Information Technology English

信息技术专业英语教程

主编◎杨清波　张德盛　魏晓芹

科学出版社

信息技术专业英语教程

Information Technology English

主　编：杨清波　张德盛　魏晓芹
副主编：吕高焕　张利锋　贾世祥
编　委：陈　琳　郭　坤　林明东
徐明铭　韩婷婷　杨　坤
杨延村　马松梅

科学出版社

北京

内 容 简 介

本教材在编写过程中，充分考虑信息与电气工程专业特点，紧扣专业发展趋势，因地制宜、因人制宜，既注重专业知识，也强调英语语言技能的训练；既强调专业能力的培养，也注重与专业相关的人文知识的学习。其目的，一方面是要巩固、提高学生在基础阶段的英语语言知识和技能；另一方面是要在专业领域中进一步实践和应用已经掌握的英语知识，并拓展学生对本专业具有典型意义的人文方面知识的认知，从而达到开阔视野、提高人文修养和利用英语获取信息、输出信息的能力。本教材主要适用于已完成基础阶段学习的高等学校信息与电气工程专业本科生，可以作为应用提高阶段的专业英语学习必修课或选修课教材，也可以作为信电专业研究生教学或信电工程技术人员的外语培训教材。

图书在版编目(CIP)数据

信息技术专业英语教程/杨清波，张德盛，魏晓芹主编. —北京：科学出版社，2015.12

ISBN 978-7-03-046466-8

Ⅰ. ①信… Ⅱ. ①杨… ②张… ③魏… Ⅲ. ①信息技术-英语-教材 Ⅳ. ①H31

中国版本图书馆 CIP 数据核字(2015)第 277601 号

责任编辑：阎 莉 王瑞媛 / 责任校对：杜子昂
责任印制：徐晓晨 / 封面设计：铭轩堂

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京京华虎彩印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2015 年 12 月第 一 版 开本：787×1092 1/16
2015 年 12 月第一次印刷 印张：12 3/4

字数：310 000

定价：48.00 元

(如有印装质量问题，我社负责调换)

前言

大学英语教学大纲（修订本）将大学英语教学分为基础阶段和应用提高阶段两部分，其中应用提高阶段的教学主要以专业英语教学为主。大纲明确规定，大学英语教学的目的是“使学生能用英语交流信息，适应社会发展和经济建设的需要。”编写本教材的目的是为鲁东大学信息与电气工程学院的学生提供一套基于“需求分析”的符合学校、学院具体情况的教材，帮助学生在应用提高阶段进一步发展、巩固和提高他们在基础阶段已经掌握的听、说、读、写、译等英语技能，同时提高学生在专业英语阅读、写作等方面信息获取和信息输出的能力，培养符合社会发展需求的高级应用型专业人才。

本教材由鲁东大学外语教学部英语教师和信息与电气工程学院专业教师合作编写而成。在教材编写的设计中，因地制宜、因人制宜，既注重专业知识，也强调英语语言技能的训练；既强调专业能力的培养，也注重与专业相关的人文知识的学习。一方面，巩固、提高学生在基础阶段的英语语言知识和技能，另一方面，在专业领域中进一步实践和应用已经掌握的英语知识，并拓展学生对本专业具有典型意义的人文方面知识的认知，从而达到开阔视野、提高人文修养和利用英语获取信息、输出信息的能力。

本教材在编写过程中充分考虑信息与电气工程专业特点，紧扣专业发展趋势，同时考虑我国高等教育特色，力争做到具有较高普适性，尽可能满足我国高等院校信电专业教学需要。本教材主要适用于已完成基础阶段学习的高等学校信息与电气工程专业本科生，可以作为应用提高阶段的专业英语学习必修课或选修课教材，也可以作为信电专业研究生教学或信电工程技术人员的外语培训教材。

