目，包括参考信息，例如 ASCII 表、指令集表等，供考试和自学之用。这些对用本书作为教材的教师开放。

**数字电路实验** 有一组 6 个数字电路实验，能够让学生在物理实验电路板上亲身体验。这些实验说明了第 10 章和第 11 章的组合和时序设备，使用许多本书中没讲到的电路。学生

可以自学实际的电子设计和实现概念，这些超出了本书的讲述范围，它们可以按照书中讨论的主题顺序，从组合电路开始，然后是时序电路和 ALU。

**答案手册** 附录中有部分练习的答案。剩下练习的答案对用本书作为教材的教师开放。出于安全原因，答案直接从出版社获取。相关信息请联系 Jones 和 Bartlett Publishers Representative，电话 1-800-832-0034。

#### 第 4 版所做的修改

第 4 版对第 3 版做了大量修改，包括使用了 Pep/8，它是对 Pep/7 架构的彻底重新设计。前几版的用户已经接受了 Pep/8 的教学特性，第 4 版中还是保留了 Pep/8 架构。本版的每一章都有改进，下面只列出了其中一些主要的：

- 改进了 C++ 回顾——扩展了第 3 版中引入的 C++ 内存模型，更系统地从头开始描述。内存分配图更现实，与主函数一致，显示了对主函数返回地址和返回值的分配。重命名了所有以前命名为 *i* 的变量。当程序翻译成 Pep/8 汇编语言后容易有误会，Pep/8 汇编语言用字母 *i* 表示立即数寻址。
- 改进了字符编码的内容——讲述了 Unicode 字符集，代替了 EBCDIC。
- 跟踪标签——Pep/8 汇编器和模拟器包括一个新的符号跟踪特性，当用户单步跟踪程序时，能够实时显示全局变量和运行时栈。使用这个新特性要求程序员在某些汇编语言语句的注释字段放置跟踪标签，翻译器将忽略这些标签，但是调试器将使用它们。跟踪标签的一个巨大好处是迫使程序员做好文件说明。要使用调试器，学生必须在注释字段精确说明哪些变量要在运行时栈上分配以及分配的顺序。汇编器验证分配的字节数是否与变量列表要求的字节数匹配。跟踪标签的文件说明优点很大，现在这个版本中描述了跟踪标签语法，书中和答案手册中的每个汇编语言程序中都包括了跟踪标签。
- 改进了语言翻译的内容——前面版本中，第 7 章讲述的语言翻译原理假设学生没有面向对象知识。本版假设学生已经学习过基本的面向对象设计原理，展示的语法分析程序使用了对象组合、继承和多态调度以及 UML 图。
- 新的项目问题——这一版本有两个项目问题，一个是新的，在第 6 章中，要求写一个 Pep/8 机器模拟器；另一个是改进的，在第 7 章中，要求写一个 Pep/8 汇编器。这两个项目要求开发上千行代码的程序，它们都有多个部分，每一个都往应用程序中增加了更多的功能。项目的目的有两个：1) 让学生获得编写较复杂程序的经验；2) 增强对这门课程问题域中计算机系统概念的理解。
- 改进了 RAID 的内容——这个版本介绍了更广泛的 RAID 磁盘系统内容，扩展了 RAID 等级 01 和 10 的区别，增加了新的图和新的量化分析习题。
- 改进了 MIPS 的内容——扩展了 MIPS 的内容，更系统地比较了 CISC 架构的 Pep/8 和 RISC 架构的 MIPS。新的 MIPS 章节用新的指令集表描述了 5 种寻址方式。MIPS 机器的数据区的图包括了伪直接寻址方式所要求的数据通路和复用器。明确命名的控制信号使用与 Pep/8 控制信号相同的语法，提供了 MIPS 指令实现更简洁而详细的描述。

