现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作

丛书主编 金兰

COMPUTER ORGANIZATION
PRINCIPLES,ANALYSIS,AND DESIGN
计算机组织：原理、分析与设计

LAN JIN BO HATFIELD

清华大学出版社
Tsinghua University Press

现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作

丛书主编 金兰

COMPUTER ORGANIZATION

PRINCIPLES, ANALYSIS, AND DESIGN

计算机组织：原理、分析与设计

JAN L. HATFIELD

清华大学出版社
Tsinghua University Press

内 容 简 介

本书是大学本科计算机科学和计算机工程专业讲授“计算机组织”课程的教科书或教学参考书。其内容的深度和广度，既适用于大学本科的课堂教学，也适用于计算机专职技术人员阅读参考。

本书内容自成体系，深入浅出地介绍了计算机各主要功能部件的组成原理，通过具体的性能分析，了解其基本设计方法。为了适应广大读者不同的专业背景以及不同专业课程体系对“计算机组织”课程内容的要求，本书在选材上着重考虑从基本内容（数字信息和基本逻辑设计）出发，逐步深入到计算机各主要功能部件以及由它们相互连接组成的数据路径和控制器的设计。

本书区别于其他同类教材的主要特点之一是将同时出版由原作者撰写的英文和中文教材，以便于大学推广双语教学，并将有利于学生在学习本书的同时，提高英文阅读写作能力，同时还可以进一步参考有关的英文文献。

图书在版编目(CIP)数据

计算机组织：原理、分析与设计 = COMPUTER ORGANIZATION：PRINCIPLES，ANALYSIS，AND DESIGN/ LAN JIN，BO HATFIELD 著. —北京：清华大学出版社，2003.12
（现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作）
ISBN 7-302-07719-3

Ⅰ. 计… Ⅱ. ①J… ②B… Ⅲ. ①计算机体系结构－教材－英文 ②电子计算机－设计－教材－英文 Ⅳ. TP30

中国版本图书馆 CIP 数据核字(2003)第 111078 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
责任编辑：谢　琛
封面设计：孟繁聪
版式设计：刘祎淼
印 刷 者：北京市清华园胶印厂
装 订 者：三河市印务有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×230　**印张**：36.75　**字数**：754 千字
版　　次：2004 年 1 月第 1 版　2004 年 1 月第 1 次印刷
书　　号：ISBN 7-302-07719-3/TP·5651
印　　数：1～3200
定　　价：56.00 元

出版说明

为了使我国计算机的教育水平赶上国际步伐，缩小与世界计算机技术水平的差距，清华大学出版社隆重推出“现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作”。

这套教材由我国著名计算机专家金兰教授主编。金兰教授1949年毕业于清华大学电机系，后留校任教，1952年赴前苏联留学，获副博士学位。1956年回国后，金兰教授主持清华大学计算机专业的创建工作，先后担任计算机教研室主任和副系主任。1984年后，金兰教授在美国从事计算机专业教育工作，目前是美国Fresno加州大学计算机科学系终身教授。

本套教材的作者均是目前在美国等计算机发展水平较高的国家担任大学教授的专家学者，他们在世界各知名大学担任主讲教授，并且，他们也将在自己任教的学校使用这些教材。我们希望这套教材的出版能使我国的计算机教育尽快与世界计算机教育接轨。英文教材的编写将尽量考虑中国国情，使之适合中国学生学习。在出版英文教材的同时，还将组织与之相应的中文翻译版教材的出版，使那些学习中文版教材的学生也能赶上国际计算机教学的水平。

我们尝试用这种聘请国外教授编著教材的方法出版教材，旨在希望这套教材能够加速我国计算机教育水平的发展，缩小与发达国家的差距。同时，也能提高我国计算机教材的出版水平，为我国计算机教材注入更新的活力。希望这套教材能够得到我国计算机教育领域的广大师生的关注，并提供宝贵的建议。