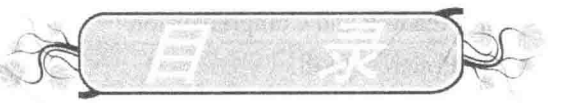
本教材由 10 个单元组成，每个单元由 3 个模块构成。第一模块语料的选取以信电专业典型的代表人物、事件、领域内最新发展为主，旨在提升学生对本专业领域内领军人物、重大事件和学科发展的认识。第二模块语料的选取以信电专业的专业知识为主，突出专业知识的前沿性和代表性。第三模块以英语应用性写作为主，注重应用文写作的实用性和可操作性。三个模块在文章语料的选取上力求新颖、实用，练习的设计尽量做到实用性与多样性结合，学生可以通过各种练习在多个方面提高语言技能。

本教材由鲁东大学大学外语教学部的张德盛、郭坤、林明东、魏晓芹和陈琳老师以及滨州医学院外国语与国际交流学院马松梅老师依次承担本教材 10 个单元的编写，鲁东大学信息与电气工程学院的吕高焕、张利锋、贾世祥、徐明铭、韩婷婷、杨坤和杨延村老师参与了教材材料的遴选工作。鲁东大学大学外语教学部的杨清波副教授负责编写本教材各单元第三模块内容。

教材中若有不妥之处，还望使用者提出宝贵意见。

主　编

2015 年 6 月



I . Reading and Practice

Introducing People of ACM—An Interview with David Patterson

David Andrew Patterson (born on November 16, 1947) is an American computer pioneer and academic who has held the position of Professor of Computer Science at the University of California, Berkeley since 1977. David Patterson is the founding director of the Parallel Computing Laboratory (PAR Lab) at University of California, Berkeley, which addresses the multicore challenge to software and hardware. He founded the Reliable, Adaptive and Distributed Systems Laboratory (RAD Lab), which focuses on dependable computing systems designs. He led the design and implementation of RISC I, likely the first VLSI Reduced Instruction Set Computer.

A former ACM president, Patterson chaired ACM's Special Interest Group in Computer Architecture (SIGARCH), and headed the Computing Research Association (CRA). He is a Fellow of ACM, IEEE, the Computer History Museum, and the American Association for the Advancement of Science, and a member of the National Academy of Engineering and the National Academy of Sciences. He received the Eckert-Mauchly Award from ACM and IEEE-CS, and ACM's Distinguished Service and Karl V. Karlstrom Outstanding Educator Awards. He served on the Information Technology Advisory Committee for the US President (PITAC).

Patterson is a graduate of the University of California at Los Angeles (UCLA), where he earned his A.B., M.S. and Ph.D. degrees. He has consulted for Hewlett Packard, (HP), Digital Equipment (now HP), Intel, Microsoft, and Sun Microsystems, and is on the technical advisory board of several companies.

Patterson is noted for his pioneering contributions to RISC processor design, having coined the term RISC, and by leading the Berkeley RISC project. He is also noted for his research on RAID disks.

His book on computer architecture (co-authored with John L. Hennessy) is widely used in computer science education. Patterson is a Fellow of the American Association for the Advancement of Science.

He is an important proponent of the concept of Reduced Instruction Set Computer and coined the term "RISC". He led the Berkeley RISC project from 1980 and onwards along with Carlo H. Sequin, where the technique of register windows was introduced. He is also one of the innovators of the Redundant Arrays of Independent Disks (RAID) (in collaboration with Randy Katz and Garth Gibson), and Network of Workstations (NOW) (in collaboration with Eric Brewer and David Culler).

He co-authored six books, including two with John L. Hennessy on computer architecture: Computer Architecture: A Quantitative Approach (5 editions—latest is ISBN 0-12-383872-X) and Computer Organization and Design: the Hardware/Software Interface (4 editions—latest is ISBN 0-12-374493-8). They have been widely used as textbooks for graduate and undergraduate courses since 1990.

His work has been recognized by about 35 awards for research, teaching, and service, including Fellow of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) as well as by election to the National Academy of Engineering, National Academy of Sciences, and the Silicon Valley Engineering Hall of Fame.

In the February 28, 2013 second installment of Introducing People of ACM interview, David Patterson, director of the Parallel Computing Lab at UC Berkeley and former ACM president, answers questions, revealing his insight into the pervasive and booming expansion of big data now inherent in the computing technology field.