#### 独特的特性

本书有几个独特的特性，使之有别于其他计算机系统、汇编语言和计算机组成的书。

- 概念的方法——许多教科书试图跟上领域的变化，包括最新的技术发展。例如，最新外围设备的通信协议规范。这类书通常通篇是“设备如何工作”的描述性解释。本书避开了这类资料，而只选择基础的计算概念，掌握了这些就有了理解当前和未来技术的基础。例如，以数字电路设计问题来说，让学生掌握空间 / 时间折中的概念比简单地阅读通用描述更重要。再举一个例子，通过学习如何在 ISA 指令的微代码实现中合并周期来掌握硬件并行的概念，这样才是最好的。
  - 强调问题解决——如果只听说或者读到某个主题，学生能记住的很少；但当他们体验到时，才会记住很多。本书强调问题解决，每章后面有近 400 道练习，其中很多有多个部分。这些练习不会让学生重复课本中的原话，而是要求量化地解答、分析或者设计系统某个抽象层次上的一个程序或电子电路。
  - 一致的机器模型——Pep/8 机器是一个小型的 CISC 计算机，是描述系统所有层次的载体。学生可以清晰地看到抽象层次之间的关系，因为他们要在所有的层次上为这个机器编程或者设计电子电路。例如，当在 LG1 层设计 ALU 组件时，他们知道 ALU 在 ISA3 层的实现中应该在哪个位置。通过像编译器那样把 C++ 程序翻译成汇编语言，他们学到优化编译器和非优化编译器之间的差别。在不同层次上都使用同样的机器模型做工作在效率上有很大的优势，因为模型从上至下都是一致的。不过本书也讲述了 MIPS 机器，对比了 RISC 设计原理和微程序设计的 CISC 设计。
  - 完整的程序示例——许多计算机组成和汇编语言的书会受到代码片段综合征的影响。Pep/8 的内存模型、寻址方式和输入 / 输出特性使得学生能写出完整的程序，容易执行和测试，而不只是代码片段。真实的机器，特别是 RISC 机器，有复杂的函数调用协议，涉及寄存器分配、寄存器溢出和内存对齐限制之类的问题。Pep/8 是少数几种教学机之一（有可能是唯一一个），允许学生书写具有输入 / 输出的完整程序，可以使用全局变量和局部变量、全局数组和局部数组、传值调用和传引用调用、数组参数、使用转移表的开关语句、递归、使用指针的链式结构和堆。写完整程序的作业进一步实现了通过动手来学习的目标，而不是通过读代码片段来学习。
  - 理论和实践的结合——有些读者注意到了，讲述语言翻译原理的第 7 章在计算机系统书中不常见。这种现象可悲地说明了计算机科学课程体系甚至计算机科学领域中理论和实践之间的鸿沟。因为本书讲述了 HOL6 层的 C++ 语言、Asmb5 层的汇编语言和 ISA3 层的机器语言，而且都有一个目标，即理解层次之间的关系，一个更好的问题是：“为什么不能包括讲述语言翻译原理的一章呢？”本书尽可能地加入理论以支持实践。例如，介绍布尔代数作为一个公理系统，配合练习来证明定理。
  - 宽度和深度——第 1 ~ 6 章中的内容对计算机系统或汇编语言编程的书来说是很典型的，第 8 ~ 12 章对计算机组成的书来说是很典型的。在一本书中包括这么广泛的内容是很独特的，而且它还在一个完整机器的各个抽象层次上使用一个一致的机器模型。数字电路 LG1 层内容的深度也是很特别的，它使得 CPU 的组成部分不再神秘。例如，本书描述了 Pep/8 CPU 的复用器、加法器、ALU、寄存器、内存子系统和双向总线的实现。学生学习逻辑门层的实现，没有概念上的空洞，而如果只是泛泛地描述，就只能选择相信而不能完全理解。
- 本书回答了这个问题：“汇编语言编程和计算机组成在计算机科学课程体系中的位置是什么？”它提供了对无处不在的冯·诺依曼机器架构的深入理解。本书的目标是提供本领域

内所有主要知识域的综合概述，包括软件和硬件的结合，理论和实践的结合。

## 计算机课程体系 2001

ACM 和 IEEE 计算机学会建立了计算机科学的“课程体系 2001”指导原则。该指导原则给出了知识体的分类和具体核心。本书适用于体系结构和组成 (Architecture and Organization, AR) 类别，几乎包含 AR 知识体中所有的核心主题。初期报告中的 AR 核心域以及本书中覆盖这些域的章节如下所示：

- AR1. 数字逻辑和电子系统，第 10、11、12 章。
- AR2. 数据的机器级表示，第 3 章。
- AR3. 汇编层机器组成，第 4、5、6 章。
- AR4. 内存系统组成和体系结构，第 9、11 章。
- AR5. 接口与通信，第 8、9 章。
- AR6. 功能组成，第 11、12 章。
- AR7. 多处理和其他体系结构，第 8 章。

## 致谢

Pep/1 有 16 条指令、一个累加器和一种寻址方式。Pep/2 增加了变址寻址。John Vannoy 用 ALGOL W 语言写了 2 个模拟器。Pep/3 有 32 条指令，用 Pascal 语言编写，是学生软件项目，由 Steve Dimse、Russ Hughes、Kazuo Ishikawa、Nancy Brunet 和 Yvonne Smith 完成。Harold Stone 在早期审阅中提出许多对 Pep/3 架构的改进意见，后来被加到 Pep/4 中，并延续到后续的机器中。Pep/4 有特殊的栈指令，模拟 ROM 和软件陷阱。Pep/5 有更正交的设计，允许任何指令使用任何寻址方式。John Rooker 写了 Pep/4 系统和早期的 Pep/5 版本。Gerry St. Romain 实现了一个 MacOS 版本和一个 MS-DOS 版本。Pep/6 简化了变址寻址方式，也包括了一组完整的条件分支指令。John Webb 用 BlackBox 开发系统编写了跟踪功能。Pep/7 把安装的内存从 4 MB 增加到了 32 MB。Pep/8 把寻址方式的数量从 4 增加到 8，安装的内存增加到 64MB。Pep/8 汇编器和模拟器的 GUI 版本由一组学生用 Qt 开发系统和 C++ 实现和维护，小组成员包括 Deacon Bradley、Jeff Cook、Nathan Counts、Stuart Fox、Dave Grue、Justin Haight、Paul Harvey、Hermi Heimgartner、Matt Highfield、Trent Kyono、Malcolm Lipscomb、Brady Lockhart、Adrian Lomas、Ryan Okelberry、Thomas Rampelberg、Mike Spandrio、Jack Thomason、Daniel Walton、Di Wang、Peter Warford 和 Matt Wells。Ryan Okelberry 也参与编写了 Pep/8 CPU 模拟器。Luciano d'Ilori 编写了汇编器的命令行版本。

Tanenbaum 的《 Structured Computer Organization 》比其他任何一本书都更大程度地影响了本书的编写。本书扩展了 Tanenbaum 书的层次结构，在上面增加了高级语言层和应用层。

下面这些书稿审阅者和前一版本的用户极大地影响了本版本的终稿，他们是：Wayne P. Bailey、Jim Bilitzki、Fadi Deek、William Decker、Peter Drexel、Gerald S. Eisman、Victoria Evans、David Garnick、Ephraim P. Glinert、Dave Hanscom、Michael Hennessy、Michael Johnson、Andrew Malton、Robert Martin、Richard H. Mercer、Randy Molmen、John Motil、Peter Ng、Bernard Nudel、Carolyn Oberlink、Wolfgang Pelz、James F. Peters III、James C. Pleasant、Eleanor Quinlan、Glenn A. Richard、David Rosser、Gerry St. Romain、Harold S.

