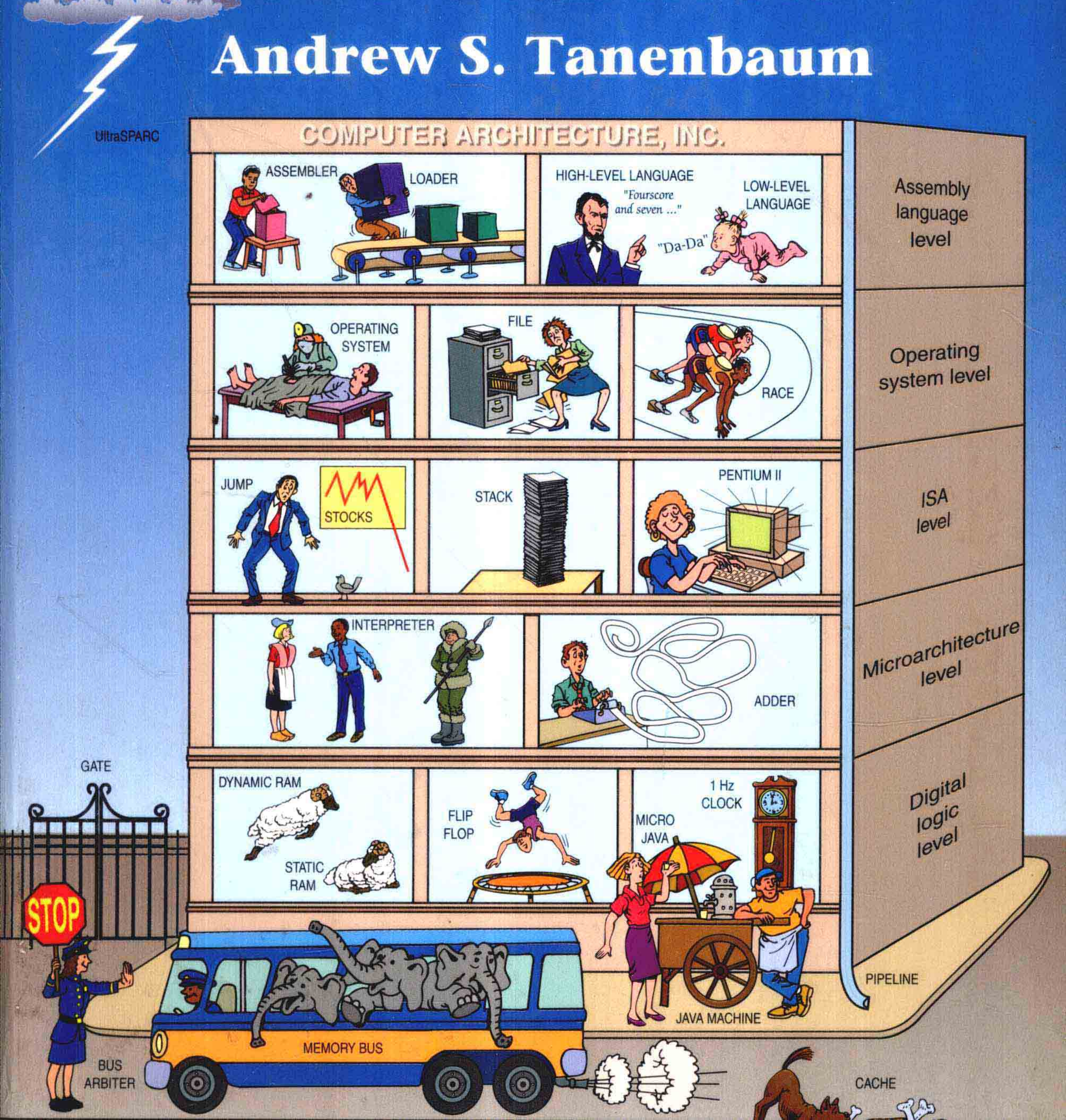
Fourth Edition

STRUCTURED COMPUTER ORGANIZATION

Andrew S. Tanenbaum



STRUCTURED COMPUTER ORGANIZATION

FOURTH EDITION

ANDREW S. TANENBAUM

Vrije Universiteit
Amsterdam, The Netherlands

With contributions from
JAMES R. GOODMAN

University of Wisconsin
Madison, WI

PRENTICE HALL

UPPER SADDLE RIVER, NEW JERSEY 07458

Library of Congress Cataloging-in-Publication Data

Tanenbaum, Andrew S.
Structured computer organization /
Andrew S. Tanenbaum. — 4th ed.
p. cm.
Includes bibliographical references and index.
ISBN: 0-13-095990-1
1. Computer programming. 2. Computer organization. I. Title.
QA76.6.T38 1999
004.2'2—dc21 98-42130
CIP