He describes his successes over 35 years as researcher and professor at Berkeley as the embodiment of projects developed by grad students that would later be adapted into commercial products, most notably Reduced Instruction Set Computers (RISC)

- Redundant Array of Inexpensive Disks (RAID)
- Networks of Workstations (NOW)

In the interview, Patterson discusses how his AMP lab—algorithms, machine, people—will address the expectations of big data analytics in the field of health care, and in particular, cancer research through the intersection of machine learning, cloud computing, and crowd sourcing.

Finally, Patterson advises prospective data analysis technologists to look into the study of statistics and machine learning, as well as taking courses in databases and operating systems, all the while taking part in development of software as a service agile programming languages such as ruby on rails and Django.

As a researcher, professor, and practitioner of computer science, how have these

overlapping roles influenced both your career and the direction of computing technologies?

My research style is to identify critical questions for the IT industry and gather interdisciplinary groups of faculty and graduate students to answer them as part of a five-year project. The answer is typically embodied in demonstration systems that are later mirrored in commercial products. In addition, these projects train students who go on to successful careers.

When I look in the rear view mirror at my 35 years at Berkeley, I see some successes. My best-known projects were all born in Berkeley graduate classes:

• *Reduced Instruction Set Computers* (*RISC*): The R of the ARM processor stands for RISC. ARM is now the standard instruction set of Post PC devices, with nearly 9B ARM chips shipped last year vs. 0.3B x86 chips.

• *Redundant Array of Inexpensive Disks* (*RAID*): Virtually all storage systems offer some version of RAID today; RAID storage is a $25B business today.

Networks of Workstations (*NOW*): NOW showed that Internet services were an excellent match to large sets of inexpensive computers connected over switched local area networks, offering low cost, scalability, and fault isolation. Today, these large clusters are the hardware foundation of search, video, and social networking.

The research shapes the teaching too. The RISC research led to the graduate textbook *Computer Architecture: A Quantitative Approach and the undergraduate* textbook *Computer Organization and Design: The Hardware-Software Interface*, both co-authored with John Hennessy of Stanford University.

How is your AMP Lab involved in addressing the challenges of Big Data research? What can we expect over the next decade in the development of Big Data research and its impact on cancer tumor genomics and other health care issues?

Working at the intersection of three massive trends: powerful machine learning, cloud computing, and crowd sourcing, the AMP Lab integrates Algorithms, Machines, and People to make sense of Big Data. We are creating a new generation of analytics tools to answer deep questions over dirty and heterogeneous data by extending and fusing machine learning, warehouse-scale computing, and human computation.

We validate these ideas on real-world problems, such as cancer genomics. Recently, biologists discovered that cancer is a genetic disease, caused primarily by mutations in our DNA. Changes to the DNA also cause the diversity within a cancer tumor that makes it so hard to eradicate completely. The cost of turning pieces of DNA into digital information has dropped a hundredfold in the last three years. It will soon cost just $1,000 per individual genome, which means we could soon afford to sequence the genomes of the millions of cancer patients.

We need to build fast, efficient software pipelines for genomic analysis to handle the

upcoming tsunami of DNA data that will soon be flowing from these low-cost sequencing machines. Then we need a safe place to store the results. If we could create a warehouse that stores the DNA signatures of millions of cancer patients, tracks how the tumors change over time, and records both the treatments and the outcomes, we could create a gold mine of cancer fighting information. By participating, computer scientists can help ensure that such a "Million Genome Warehouse" is dependable, cost effective, and secure, and privacy protective.

We can't yet know how many cancer patients the faster software pipelines and Million Genome Warehouses will help—it could be tens, hundreds, thousands, or millions each year—but the sooner we create these tools, the more lives we can save.

What advice would you give to budding technologists who are considering careers in computing in this burgeoning new era in data analysis?

Study statistics and machine learning along with traditional CS courses like databases and operating systems.

As Big Data will surely be in the cloud, practice developing for Software as a Service (SaaS) deployed in the cloud rather the shrink-wrap software aimed at ground-bound PCs. Since Agile development is a perfect match for fast-changing SaaS apps, take a modern software engineering course to learn about Agile as well as productive programming environments for SaaS apps like Ruby on Rails or Python and Django.