Stone、J. Peter Weston 和 Norman E. Wright。Joe Piasentin 提供了艺术咨询。有两个人极大地影响了 Pep/8 的设计，一位是 Myers Foreman，有关指令集的很多想法都来自于他；另一位是 Douglas Harms，他提出很多改进意见，其中之一是 MOVSPA 指令，使得可以用传引用方式传递局部变量。

Jones and Bartlett Publishers 的责任编辑 Tim Anderson、产品主管 Amy Rose 和编辑助理 Melissa Potter 提供了宝贵的支持，很高兴与他们一起工作。Kristin Parker 设计的吸引人的封面正符合本书的风格。

我很幸运在一所致力于在本科教学中追求卓越的学校。佩珀代因（Pepperdine）大学的 Ken Perrin，提供了富有创造性的环境和专业的支持，正是在这种环境中，本书得到孕育。我的妻子 Ann 给予我无尽的支持，我要为本书占用的时间向她道歉，并送上我由衷的感谢。

3.3.1 基础运算符	3.3.2 算术表达式	3.3.3 逻辑运算符	3.3.4 循环语句	3.4.1 顺序语句	3.4.2 条件语句	3.4.3 循环语句	3.4.5 序列数据	3.5.1 二进制表示	3.5.2 字码表示	3.5.3 跳转语句	3.5.4 表达式语句	3.5.5 语句块	3.6.1 另一种表示法	3.6.2 另一种表示法	3.6.3 模拟	3.6.4 问题	3.6.5 缓存	3.6.6 缓存	3.6.7 缓存	3.6.8 缓存	3.6.9 缓存	3.6.10 缓存	3.6.11 缓存	3.6.12 缓存	3.6.13 缓存	3.6.14 缓存	3.6.15 缓存	3.6.16 缓存	3.6.17 缓存	3.6.18 缓存	3.6.19 缓存	3.6.20 缓存	3.6.21 缓存	3.6.22 缓存	3.6.23 缓存	3.6.24 缓存	3.6.25 缓存	3.6.26 缓存	3.6.27 缓存	3.6.28 缓存	3.6.29 缓存	3.6.30 缓存	3.6.31 缓存	3.6.32 缓存	3.6.33 缓存	3.6.34 缓存	3.6.35 缓存	3.6.36 缓存	3.6.37 缓存	3.6.38 缓存	3.6.39 缓存	3.6.40 缓存	3.6.41 缓存	3.6.42 缓存	3.6.43 缓存	3.6.44 缓存	3.6.45 缓存	3.6.46 缓存	3.6.47 缓存	3.6.48 缓存	3.6.49 缓存	3.6.50 缓存	3.6.51 缓存	3.6.52 缓存	3.6.53 缓存	3.6.54 缓存	3.6.55 缓存	3.6.56 缓存	3.6.57 缓存	3.6.58 缓存	3.6.59 缓存	3.6.60 缓存	3.6.61 缓存	3.6.62 缓存	3.6.63 缓存	3.6.64 缓存	3.6.65 缓存	3.6.66 缓存	3.6.67 缓存	3.6.68 缓存	3.6.69 缓存	3.6.70 缓存	3.6.71 缓存	3.6.72 缓存	3.6.73 缓存	3.6.74 缓存	3.6.75 缓存	3.6.76 缓存	3.6.77 缓存	3.6.78 缓存	3.6.79 缓存	3.6.80 缓存	3.6.81 缓存	3.6.82 缓存	3.6.83 缓存	3.6.84 缓存	3.6.85 缓存	3.6.86 缓存	3.6.87 缓存	3.6.88 缓存	3.6.89 缓存	3.6.90 缓存	3.6.91 缓存	3.6.92 缓存	3.6.93 缓存	3.6.94 缓存	3.6.95 缓存	3.6.96 缓存	3.6.97 缓存	3.6.98 缓存	3.6.99 缓存	3.6.100 缓存	3.6.101 缓存	3.6.102 缓存	3.6.103 缓存	3.6.104 缓存	3.6.105 缓存	3.6.106 缓存	3.6.107 缓存	3.6.108 缓存	3.6.109 缓存	3.6.110 缓存	3.6.111 缓存	3.6.112 缓存	3.6.113 缓存	3.6.114 缓存	3.6.115 缓存	3.6.116 缓存	3.6.117 缓存	3.6.118 缓存	3.6.119 缓存	3.6.120 缓存	3.6.121 缓存	3.6.122 缓存	3.6.123 缓存	3.6.124 缓存	3.6.125 缓存	3.6.126 缓存	3.6.127 缓存	3.6.128 缓存	3.6.129 缓存	3.6.130 缓存	3.6.131 缓存	3.6.132 缓存	3.6.133 缓存	3.6.134 缓存	3.6.135 缓存	3.6.136 缓存	3.6.137 缓存	3.6.138 缓存	3.6.139 缓存	3.6.140 缓存	3.6.141 缓存	3.6.142 缓存	3.6.143 缓存	3.6.144 缓存	3.6.145 缓存	3.6.146 缓存	3.6.147 缓存	3.6.148 缓存	3.6.149 缓存	3.6.150 缓存	3.6.151 缓存	3.6.152 缓存	3.6.153 缓存	3.6.154 缓存	3.6.155 缓存	3.6.156 缓存	3.6.157 缓存	3.6.158 缓存	3.6.159 缓存	3.6.160 缓存	3.6.161 缓存	3.6.162 缓存	3.6.163 缓存	3.6.164 缓存	3.6.165 缓存	3.6.166 缓存	3.6.167 缓存	3.6.168 缓存	3.6.169 缓存	3.6.170 缓存	3.6.171 缓存	3.6.172 缓存	