联系方式：E-mail：xiech@tup.tsinghua.edu.cn
联系人：谢琛

清华大学出版社
2003年8月

作者介绍

金兰博士 在中国清华大学获得电机工程学士学位和在前苏联莫斯科动力学院获得电机工程副博士学位。自1957年开始他担任中国清华大学计算机科学与工程系副教授、教授。1984—1987年他任美国麻省理工学院(MIT)和宾州州立大学(PSU)的教授。自1989年至今他担任美国Fresno加州州立大学计算机科学教授。在这些大学中,他为大学本科生和研究生讲授多门计算机专业课程。他自1978年以来从事高等计算机结构和并行处理的教学科研工作,并在1982—1986年间出版"并行处理计算机结构"和"计算机组织与结构"两本著作,二者都曾经在大学中作为计算机科学和计算机工程专业教科书。

他目前的研究兴趣是并行和分布处理。

金波博士 在中国西南交通大学获得计算机科学学士学位和在美国宾州州立大学(PSU)获得计算机工程硕士和博士学位。1993—1995年她在美国Christopher Newport大学担任计算机科学助理教授。1995—1999年她加入到美国Meredith学院的教授队伍,并有将近一年时间离职在Motorola公司担任高级设计工程师。她也曾在Multilink(现在的Spectel)公司工作,担任高级软件工程师。自2001年至今她在Salem州立学院担任副教授。在前后8年的教学工作中,她曾经讲授多门计算机科学和计算机工程专业课程。

她目前的研究兴趣是数据挖掘,人工神经网络以及并行和分布计算机系统。

About the Authors

Dr. Lan Jin received his B. S. degree in Electrical Engineering from Tsinghua University, China, and a Ph. D. degree in Electrical Engineering from Moscow Electrical Engineering Institute, former USSR. Since 1957 he has been an Associate Professor, and then a Professor in Department of Computer Science and Technology at Tsinghua University, China. From 1984 to 1987 he joined the faculty of Massachusetts Institute of Technology (M. I. T.) and The Pennsylvania State University, USA. Since 1989 till now he has been serving as a Professor of Computer Science at California State University, Fresno. In these universities, he has taught numerous computer science courses to undergraduate and graduate students. He has been engaged in teaching and research in advanced computer architecture and Parallel Processing since 1978 and published two books, "Parallel Processing Computer Architecture" and "Computer Organization and Architecture", in 1982 to 1986. Both books have been used as textbooks for the majors of Computer Science and Computer Engineering in Chinese universities.

His current interests of research are in parallel and distributed computer systems.

Dr. Bo Hatfield received her B. S. degree in Computer Science from Southwestern Jiaotong University, China, a M. S. degree and a Ph. D. degree in Computer Engineering from The Pennsylvania State University, USA. From 1993 to 1995 she served as an assistant professor of computer science at The Christopher Newport University, USA. She joined the faculty of Meredith College, USA, from 1995 to 1999, during which she left the college to join Motorola Inc. to work as a senior design engineer for almost a year. She also worked as a senior software engineer for Multilink Inc., the current Spectel Inc. Since 2001 till now Dr.

Hatfield has been serving as an Associate Professor at Salem State College, USA. During her eight years of teaching, Dr. Hatfield has taught numerous computer science and computer engineering courses.

Her current interests of research are in data mining, artificial neural networks, and parallel and distributed computer systems.

前言

本书涵盖范围

本书写作目的是为计算机科学和计算机工程专业提供一本可用于“计算机组织”课程的基础教科书。本书内容自成体系，因而只需具备计算机高级语言程序设计的基础知识即可阅读。在选材广度上包括计算机信息和数字逻辑的基础知识，在深度上则为学生进一步学习和将来从事计算机专职工作打下坚实基础。

写作一本“计算机组织”教材的难点之一是关于“计算机组织”和“计算机结构”缺乏被共同接受的定义。通常在科研文献中对这两个名词的了解从概念上有如下区分：计算机结构是指由汇编语言程序设计人员所看到的计算机系统的特性，而计算机组织则是为实现计算机结构而组成的功能部件及其相互连接。我们比较赞同这一理解，从而认为“计算机组织”是一门低层次的偏重硬件的课程，而“计算机结构”则是一门高层次的偏重系统的课程。另一方面，我们理解计算机组织和计算机结构的课程体系又是相互紧密关联的，以至于我们很难把它们明确区分开来。我们宁愿把计算机组织中的“硬件”理解为一种“逻辑”个体而不是一种“物理”个体。