Words and Expressions

multicore	/ˈmʌltɪkɔ:/	*a.*	多心的， 多核的
implementation	/ɪmplɪmenˈteɪʃn/	*n.*	履行；落实；装置
consult	/kənˈsʌlt/	*v.*	查阅；请教；商讨；就诊
coin	/kɔɪn/	*vt.*	铸造（钱币）；创造（新词）
installment	/ɪnˈstɔ:lmənt/	*n.*	分期付款；部分；分册
insight	/ˈɪnsaɪt/	*n.*	洞察力；见识；深刻的理解
expansion	/ɪkˈspænʃn/	*n.*	膨胀；扩展；扩充
embodiment	/ɪmˈbɒdimənt/	*n.*	化身；体现
address	/əˈdres/	*v.*	称呼；发表演说；提出；处理
analytics	/ˌænəˈlɪtɪks/	*n.*	分析学；解析学；分析论
agile	/ˈædʒaɪl/	*a.*	敏捷的；灵活的；机灵的
Ruby on Rails			一个相对较新的 Web 应用程序框架，构建在 Ruby 语言之上
Python			一种面向对象、直译式计算机程序设计语言

Django			一种 WEB 编程的框架，基于 Python 语言
practitioner	/prækˈtɪʃənə/	*n.*	从业者；实践者
interdisciplinary	/ˌɪntəˈdɪsəplɪnəri/	*a.*	各学科间的；跨学科的
embody	/ɪmˈbɒdi/	*vt.*	使具体化；包含；代表，体现
demonstration	/ˌdemənˈstreɪʃn/	*n.*	示范；表达；实证
virtually	/ˈvɜːtʃuəli/	*ad.*	实际上；几乎
scalability	/skeɪləˈbɪlɪtɪ/	*n.*	可伸缩性；可量测性
fuse	/fjuːz/	*vt.*	熔化；融合；装导线；装引信
computation	/ˌkɒmpjuˈteɪʃn/	*n.*	计算；估计；计算机的使用
validate	/ˈvælɪdeɪt/	*vt.*	使生效；证实；确认；验证
mutation	/mjuːˈteɪʃn/	*n.*	突变；变异
eradicate	/ɪˈrædɪkeɪt/	*vt.*	根除；根绝；消灭
bud	/bʌd/	*vt.*	使发芽
burgeoning	/ˈbɜːdʒənɪŋ/	*a.*	生机勃勃的；迅速发展的
deploy	/dɪˈplɔɪ/	*v.*	部署；展开；进入位置；配置

☞ ***Exercise A***

1. Match the words from the passage in Column A with the right explanations in Column B.

A	B
(1) academic	A. a person who pleads for a cause or propounds an idea
(2) advisory	B. extend over and cover a part of
(3) be noted for	C. associated with academia or an academy
(4) proponent	D. very successful and profitable
(5) pervasive	E. spreading or spread around
(6) booming	F. in particular
(7) inherent	G. make into a whole or make part of a whole
(8) notably	H. giving advice
(9) overlap	I. be well known for
(10) integrate	J. in the nature of something though not readily apparent

2. Choose a word or phrase from Column A to complete the following sentences, change the form if necessary.

(1) Numbering 3.1 million, the Bambara has their own writing system and ____ their sculpture in wood and metal.

(2) She finally succeeded in getting the good salary job just because she already has good ____ qualifications under her belt.

(3) This book explodes some____ myths about how to educate infants and children in the Middle Ages.

(4) Industry, frugality, kindness, hospitality are the ____ qualities of the Chinese nation.

(5) Farmers abandoned the land for more lucrative (赚钱的) employment in the____ construction industry.

(6) As many of my Boston University swimmers will attest, I have never been a major ____ of Laughlin's theories.

(7) Shops not only run special ____ services for the newcomer, but also offer consumers bits and pieces which they can assemble at home.

(8) The world of entertainment, most____Hollywood, has also contributed to the popularization of English.

(9) We should help the new-comers ____ quickly into the community.