3.6.173 缓存	3.6.174 缓存	3.6.175 缓存	3.6.176 缓存	3.6.177 缓存	3.6.178 缓存	3.6.179 缓存	3.6.180 缓存	3.6.181 缓存	3.6.182 缓存	3.6.183 缓存	3.6.184 缓存	3.6.185 缓存	3.6.186 缓存	3.6.187 缓存	3.6.188 缓存	3.6.189 缓存	3.6.190 缓存	3.6.191 缓存	3.6.192 缓存	3.6.193 缓存	3.6.194 缓存	3.6.195 缓存	3.6.196 缓存	3.6.197 缓存	3.6.198 缓存	3.6.199 缓存	3.6.200 缓存	3.6.201 缓存	3.6.202 缓存	3.6.203 缓存	3.6.204 缓存	3.6.205 缓存	3.6.206 缓存	3.6.207 缓存	3.6.208 缓存	3.6.209 缓存	3.6.210 缓存	3.6.211 缓存	3.6.212 缓存	3.6.213 缓存	3.6.214 缓存	3.6.215 缓存	3.6.216 缓存	3.6.217 缓存	3.6.218 缓存	3.6.219 缓存	3.6.220 缓存	3.6.221 缓存	3.6.222 缓存	3.6.223 缓存	3.6.224 缓存	3.6.225 缓存	3.6.226 缓存	3.6.227 缓存	3.6.228 缓存	3.6.229 缓存	3.6.230 缓存	3.6.231 缓存	3.6.232 缓存	3.6.233 缓存	3.6.234 缓存	3.6.235 缓存	3.6.236 缓存	3.6.237 缓存	3.6.238 缓存	3.6.239 缓存	3.6.240 缓存	3.6.241 缓存	3.6.242 缓存	3.6.243 缓存	3.6.244 缓存	3.6.245 缓存	3.6.246 缓存	3.6.247 缓存	3.6.248 缓存	3.6.249 缓存	3.6.250 缓存	3.6.251 缓存	3.6.252 缓存	3.6.253 缓存	3.6.254 缓存	3.6.255 缓存	3.6.256 缓存	3.6.257 缓存	3.6.258 缓存	3.6.259 缓存	3.6.260 缓存	3.6.261 缓存	3.6.262 缓存	3.6.263 缓存	3.6.264 缓存	3.6.265 缓存	3.6.266 缓存	3.6.267 缓存	3.6.268 缓存	3.6.269 缓存	3.6.270 缓存	3.6.271 缓存	3.6.272 缓存	3.6.273 缓存	3.6.274 缓存	3.6.275 缓存	3.6.276 缓存	3.6.277 缓存	3.6.278 缓存	3.6.279 缓存	3.6.280 缓存	3.6.281 缓存	3.6.282 缓存	3.6.283 缓存	3.6.284 缓存	3.6.285 缓存	3.6.286 缓存	3.6.287 缓存	3.6.288 缓存	3.6.289 缓存	3.6.290 缓存	3.6.291 缓存	3.6.292 缓存	3.6.293 缓存	3.6.294 缓存	3.6.295 缓存	3.6.296 缓存	3.6.297 缓存	3.6.298 缓存	3.6.299 缓存	3.6.300 缓存	3.6.301 缓存	3.6.302 缓存	3.6.303 缓存	3.6.304 缓存	3.6.305 缓存	3.6.306 缓存	3.6.307 缓存	3.6.308 缓存	3.6.309 缓存	3.6.310 缓存	3.6.311 缓存	3.6.312 缓存	3.6.313 缓存	3.6.314 缓存	3.6.315 缓存	3.6.316 缓存	3.6.317 缓存	3.6.318 缓存	3.6.319 缓存	3.6.320 缓存	3.6.321 缓存	3.6.322 缓存	3.6.323 缓存	3.6.324 缓存	3.6.325 缓存	3.6.326 缓存	3.6.327 缓存	3.6.328 缓存	3.6.329 缓存	3.6.330 缓存	3.6.331 缓存	3.6.332 缓存	3.6.333 缓存	3.6.334 缓存	3.6.335 缓存	3.6.336 缓存	3.6.337 缓存	3.6.338 缓存	3.6.339 缓存	3.6.340 缓存	3.6.341 缓存	3.6.342 缓存	3.6.343 缓存	3.6.344 缓存	3.6.345 缓存	3.6.346 缓存	3.6.347 缓存	3.6.348 缓存	3.6.349 缓存	3.6.350 缓存	3.6.351 缓存	3.6.352 缓存	3.6.353 缓存	3.6.354 缓存	3.6.355 缓存	3.6.356 缓存	3.6.357 缓存	3.6.358 缓存	3.6.359 缓存	3.6.360 缓存	3.6.361 缓存	3.6.362 缓存	3.6.363 缓存	3.6.364 缓存	3.6.365 缓存	3.6.366 缓存	3.6.367 缓存	3.6.368 缓存	3.6.369 缓存	3.6.370 缓存	3.6.371 缓存	3.6.372 缓存	3.6.373 缓存	3.6.374 缓存	3.6.375 缓存	3.6.376 缓存	3.6.377 缓存	3.6.378 缓存	3.6.379 缓存	3.6.380 缓存	3.6.381 缓存	3.6.382 缓存	3.6.383 缓存	3.6.384 缓存	3.6.385 缓存	3.6.386 缓存	3.6.387 缓存	3.6.388 缓存	3.6.389 缓存	3.6.390 缓存	