从系统的角度看，计算机系统可以被视为一种多层次结构的抽象体，而这种观点对于理解计算机结构和组织有很重要的影响。有些关于计算机组织的教科书按照计算机层次结构从上而下或从下而上地表达，这种方法特别适合于对只有极少背景知识而缺乏这种层次结构概念的学生讲授。然而，更多的计算机组织教科书是根据计算机的主要功能部件（中央处理器、存储器、输入输出、控制器等）来表述，同时在观念上仍保持计算机系统的全局性层次结构。后一种方法的优点是可以集中叙述计算机各个功能部件的原理和设计，从而避免导致课程涉及面过广的毛病。我们在本书中选择了第二种方法，首先在第1章介绍计算机系统的层次结构，这样使学生知道如何把各章的内容与计算机的全局结构

联系起来。

我们假定计算机层次模型从上而下分为6级——高级语言、操作系统、汇编语言、常规机器(寄存器传送)、微程序以及数字逻辑。如果我们试图区分结构和组织,便可能认为高级语言级和操作系统级较为适合"结构"的范畴。而常规机器级,微程序级和数字逻辑级则较为适合"组织"的范畴,同时汇编语言级则可能位于结构和组织两者之间。

基于这一理解,我们认为下列各课题是位于层次结构的低层,它们都属于本书的范围:

- 汇编语言级(本级的低层讨论由机器指令和数据组成的二进制信息以及对这些数据的各种操作):
  - 数据和信息的机器表示(第2章);
  - 复杂算术操作(第6章);
  - 指令系统结构(第7章)。
- 常规机器或寄存器传送级(汇编级机器组织和功能组织,其中计算机硬件被视为一些信息存储和处理单元,相互连接成数据路径,用于传送各种数据和控制信息):
  - 算术逻辑单元(第5章);
  - 中央处理器(第8章);
  - 控制器(第9章,硬联控制部分);
  - 存储器(第10章,主存储器);
  - 接口和通信(第11章,输入输出)。
- 微程序级(第9章,微程序控制部分)。
- 数字逻辑级(数字逻辑和数字系统,其中上列各功能部件被视为由基本构件组成的逻辑线路和有限状态机):
  - 组合逻辑(第3章);
  - 时序逻辑(第4章)。
- 高等计算机组织(第12章,流水线)。

关于本书涵盖范围的详细情况,请见每一章被标题为Summary的最后一节的内容。

本书的使用

本书是在作者们讲授计算机科学专业课程"计算机组织概论"经验的基础上写成的。一般而言,像我们这样的计算机科学专业,其课程体系中没有一门关于数字逻辑的专门课程,但可能会有一门被称为"计算机系统"的课程。这门课程可能包括某些数字逻辑的概

念,但主要讲授汇编语言程序设计,至少会讲授汇编语言的基本概念作为计算机系统层次的重要一级。因此本书不讲授一种特定的汇编语言并让学生实际练习程序设计,但是我们仍列入一章"指令系统结构",集中讲授与计算机组织紧密相关的一些课题(例如指令形式、变址方式和指令系统设计)。因而本书不强调汇编语言,而强调数字逻辑,并且我们处理数字逻辑的教学材料是从实际的观点出发的,即强调它对计算机线路和部件设计基础的重要性。学生应从介绍数字逻辑的两章(第3章和第4章)学习这些线路和部件的分析和设计方法。如果条件(资源和课程进度计划)允许,他们还可以使用模拟软件来实现和测试这些线路和部件。这样的实际动手经验会给学生奠定扎实的基础,去学习以后各章在计算机功能部件一级进行计算机设计的有关内容。