(10) Obviously, the two sets of policies ____ and can complement each other.

☞ ***Exercise B***

Answer the following questions.

(1) What is RAD Lab? What's it mainly for?

(2) Among the 35 awards Patterson got for research, teaching and service, how many awards are mentioned in the passage?

(3) What are Patterson's pioneering contributions to RISC processor design?

(4) What do you know about Patterson's research characteristics in computer science?

(5) Does Patterson's research in computer science shape his career as a professor? Why?

(6) In the sentence "*We validate these ideas on real-world problems, such as cancer genomics. Recently, biologists discovered that cancer is a genetic disease, caused primarily by mutations in our DNA.*" What does "these ideas" refer to?

(7) What can "Million Genome Warehouse" do in Patterson's idea?

(8) How many cancer patients will be cured with the Big Data research?

(9) What does Patterson advise data analysis technologists to do in the interview?

(10) Do you think the Big Data research will have a great impact on cancer cure and other health care issues?

☞ ***Exercise C***

Translate the following paragraph into English.

美国国际商用机器公司（IBM）近日为一家名为"NuTec 科学"的生物科学公司研

制出一种超级计算机，可以迅速查询以往癌症病人的基因信息，以及何种疗法更为有效。医生可以据此信息对相同基因的新病人制订治疗方案。IBM 等公司的科学家希望，随着技术的进步，以及进一步了解基因和药物的相互作用关系，基于超级计算机的研究项目可以为不同的癌症患者研制专用的药物。

☞ ***Exercise D***

Write a short summary based on the article you have read in *Reading and Practice*, using as many new words and phrases you have learnt as possible. Your writing should be no less than 120 words.

Ⅱ. Reading and Comprehension

Fundamentals of Computer Design

Computer technology has made incredible progress in the roughly 60 years since the first general-purpose electronic computer was created. Today, less than $500 will purchase a personal computer that has more performance, more main memory, and more disk storage than a computer bought in 1985 for 1 million dollars. This rapid improvement has come both from advances in the technology used to build computers and from innovation in computer design. Although technological improvements have been fairly steady, progress arising from better computer architectures has been much less consistent. During the first 25 years of electronic computers, both forces made a major contribution, delivering performance improvement of about 25% per year. The late 1970s saw the emergence of the microprocessor. The ability of the microprocessor to ride the improvements in integrated circuit technology led to a higher rate of improvement—roughly 35% growth per year in performance.

This growth rate, combined with the cost advantages of a mass-produced microprocessor, led to an increasing fraction of the computer business being based on microprocessors. In addition, two significant changes in the computer marketplace made it easier than ever before to be commercially successful with a new architecture. First, the virtual elimination of assembly language programming reduced the need for object-code compatibility. Second, the creation of standardized, vendor-independent operating systems, such as UNIX and its clone, Linux, lowered the cost and risk of bringing out a new architecture. These changes made it possible to develop successfully a new set of architectures with simpler instructions, called RISC (Reduced Instruction Set Computer) architectures, in the early 1980s. The RISC-based machines focused the attention of designers on two critical performance techniques, the

exploitation of instruction level parallelism (initially through pipelining and later through multiple instruction issue) and the use of caches (initially in simple forms and later using more sophisticated organizations and optimizations). The RISC-based computers raised the performance bar, forcing prior architectures to keep up or disappear. The Digital Equipment Vax could not, and so it was replaced by a RISC architecture. Intel rose to the challenge, primarily by translating x86 (or IA-32) instructions into RISC-like instructions internally, allowing it to adopt many of the innovations first pioneered in the RISC designs. As transistor counts soared in the late 1990s, the hardware overhead of translating the more complex x86 architecture became negligible.

Figure 1 shows that the combination of architectural and organizational enhancements led to 16 years of sustained growth in performance at an annual rate of over 50%—a rate that is unprecedented in the computer industry. The effect of this dramatic growth rate in the 20th century has been twofold. First, it has significantly enhanced the capability available to computer users. For many applications, the highest-performance microprocessors of today outperform the supercomputer of less than 10 years ago.