3.6.391 缓存	3.6.392 缓存	3.6.393 缓存	3.6.394 缓存	3.6.395 缓存	3.6.396 缓存	3.6.397 缓存	3.6.398 缓存	3.6.399 缓存	3.6.400 缓存	3.6.401 缓存	3.6.402 缓存	3.6.403 缓存	3.6.404 缓存	3.6.405 缓存	3.6.406 缓存	3.6.407 缓存	3.6.408 缓存	3.6.409 缓存	3.6.410 缓存	3.6.411 缓存	3.6.412 缓存	3.6.413 缓存	3.6.414 缓存	3.6.415 缓存	3.6.416 缓存	3.6.417 缓存	3.6.418 缓存	3.6.419 缓存	3.6.420 缓存	3.6.421 缓存	3.6.422 缓存	3.6.423 缓存	3.6.424 缓存	3.6.425 缓存	3.6.426 缓存	3.6.427 缓存	3.6.428 缓存	3.6.429 缓存	3.6.430 缓存	3.6.431 缓存	3.6.432 缓存	3.6.433 缓存	3.6.434 缓存	3.6.435 缓存	3.6.436 缓存	3.6.437 缓存	3.6.438 缓存	3.6.439 缓存	3.6.440 缓存	3.6.441 缓存	3.6.442 缓存	3.6.443 缓存	3.6.444 缓存	3.6.445 缓存	3.6.446 缓存	3.6.447 缓存	3.6.448 缓存	3.6.449 缓存	3.6.450 缓存	3.6.451 缓存	3.6.452 缓存	3.6.453 缓存	3.6.454 缓存	3.6.455 缓存	3.6.456 缓存	3.6.457 缓存	3.6.458 缓存	3.6.459 缓存	3.6.460 缓存	3.6.461 缓存	3.6.462 缓存	3.6.463 缓存	3.6.464 缓存	3.6.465 缓存	3.6.466 缓存	3.6.467 缓存	3.6.468 缓存	3.6.469 缓存	3.6.470 缓存	3.6.471 缓存	3.6.472 缓存	3.6.473 缓存	3.6.474 缓存	3.6.475 缓存	3.6.476 缓存	3.6.477 缓存	3.6.478 缓存	3.6.479 缓存	3.6.480 缓存	3.6.481 缓存	3.6.482 缓存	3.6.483 缓存	3.6.484 缓存	3.6.485 缓存	3.6.486 缓存	3.6.487 缓存	3.6.488 缓存	3.6.489 缓存	3.6.490 缓存	3.6.491 缓存	3.6.492 缓存	3.6.493 缓存	3.6.494 缓存	3.6.495 缓存	3.6.496 缓存	3.6.497 缓存	3.6.498 缓存	3.6.499 缓存	3.6.500 缓存	3.6.501 缓存	3.6.502 缓存	3.6.503 缓存	3.6.504 缓存	3.6.505 缓存	3.6.506 缓存	3.6.507 缓存	3.6.508 缓存	3.6.509 缓存	3.6.510 缓存	3.6.511 缓存	3.6.512 缓存	3.6.513 缓存	3.6.514 缓存	3.6.515 缓存	3.6.516 缓存	3.6.517 缓存	3.6.518 缓存	3.6.519 缓存	3.6.520 缓存	3.6.521 缓存	3.6.522 缓存	3.6.523 缓存	3.6.524 缓存	3.6.525 缓存	3.6.526 缓存	3.6.527 缓存	3.6.528 缓存	3.6.529 缓存	3.6.530 缓存	3.6.531 缓存	3.6.532 缓存	3.6.533 缓存	3.6.534 缓存	3.6.535 缓存	3.6.536 缓存	3.6.537 缓存	3.6.538 缓存	3.6.539 缓存	3.6.540 缓存	3.6.541 缓存	3.6.542 缓存	3.6.543 缓存	3.6.544 缓存	3.6.545 缓存	3.6.546 缓存	3.6.547 缓存	3.6.548 缓存	3.6.549 缓存	3.6.550 缓存	3.6.551 缓存	3.6.552 缓存	3.6.553 缓存	3.6.554 缓存	3.6.555 缓存	3.6.556 缓存	3.6.557 缓存	3.6.558 缓存	3.6.559 缓存	3.6.560 缓存	3.6.561 缓存	3.6.562 缓存	3.6.563 缓存	3.6.564 缓存	3.6.565 缓存	3.6.566 缓存	3.6.567 缓存	3.6.568 缓存	3.6.569 缓存	3.6.570 缓存	3.6.571 缓存	3.6.572 缓存	3.6.573 缓存	3.6.574 缓存	3.6.575 缓存	3.6.576 缓存	3.6.577 缓存	3.6.578 缓存	3.6.579 缓存	3.6.580 缓存	3.6.581 缓存	3.6.582 缓存	3.6.583 缓存	3.6.584 缓存	3.6.585 缓存	3.6.586 缓存	3.6.587 缓存	3.6.588 缓存	3.6.589 缓存	3.6.590 缓存	3.6.591 缓存	3.6.592 缓存	3.6.593 缓存	3.6.594 缓存	3.6.595 缓存	3.6.596 缓存	3.6.597 缓存	3.6.598 缓存	3.6.599 缓存	3.6.600 缓存	3.6.601 缓存	3.6.602 缓存	3.6.603 缓存	3.6.604 缓存	3.6.605 缓存	3.6.606 缓存	3.6.607 缓存	3.6.608 缓存	