本书的副标题强调计算机组织的分析与设计。但是,我们并不是企图教会学生设计能作为商品的实际计算机。我们的目的只是为了让学生根据本书所叙述的方法,通过分析和设计加深对计算机组织的了解。因此,本书并不强调商品处理机的典型研究,因为它们过于复杂,可能会妨碍学生学习基本层次设计方法。我们的意图仍然是教会学生设计现代计算机的概念和技术,但我们使用实际计算机的简化版本作为例子,从高层的指令系统一级直到基础层的逻辑线路级和微程序级来说明计算机的数据路径和控制器的设计方法。这样"从动手中学习",学生不但能设计简单的处理机,并且能利用适当的模拟软件在个人计算机上去实现一个"真实的"工作模型。我们的教学经验表明这一学习方法对于计算机科学专业的学生是有效的。他们不需学习额外的硬件课程而学到现代化计算机的内部工作原理。

为了针对不同专业的不同课程体系使用本书,我们按照讲授一个学期计算机组织课程的安排做下列建议:

对于未学过数字逻辑课的学生:

A. 讲授范围可包括除第12章以外的所有各章,但某些讲授专题的节次可以跳过去,例如:逻辑线路的动态特性(3.5节),有限状态机的设计(4.6节),双精度算术运算(6.2节和6.4节),中央处理器位片器件(8.5节),查错和纠错代码(11.6.2节),以及除PCI以外的总线标准(11.7.2~11.7.5节)。

B. 为了能在(15周的)一个学期课程中容纳本书的基本内容,每一章可以着重讲授下列基本选题:二进制补码和操作(2.3.2节和2.4.2节),组合逻辑线路在加法器以及中集成度(MSI)器件上的设计与实现(3.1~3.4节,3.6节),带时钟脉冲的时序逻辑线路在计数器和MSI器件上的设计与实现(4.2~4.5节),算术逻辑单元的设计与实现(5.2节、5.3节、5.4.1节、5.4.2节和5.5.2节),浮点运算(6.5节),指令形式与寻址方式(7.1节和7.2节),基于累加器和通用寄存器组的中央处理器(8.3.1节和8.4节),根据基本指令周期设计操作表和微程序流程图(9.4.1~9.4.3节以及9.5.3~9.5.4节),存

储层次结构和主存储器(10.1 节和 10.2 节),以及输入输出访问和接口(11.3 ~ 11.5 节)。

对于学过数字逻辑的学生:

C. 可以讲授或要求学生自学第 3 章和第 4 章(3.3 ~ 3.5 节和 4.3 ~ 4.5 节)的设计例题以求复习一些选题。这样便可能讲授上述 A 项中列为可选或高等的课题以及第 12 章的流水线。

D. 讲授上述 B 项中列为常规的课题便可以强调设计以及在逻辑线路级上的实现。如果可能,还应配合实验环节。

致 谢

我们谨向在本书写作过程中提供鼓励、支持和帮助的人们表示感谢。我们特别感谢 Christopher W. Hatfield 博士帮助校阅全书原稿,并进行文字加工以求表述更为清晰。我们衷心感谢清华大学出版社及其总编和责任编辑们的远见共识,得以倡导和支持本书的出版。

我们感谢美国 Fresno 加州州立大学计算机科学系系主任 Henderson Yeung 博士对本书写作一贯的支持,还感谢我们学校的学生集体,他们在本书早期版本的多年课堂试教过程中为我们奠定了重要的试验和改进的基础。

由于我们教学工作比较繁忙,写作时间比较仓促,本书的初版无疑会有一些缺点和错误。我们欢迎读者提出宝贵意见和指正书中的错误。

作 者

2003 年 8 月

Preface

The Scope of the Book

We wrote this book with the intention that it would be used in a first-level course on computer organization in computer science and computer engineering curricula. The book is self-contained, so that reading it requires only a basic knowledge of computer programming in a high-level language. The breadth of material has been chosen to provide fundamental knowledge of computer information and basic digital logic. The depth of material has been chosen to provide students with a solid foundation for their future studies and a career as a computer professional.