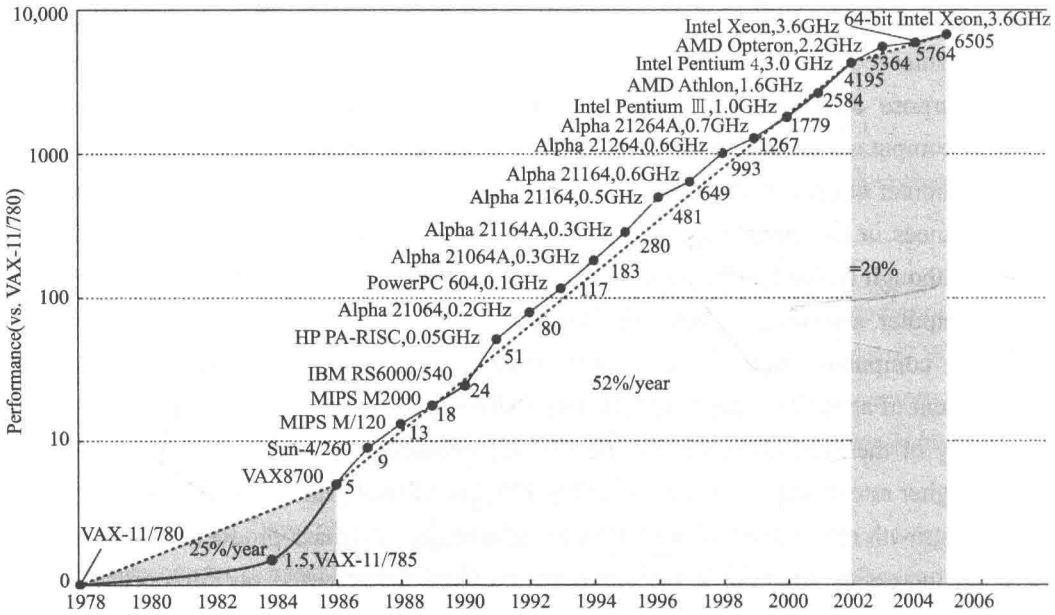


Figure 1 Growth in processor performance since the mid-1980s

This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks. Prior to the mid-1980s, processor performance growth was largely technology driven and averaged about 25% per year. The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. By 2002, this growth led to a difference in performance of about a factor of seven. Performance for floating-point-oriented calculations has increased even faster. Since 2002, the limits of power, available

instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 20% per year. Since SPEC has changed over the years, performance of newer machines is estimated by a scaling factor that relates the performance for two different versions of SPEC (e.g., SPEC92, SPEC95, and SPEC2000).

Second, this dramatic rate of improvement has led to the dominance of microprocessor-based computers across the entire range of the computer design. PCs and Workstations have emerged as major products in the computer industry. Minicomputers, which were traditionally made from off-the-shelf logic or from gate arrays, have been replaced by servers made using microprocessors. Mainframes have been almost replaced with multiprocessors consisting of small numbers of off-the-shelf microprocessors. Even high-end supercomputers are being built with collections of microprocessors. These innovations led to a renaissance in computer design, which emphasized both architectural innovation and efficient use of technology improvements. This rate of growth has compounded so that by 2002, high-performance microprocessors are about seven times faster than what would have been obtained by relying solely on technology, including improved circuit design.

However, Figure 1 also shows that this 16-year renaissance is over. Since 2002, processor performance improvement has dropped to about 20% per year due to the triple hurdles of maximum power dissipation of air-cooled chips, little instruction-level parallelism left to exploit efficiently, and almost unchanged memory latency. Indeed, in 2004 Intel canceled its high-performance uniprocessor projects and joined IBM and Sun in declaring that the road to higher performance would be via multiple processors per chip rather than via faster uniprocessors. This signals a historic switch from relying solely on instruction level parallelism (ILP), the primary focus of the first three editions of this book, to *thread-level parallelism* (TLP) and *data-level parallelism* (DLP), which are featured in this edition. Whereas the compiler and hardware conspire to exploit ILP implicitly without the programmer's attention, TLP and DLP are explicitly parallel, requiring the programmer to write parallel code to gain performance. People need to know about the architectural ideas and accompanying compiler improvements that made the incredible growth rate possible in the last century, the reasons for the dramatic change, and the challenges and initial promising approaches to architectural ideas and compilers for the 21st century. At the core is a quantitative approach to computer design and analysis that uses empirical observations of programs, experimentation, and simulation as its tools. The approach will work for explicitly parallel computers of the future just as it worked for the implicitly parallel computers of the past.