3.6.609 缓存	3.6.610 缓存	3.6.611 缓存	3.6.612 缓存	3.6.613 缓存	3.6.614 缓存	3.6.615 缓存	3.6.616 缓存	3.6.617 缓存	3.6.618 缓存	3.6.619 缓存	3.6.620 缓存	3.6.621 缓存	3.6.622 缓存	3.6.623 缓存	3.6.624 缓存	3.6.625 缓存	3.6.626 缓存	3.6.627 缓存	3.6.628 缓存	3.6.629 缓存	3.6.630 缓存	3.6.631 缓存	3.6.632 缓存	3.6.633 缓存	3.6.634 缓存	3.6.635 缓存	3.6.636 缓存	3.6.637 缓存	3.6.638 缓存	3.6.639 缓存	3.6.640 缓存	3.6.641 缓存	3.6.642 缓存	3.6.643 缓存	3.6.644 缓存	3.6.645 缓存	3.6.646 缓存	3.6.647 缓存	3.6.648 缓存	3.6.649 缓存	3.6.650 缓存	3.6.651 缓存	3.6.652 缓存	3.6.653 缓存	3.6.654 缓存	3.6.655 缓存	3.6.656 缓存	3.6.657 缓存	3.6.658 缓存	3.6.659 缓存	3.6.660 缓存	3.6.661 缓存	3.6.662 缓存	3.6.663 缓存	3.6.664 缓存	3.6.665 缓存	3.6.666 缓存	3.6.667 缓存	3.6.668 缓存	3.6.669 缓存	3.6.670 缓存	3.6.671 缓存	3.6.672 缓存	3.6.673 缓存	3.6.674 缓存	3.6.675 缓存	3.6.676 缓存	3.6.677 缓存	3.6.678 缓存	3.6.679 缓存	3.6.680 缓存	3.6.681 缓存	3.6.682 缓存	3.6.683 缓存	3.6.684 缓存	3.6.685 缓存	3.6.686 缓存	3.6.687 缓存	3.6.688 缓存	3.6.689 缓存	3.6.690 缓存	3.6.691 缓存	3.6.692 缓存	3.6.693 缓存	3.6.694 缓存	3.6.695 缓存	3.6.696 缓存	3.6.697 缓存	3.6.698 缓存	3.6.699 缓存	3.6.700 缓存	3.6.701 缓存	3.6.702 缓存	3.6.703 缓存	3.6.704 缓存	3.6.705 缓存	3.6.706 缓存	3.6.707 缓存	3.6.708 缓存	3.6.709 缓存	3.6.710 缓存	3.6.711 缓存	3.6.712 缓存	3.6.713 缓存	3.6.714 缓存	3.6.715 缓存	3.6.716 缓存	3.6.717 缓存	3.6.718 缓存	3.6.719 缓存	3.6.720 缓存	3.6.721 缓存	3.6.722 缓存	3.6.723 缓存	3.6.724 缓存	3.6.725 缓存	3.6.726 缓存	3.6.727 缓存	3.6.728 缓存	3.6.729 缓存	3.6.730 缓存	3.6.731 缓存	3.6.732 缓存	3.6.733 缓存	3.6.734 缓存	3.6.735 缓存	3.6.736 缓存	3.6.737 缓存	3.6.738 缓存	3.6.739 缓存	3.6.740 缓存	3.6.741 缓存	3.6.742 缓存	3.6.743 缓存	3.6.744 缓存	3.6.745 缓存	3.6.746 缓存	3.6.747 缓存	3.6.748 缓存	3.6.749 缓存	3.6.750 缓存	3.6.751 缓存	3.6.752 缓存	3.6.753 缓存	3.6.754 缓存	3.6.755 缓存	3.6.756 缓存	3.6.757 缓存	3.6.758 缓存	3.6.759 缓存	3.6.760 缓存	3.6.761 缓存	3.6.762 缓存	3.6.763 缓存	3.6.764 缓存	3.6.765 缓存	3.6.766 缓存	3.6.767 缓存	3.6.768 缓存	3.6.769 缓存	3.6.770 缓存	3.6.771 缓存	3.6.772 缓存	3.6.773 缓存	3.6.774 缓存	3.6.775 缓存	3.6.776 缓存	3.6.777 缓存	3.6.778 缓存	3.6.779 缓存	3.6.780 缓存	3.6.781 缓存	3.6.782 缓存	3.6.783 缓存	3.6.784 缓存	3.6.785 缓存	3.6.786 缓存	3.6.787 缓存	3.6.788 缓存	3.6.789 缓存	3.6.790 缓存	3.6.791 缓存	3.6.792 缓存	3.6.793 缓存	3.6.794 缓存	3.6.795 缓存	3.6.79
-------------	-------------	-------------	------------	------------	------------	------------	------------	-------------	------------	------------	-------------	-----------	--------------	--------------	----------	----------	----------	----------	----------	----------	----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	--------