Publisher: **ALAN APT**
Development editor: **SONDRA CHAVEZ**
Editor-in-chief: **MARCIA HORTON**
Production editor: **IRWIN ZUCKER**
Managing editor: **EILEEN CLARK**
Manufacturing buyer: **PAT BROWN**
Copy editor: **MARTHA WILLIAMS**
Director of production and manufacturing: **DAVID W. RICCARDI**
Composition and interior design: **ANDREW S. TANENBAUM**
Cover concept: **ANDREW S. TANENBAUM**
Cover director: **ANN FRANCE**
Cover illustrator: **DON MARTINETTI, DM GRAPHICS, INC.**
Editorial assistant: **TONI HOLM**

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-095990-1

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall of Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To Suzanne, Barbara, Marvin, Bram, and the memory of Sweetie π

PREFACE

The first three editions of this book were based on the idea that a computer can be regarded as a hierarchy of levels, each one performing some well-defined function. This fundamental concept is as valid today as it was when the first edition came out, so it has been retained as the basis for the fourth edition. As in the first three editions, the digital logic level, the microarchitecture level, the instruction set architecture level, the operating system machine level, and the assembly language level are all discussed in detail (although we have changed some of the names to reflect modern practice).

Although the basic structure has been maintained, this fourth edition contains many changes, both small and large, that bring it up to date in the rapidly changing computer industry. For example, all the code examples, which were in Pascal, have been rewritten in Java, reflecting the popularity of Java in the computer world. Also, the example machines used have been brought up to date. The current examples are the Intel Pentium II, the Sun UltraSPARC II, and the Sun picoJava II, an embedded low-cost hardware Java chip.

Multiprocessors and parallel computers have also come in widespread use since the third edition, so the material on parallel architectures has been completely redone and greatly expanded, now covering a wide range of topics, from multiprocessors to COWs.

The book has become longer over the years (although still not as long as some other popular books on the subject). Such an expansion is inevitable as a subject develops and there is more known about it. As a result, when the book is used for a course, it may not always be possible to finish the book in a single course (e.g., in a trimester system). A possible approach would be to do all of Chaps. 1, 2, and 3, the first part of Chap. 4 (up through and including Sec. 4.4), and Chap. 5 as a bare minimum. The remaining time could be filled with the rest of Chap. 4, and parts of Chaps. 6, 7, and 8, depending on the interest of the instructor.

A chapter-by-chapter rundown of the major changes since the third edition follows. Chapter 1 still contains an historical overview of computer architecture, pointing out how we got where we are now and what the milestones were along the way. The enlarged spectrum of computers that exist is now discussed, and our three major examples (Pentium II, UltraSPARC II, and picoJava II) are introduced.

In Chapter 2, the material on input/output devices has been updated, emphasizing the technology of modern devices, including RAID disks, CD-Recordables, DVD, and color printers, among many others.

Chapter 3 (digital logic level) has undergone some revision and now treats computer buses and modern I/O chips. The major change here is additional material on buses, especially the PCI bus and the USB bus. The three new examples are described here at the chip level.

Chapter 4 (now called the microarchitecture level) has been completely rewritten. The idea of using a detailed example of a microprogrammed machine to illustrate the ideas of data path control has been retained, but the example machine is now a subset of the Java Virtual Machine. The underlying microarchitecture has been correspondingly changed. Several iterations of the design are given, showing what trade-offs are possible in terms of cost and performance. The last example, the Mic-4, uses a seven-stage pipeline and provides an easy introduction to how important modern computers, such as the Pentium II, work. A new section on improving performance has been added, focusing on the most recent techniques such as caching, branch prediction, (superscalar) out-of-order execution, speculative execution, and predication. The new example machines are discussed at the microarchitecture level.