➔ Defining Computer Architecture

The task the computer designer faces is a complex one: Determine what attributes are important for a new computer, then design a computer to maximize performance while

staying within cost, power, and availability constraints. This task has many aspects, including instruction set design, functional organization, logic design, and implementation. The implementation may encompass integrated circuit design, packaging, power, and cooling. Optimizing the design requires familiarity with a very wide range of technologies, from compilers and operating systems to logic design and packaging.

In the past, the term *computer architecture* often referred only to instruction set design. Other aspects of computer design were called *implementation,* often insinuating that implementation is uninteresting or less challenging. We believe this view is incorrect. The architect's or designer's job is much more than instruction set design, and the technical hurdles in the other aspects of the project are likely more challenging than those encountered in instruction set design. We'll quickly review instruction set architecture before describing the larger challenges for the computer architect.

➔ ***Instruction Set Architecture***

We use the term *instruction set architecture* (ISA) to refer to the actual programmer visible instruction set in this book. The ISA serves as the boundary between the software and hardware. This quick review of ISA will use examples from MIPS and 80 × 86 to illustrate the seven dimensions of an ISA.

1. *Class of ISA*—Nearly all ISAs today are classified as general-purpose register architectures, where the operands are either registers or memory locations. The 80 × 86 has 16 general-purpose registers and 16 that can hold floating point data, while MIPS has 32 general-purpose and 32 floating-point registers (see Figure 2). The two popular versions of this class are *register-memory* ISAs such as the 80x86, which can access memory as part of many instructions, and *load-store* ISAs such as IPS, which can access memory only with load or store instructions. All recent ISAs are load-store.

2. *Memory addressing*—Virtually all desktop and server computers, including the 80x86 and MIPS, use byte addressing to access memory operands. Some architectures, like MIPS, require that objects must be *aligned* . An access to an object of size s bytes at byte address A is aligned if A mod s =0. The 80 × 86 does not require alignment, but accesses are generally faster if operands are aligned.

3. *Addressing modes*—In addition to specifying registers and constant operands, addressing modes specify the address of a memory object. MIPS addressing modes are Register, Immediate (for constants), and Displacement, where a constant offset is added to a register to form the memory address. The 80 × 86 supports those three plus three variations of displacement: no register (absolute), two registers (based indexed with displacement), two registers

where one register is multiplied by the size of the operand in bytes (based with scaled index and displacement). It has more like the last three, minus the displacement field: register indirect, indexed, and based with scaled index.

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

Figure 2 MIPS registers and usage convention

In addition to the 32 general purpose registers (R0–R31), MIPS has 32 floating-point registers (F0–F31) that can hold either a 32-bit single-precision number or a 64-bit double-precision number.

4. *Types and sizes of operands*—Like most ISAs, MIPS and 80 × 86 support operand sizes of 8-bit (ASCII character), 16-bit (Unicode character or half word), 32-bit (integer or word), 64-bit (double word or long integer), and IEEE 754 floating point in 32-bit (single precision) and 64-bit (double precision). The 80 × 86 also supports 80-bit floating point (extended double precision).

5. *Operations*—The general categories of operations are data transfer, arithmetic logical, control (discussed next), and floating point. MIPS is a simple and easy-to-pipeline instruction set architecture, and it is representative of the RISC architectures being used in 2006. The 80x86 has a much richer and larger set of operations.

6. *Control flow instructions*—Virtually all ISAs, including 80x86 and MIPS, support conditional branches, unconditional jumps, procedure calls, and returns. Both use PC-relative addressing, where the branch address is specified by an address field that is added to the PC. There are some small differences. MIPS