# 目 录

Computer Systems, Fourth Edition

出版者的话  
中文版序  
译者序  
前言

## 第一部分 应用层(第7层)

### 第1章 计算机系统 ..... 2

1.1 抽象层次 ..... 2
1.1.1 艺术中的抽象 ..... 3
1.1.2 文档中的抽象 ..... 4
1.1.3 组织中的抽象 ..... 5
1.1.4 机器中的抽象 ..... 6
1.1.5 计算机系统中的抽象 ..... 6
1.2 硬件 ..... 8
1.2.1 输入设备 ..... 9
1.2.2 输出设备 ..... 11
1.2.3 主存储器 ..... 12
1.2.4 中央处理单元 ..... 13
1.3 软件 ..... 13
1.3.1 操作系统 ..... 14
1.3.2 软件分析和设计 ..... 15
1.4 数据库系统 ..... 16
1.4.1 关系 ..... 17
1.4.2 查询 ..... 18
1.4.3 语言结构 ..... 19
总结 ..... 20
练习 ..... 21

## 第二部分 高级语言层(第6层)

第2章 C++ ..... 24
2.1 变量 ..... 24
2.1.1 C++ 编译器 ..... 24
2.1.2 机器无关性 ..... 24
2.1.3 C++ 的内存模型 ..... 25

2.1.4 全局变量和赋值语句 ..... 26
2.1.5 局部变量 ..... 28
2.2 控制流 ..... 29
2.2.1 if/else 语句 ..... 29
2.2.2 switch 语句 ..... 30
2.2.3 while 循环 ..... 30
2.2.4 do 循环 ..... 31
2.2.5 数组和 for 循环 ..... 31
2.3 函数 ..... 32
2.3.1 空函数和传值调用的参数 ..... 32
2.3.2 函数的例子 ..... 33
2.3.3 传引用调用的参数 ..... 34
2.4 递归 ..... 36
2.4.1 阶乘函数 ..... 37
2.4.2 递归的思考方式 ..... 39
2.4.3 递归加法 ..... 40
2.4.4 二项式系数函数 ..... 41
2.4.5 逆转数组元素顺序 ..... 45
2.4.6 汉诺塔 ..... 45
2.4.7 相互递归 ..... 48
2.4.8 递归的成本 ..... 48
2.5 动态内存分配 ..... 49
2.5.1 指针 ..... 49
2.5.2 结构 ..... 50
2.5.3 链式数据结构 ..... 51
总结 ..... 52
练习 ..... 53
问题 ..... 54

## 第三部分 指令集架构层(第3层)

第3章 信息的表示 ..... 58
3.1 无符号二进制表示 ..... 58
3.1.1 二进制存储器 ..... 58
3.1.2 整数 ..... 59
3.1.3 基本转换 ..... 60

3.1.4 无符号整数的范围 .....	61	4.2.2 装入指令 .....	103
3.1.5 无符号加法 .....	62	4.2.3 存储指令 .....	103
3.1.6 进位位 .....	62	4.2.4 加法指令 .....	104
3.2 补码二进制表示 .....	63	4.2.5 减法指令 .....	105
3.2.1 补码的表数范围 .....	65	4.2.6 与和或指令 .....	105
3.2.2 基数转换 .....	66	4.2.7 按位取反和取负指令 .....	106
3.2.3 数轴 .....	66	4.2.8 装入字节和存储字节指令 .....	107
3.2.4 溢出位 .....	68	4.2.9 字符输入和输出指令 .....	108
3.2.5 负数和零位 .....	69	4.3 冯·诺依曼机器 .....	109
3.3 二进制运算 .....	69	4.3.1 冯·诺依曼执行周期 .....	109
3.3.1 逻辑运算符 .....	69	4.3.2 一个字符输出程序 .....	110
3.3.2 寄存器传送语言 .....	70	4.3.3 冯·诺依曼漏洞 .....	113
3.3.3 算术运算符 .....	70	4.3.4 一个字符输入程序 .....	113
3.3.4 循环移位运算符 .....	72	4.3.5 十进制转换为 ASCII .....	113
3.4 十六进制和符号表示 .....	72	4.3.6 一个修改自身的程序 .....	114
3.4.1 十六进制 .....	72	4.4 ISA3 层编程 .....	115
3.4.2 基数转换 .....	73	4.4.1 只读内存 .....	117
3.4.3 字符 .....	75	4.4.2 Pep/8 操作系统 .....	117
3.5 浮点数表示 .....	77	4.4.3 使用 Pep/8 系统 .....	119
3.5.1 二进制小数 .....	77	总结 .....	119
3.5.2 余码表示 .....	78	练习 .....	120
3.5.3 隐藏位 .....	79	问题 .....	121
3.5.4 特殊值 .....	80		
3.5.5 IEEE 754 浮点数标准 .....	83		
3.6 跨层的表示方法 .....	85		
3.6.1 另一种表示 .....	87		
3.6.2 模型 .....	88		
总结 .....	90		
练习 .....	90		
问题 .....	95		
<b>第 4 章 计算机体系结构 .....</b>	<b>97</b>		
4.1 硬件 .....	97		
4.1.1 中央处理单元 .....	98		
4.1.2 主存储器 .....	98		
4.1.3 输入设备 .....	99		
4.1.4 输出设备 .....	99		
4.1.5 数据和控制 .....	100		
4.1.6 指令格式 .....	100		
4.2 直接寻址 .....	102		
4.2.1 停止指令 .....	102		
		<b>第四部分 汇编层 (第 5 层)</b>	
		<b>第 5 章 汇编语言 .....</b>	<b>124</b>
		5.1 汇编程序 .....	124
		5.1.1 指令助记符 .....	124
		5.1.2 伪操作 .....	126
		5.1.3 .ASCII 和 .END 伪操作 .....	126
		5.1.4 汇编器 .....	127
		5.1.5 .BLOCK 伪操作 .....	128
		5.1.6 .WORD 和 .BYTE 伪操作 .....	129
		5.1.7 使用 Pep/8 汇编器 .....	129
		5.1.8 交叉汇编器 .....	130
		5.2 立即数寻址和陷阱指令 .....	131
		5.2.1 立即数寻址 .....	131
		5.2.2 DECI、DECO 和 BR 指令 .....	131
		5.2.3 STRO 指令 .....	133
		5.2.4 解释位模式 .....	134
		5.2.5 反汇编器 .....	135