One of the difficulties of writing a textbook on computer organization is the lack of universally accepted definitions for the terms "computer organization" and "computer architecture". Usually the two terms are distinguished conceptually in the research literature as follows: "computer architecture" refers to those attributes of a system visible to an assembly-language programmer. "Computer organization" refers to the functional units and their interconnections that realize the architecture. We prefer this understanding and consider "computer organization" as a lower-level hardware-oriented course, and "computer architecture" as a higher-level system-oriented course. On the other hand, we understand that the disciplines of computer organization and computer architecture are interrelated so closely that it is difficult to clearly distinguish between them. We would rather treat the "hardware" in computer organization as a "logical" entity rather than a "physical" entity.

From the system point of view, a computer system can be conceived as an abstraction structured as a hierarchy of levels, and this conception has a major impact on the understanding of computer architecture and organization. Some textbooks about computer organization

explicitly recognize the necessity of presenting the teaching material according to this hierarchy of levels, either top-down or bottom-up. This approach is especially suitable for teaching introductory courses to students with minimum background, i. e. , students with no prior exposure of such a concept. However, most computer organization textbooks present the material according to the major functional blocks of a computer (CPU, memory, I/O, control unit, etc.) while still keeping in mind the overall hierarchical structure of a computer system. This latter approach has the advantage of concentrating on the theory and design of separate functional blocks of a computer and avoids creating too broad a scope for the course. Our book chooses the second approach and introduces the hierarchical structure of a computer system in Chapter 1. Thus students will know how to relate the contents of separate chapters to the overall structure of a computer.

We assume a hierarchical model of six levels from top to bottom — higher-level language, operating system, assembly language, conventional machine (or register transfer), microprogramming, and digital logic. If we attempts to differentiate between architecture and organization, then the higher-level language and operating system levels are most suitably addressed in " architecture" , and the conventional machine, microprogramming, and digital-logic levels are better addressed in " organization" , while the assembly language level lies probably between the two.

With this understanding, we consider the following topics at the lower levels of the hierarchical model, all belonging to the scope of the book:

- Assembly language level (The lower end of this level deals with machine instructions and data as binary information and various operations on them) :
  - *Machine-level representation of data and information* (*Chapter* 2).
  - *Complex arithmetic operations* (*Chapter* 6).
  - *Instruction set architecture* (*Chapter* 7).
- Conventional machine or register-transfer level (assembly level machine organization and functional organization, in which computer hardware is considered to be information storage and processing units interconnected in a datapath for the transfer of data and control information) :
  - *Arithmetic-logic unit* (*Chapter* 5).
  - *Central processing unit* (*Chapter* 8).
  - *Control unit* (*Chapter* 9, *part of hardwired control*).
  - *Memory system organization* (*Chapter* 10, *primary memory*).
  - *Interfacing and communication* (*Chapter* 11, *input/output*).

- Microprogramming level (Chapter 9, part of microprogrammed control).
- Digital logic level (digital logic and digital systems, in which the above-listed functional units are considered to be logic circuits and finite-state machines constructed from basic building blocks):
  - *Combinational logic* (*Chapter* 3).
  - *Sequential logic* (*Chapter* 4).
- Advanced computer organization (Chapter 12, pipelining).

The detailed information about the coverage of the book can be found in the last section entitled "summary" of each chapter.

Use of the Book

This book has resulted from our experiences in teaching a Computer Science course entitled "Introduction to Computer Organization". Generally speaking, a Computer Science curriculum such as ours does not have a special course on digital logic, but it may have a course entitled "Computer Systems". Such a course could include some concept of digital logic, but mostly teach assembly language programming or, at least, the basic concept of assembly languages as an important layer of a computer system. Therefore, this book does not advocate a specific assembly language for students to practice programming. However, we do include a chapter entitled "Instruction Set Architecture" which focuses on topics closely related to Computer Organization (such as instruction format, addressing modes, and instruction set design). Thus, we have deemphasized assembly languages and emphasized digital logic in this book. Furthermore, we treat the material on digital logic from the practical point of view, emphasizing its fundamental importance to the design of computer circuits and modules. Students must learn the analysis and design of these circuits and modules in the two introductory chapters on digital logic (Chapters 3 and 4). If condition (resources and class schedule) permits, they can even implement and test these circuits and modules using simulation software. This hands-on experience would give students a sound basis for learning the material in later chapters related to the design of computers at the level of functional blocks.