Chapter 5 (now called the instruction set architecture level) deals with what many people refer to as "machine language." The Pentium II, UltraSPARC II and Java Virtual Machine are used as the primary examples here.

Chapter 6 (operating system machine level) has examples for the Pentium II (Windows NT) and UltraSPARC II (UNIX). The former is new and has many features that are worth looking at, but UNIX is still a reliable workhorse at many universities and companies and is well worth examining in detail as well due to its simple and elegant design.

Chapter 7 (assembly language level) has been brought up to date by using examples from the machines we have been studying. New material on dynamic linking has been added as well.

Chapter 8 (parallel computer architectures) has been completely rewritten from the third edition. It now covers both multiprocessors (UMA, NUMA, and COMA) in detail, as well as multicomputers (MPP and COW).

The bibliography has been extensively revised and brought up to date. Well over two-thirds the references refer to works published after the third edition was published. Binary numbers and floating-point numbers have not undergone much change recently, so the appendices are largely the same as in the previous edition.

Finally, some problems have been revised and many new problems have been added since the third edition. Accordingly, a new problem solutions manual is available from Prentice Hall. It is available *only* to faculty members, who can request a free copy from their Prentice Hall representative.

A Web site for this book is available. PostScript files for all the illustrations used in the book are available electronically. They can be fetched and printed, for example, for making overhead sheets. In addition, a simulator and other and software tools are there too. The URL for this site is

http://www.cs.vu.nl/~ast/sco4/

The simulator and software tools were produced by Ray Ontko. The author wishes to express his gratitude to Ray for producing these extremely useful programs.

A number of people have read (parts of) the manuscript and provided useful suggestions or have been helpful in other ways. In particular, I would like to thank Henri Bal, Alan Charlesworth, Kourosh Gharachorloo, Marcus Goncalves, Karen Panetta Lentz, Timothy Mattson, Harlan McGhan, Miles Murdocca, Kevin Normoyle, Mike O'Connor, Mitsunori Ogihara, Ray Ontko, Aske Plaat, William Potvin II, Nagarajan Prabhakaran. James H. Pugsley, Ronald N. Schroeder, Ryan Shoemaker, Charles Silio, Jr., and Dale Skrien for their help, for which I am most grateful. My students, especially Adriaan Bon, Laura de Vries, Dolf Loth, Guido van 't Noordende, have also helped debug the text. Thank you.

I would especially like to thank Jim Goodman for his many contributions to this book, especially to Chaps. 4 and 5. The idea of using the Java Virtual Machine was his, as were the microarchitectures for implementing it. Many of the advanced ideas were due to him. The book is far better for his having put in so much effort.

Finally, I would like to thank Suzanne for her patience for my long hours in front of my Pentium. From my point of view the Pentium is a big improvement over my older 386 but from hers, it does not make much difference. I also want to thank Barbara and Marvin for being great kids and Bram for always being quiet when I was trying to write.

Andrew S. Tanenbaum

CONTENTS

2 COMPUTER SYSTEMS ORGANIZATION 39

3 THE DIGITAL LOGIC LEVEL 117

5 THE INSTRUCTION SET ARCHITECTURE LEVEL 303

1

INTRODUCTION

A digital computer is a machine that can solve problems for people by carrying out instructions given to it. A sequence of instructions describing how to perform a certain task is called a **program**. The electronic circuits of each computer can recognize and directly execute a limited set of simple instructions into which all its programs must be converted before they can be executed. These basic instructions are rarely much more complicated than

Add 2 numbers.

Check a number to see if it is zero.

Copy a piece of data from one part of the computer's memory to another.

Together, a computer's primitive instructions form a language in which it is possible for people to communicate with the computer. Such a language is called a **machine language**. The people designing a new computer must decide what instructions to include in its machine language. Usually, they try to make the primitive instructions as simple as possible, consistent with the computer's intended use and performance requirements, in order to reduce the complexity and cost of the electronics needed. Because most machine languages are so simple, it is difficult and tedious for people to use them.

This simple observation has, over the course of time, led to a way of structuring computers as a series of abstractions, each abstraction building on the one below it. In this way, the complexity can be mastered and computer systems can