5.3 符号.....	137	参数 .....	180
5.3.1 带符号的程序 .....	137	6.3.6 用局部变量翻译传引用调用 参数 .....	183
5.3.2 一个冯·诺依曼示例 .....	138	6.3.7 翻译布尔类型 .....	186
5.4 从 HOL6 层翻译 .....	139	6.4 变址寻址和数组 .....	188
5.4.1 cout 语句 .....	139	6.4.1 翻译全局数组 .....	189
5.4.2 变量和类型 .....	140	6.4.2 翻译局部数组 .....	191
5.4.3 全局变量和赋值语句 .....	141	6.4.3 翻译作为参数传递的数组 .....	193
5.4.4 类型兼容 .....	143	6.4.4 翻译 switch 语句 .....	198
5.4.5 Pep/8 符号跟踪器 .....	144	6.5 动态内存分配 .....	200
5.4.6 算术移位和循环移位指令 .....	145	6.5.1 翻译全局指针 .....	200
5.4.7 常量和 .EQUATE .....	147	6.5.2 翻译局部指针 .....	204
5.4.8 指令和数据的放置 .....	149	6.5.3 翻译结构 .....	207
总结.....	149	6.5.4 翻译链式数据结构 .....	210
练习.....	150	总结 .....	214
问题.....	152	练习 .....	214
<b>第 6 章 编译到汇编层 .....</b>	<b>155</b>	问题 .....	215
6.1 栈寻址和局部变量 .....	155	<b>第 7 章 语言翻译原理 .....</b>	<b>222</b>
6.1.1 栈相对寻址 .....	155	7.1 语言、语法和语法分析 .....	222
6.1.2 访问运行时栈 .....	156	7.1.1 连接 .....	223
6.1.3 局部变量 .....	158	7.1.2 语言 .....	223
6.2 转移指令和控制流 .....	159	7.1.3 语法 .....	224
6.2.1 翻译 if 语句 .....	160	7.1.4 C++ 标识符的语法 .....	225
6.2.2 优化编译器 .....	161	7.1.5 有符号整数的语法 .....	226
6.2.3 翻译 if/else 语句 .....	162	7.1.6 上下文相关的语法 .....	227
6.2.4 翻译 while 循环 .....	163	7.1.7 语法分析问题 .....	227
6.2.5 翻译 do 循环 .....	164	7.1.8 表达式的语法 .....	228
6.2.6 翻译 for 循环 .....	165	7.1.9 C++ 语法的一部分 .....	229
6.2.7 面条代码 .....	166	7.1.10 C++ 的上下文相关性 .....	232
6.2.8 早期语言中的控制流 .....	168	7.2 有限状态机 .....	233
6.2.9 结构化编程定律 .....	169	7.2.1 用 FSM 来分析标识符 .....	233
6.2.10 goto 争论 .....	169	7.2.2 简化的有限状态机 .....	234
6.3 函数调用和参数 .....	171	7.2.3 非确定性有限状态机 .....	234
6.3.1 翻译函数调用 .....	171	7.2.4 具有空转移的状态机 .....	235
6.3.2 用全局变量翻译传值调用 参数 .....	173	7.2.5 语言符号识别器 .....	237
6.3.3 用局部变量翻译传值调用 参数 .....	176	7.3 实现有限状态机 .....	239
6.3.4 翻译非空函数调用 .....	178	7.3.1 查找表分析器 .....	240
6.3.5 用全局变量翻译传引用调用		7.3.2 直接编码分析器 .....	241
		7.3.3 输入缓冲区类 .....	244
		7.3.4 多 token 分析器 .....	244