This book's subtitle emphasizes analysis and design in computer organization. However, we do not attempt to teach students to design a practical computer as a commercial product. Our goal is to strengthen the students' understanding of the computer organization by way of analysis and design according to the methods taught in this book. Therefore, this book does

not emphasize case studies of commercially-available processors, because their complexity would prevent students from learning the basic-level design methods. Our intend is to teach students the concepts and techniques of designing modern computers. We use a simplified version of a practical processor as an example, on which we can show the design methods of its datapath and control unit from the upper level of instruction set down to the basic level of logic circuits or microprogramming. In "learning by doing", students can not only design a simple processor, but also implement a "real" working model on a PC using an appropriate simulation software. Our teaching experience has demonstrated the effectiveness of this teaching method for Computer Science students, who can learn the inner workings of a modern computer without learning additional hardware courses.

To use this book in the different curricula of different majors, we make the following recommendations for teaching a one-semester course on computer organization:

For students who have not taken a course on digital logic:

A. All chapters except Chapter 12 can be covered, but some sections teaching special topics could be skipped — the dynamic characteristics of logic circuits (Section 3.5), the design of finite-state machine (Section 4.6), double-precision arithmetic (Sections 6.2 and 6.4), CPU bit-slice devices (Section 8.5), error detection and correction codes (Section 11.6.2), and bus standards except PCI (Sections 11.7.2 to 11.7.5).

B. To accommodate the basic contents of the book in a semester, each chapter can be taught with emphasis on selected basic topics such as the following: two's complement representation and operations (Sections 2.3.2 and 2.4.2), combinational circuit design and implementation on adders and MSIs (Sections 3.1-3.4 and 3.6), clocked sequential circuit design and implementation on counters and MSIs (Sections 4.2-4.5), ALU design and implementation (Sections 5.2, 5.3, 5.4.1, 5.4.2 and 5.5.2), floating-point operations (Section 6.5), instruction format and addressing modes (Sections 7.1 and 7.2), CPU based on accumulator and general-purpose registers (Sections 8.3.1 and 8.4), the design of the operation chart and the microprogram flowchart according to the basic instruction cycles (Sections 9.4.1-9.4.3 and 9.5.3-9.5.4), memory hierarchy and main memory (Sections 10.1 and 10.2), and, finally, I/O accessing and I/O interfaces (Sections 11.3-11.5).

For students who have taken a course on digital logic:

C. Students can be taught (or asked to read on their own) the design examples from Chapters 3 and 4 (Sections 3.3-3.5 and Sections 4.3-4.5) to review selected topics. This

gives an possibility to teach the optional or advanced topics that are listed above in A, as well as pipelining in Chapter 12.

D. The teaching of the regular topics listed above in B can be taught with more emphasis on design and implementation down to the level of logic circuits. If possible, laboratory activities should be incorporated.

Acknowledgments

We wish to express our thanks to many people who have provided encouragement, support, and help during the writing of this book. We would especially like to thank Dr. Christopher W. Hatfield for editing the text of the entire manuscript and consistently enhancing the clarity of the presentation of the material. We gratefully acknowledge Tsinghua University Press with its chief editors and acquisitions editor for their foresight and vision to initiate and guide the publishing of this book.

We wish to thank Dr. Henderson Yeung, the chair of Department of Computer Science at California State University, Fresno, U. S. A., for his constant support to the writing of this book. We also thank our student populations who provided important proving and enhancing grounds for the material of this book during many years of class-testing of its earlier versions.

Since we have been busy in our regular teaching activities and had limited time in writing the book, this first edition of the book undoubtedly contains drawbacks and errors. Any comments and report of errors will be appreciated.

Contents

CHAPTER 1 INTRODUCTION

CHAPTER 2 THE REPRESENTATION OF INFORMATION IN A COMPUTER