

# STRUCTURED COMPUTER ORGANIZATION

ANDREW S. TANENBAUM

Prentice-Hall Series in Automatic Computation

TP 33  
T164  
7960261

5

# STRUCTURED COMPUTER ORGANIZATION

**ANDREW S. TANENBAUM**

*Vrije Universiteit  
Amsterdam, The Netherlands*



E7950261

**PRENTICE-HALL, INC.**

ENGLEWOOD CLIFFS, NEW JERSEY

*Library of Congress Cataloging in Publication Data*

TANENBAUM, ANDREW S. (date)  
Structured computer organization.

Bibliography

1. Electronic digital computers—Programming.

I. Title.

QA76.6.T38      001.6'42      74-30322

ISBN 0-13-854505-7

© 1976 by Prentice-Hall, Inc., Englewood Cliffs, N. J.

All rights reserved. No part of this book  
may be reproduced in any form or by any means  
without permission in writing from the publisher.

10 9 8 7 6

Printed in the United States of America

PRENTICE-HALL INTERNATIONAL, INC., *London*  
PRENTICE-HALL OF AUSTRALIA, PTY. LTD., *Sydney*  
PRENTICE-HALL OF CANADA, LTD., *Toronto*  
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*  
PRENTICE-HALL OF JAPAN, INC., *Tokyo*  
PRENTICE-HALL OF SOUTHEAST ASIA (PTE.) LTD., *Singapore*

# **STRUCTURED COMPUTER ORGANIZATION**

Prentice-Hall  
Series in Automatic Computation

- AHO, ed., *Currents in the Theory of Computing*  
AHO AND ULLMAN, *The Theory of Parsing, Translation, and Compiling*,  
Volume I: *Parsing*; Volume II: *Compiling*  
ANDREE, *Computer Programming: Techniques, Analysis, and Mathematics*  
ANSELONE, *Collectively Compact Operator Approximation Theory  
and Applications to Integral Equations*  
BATES AND DOUGLAS, *Programming Language/One*, 2nd ed.  
BLUMENTHAL, *Management Information Systems*  
BRENT, *Algorithms for Minimization without Derivatives*  
BRINCH HANSEN, *Operating System Principles*  
COFFMAN AND DENNING, *Operating Systems Theory*  
CRESS, et al., *FORTRAN IV with WATFOR and WATFIV*  
DAHLQUIST, BJÖRCK, AND ANDERSON, *Numerical Methods*  
DANIEL, *The Approximate Minimization of Functionals*  
DEO, *Graph Theory with Applications to Engineering and Computer Science*  
DESMONDE, *Computers and Their Uses*, 2nd ed.  
DRUMMOND, *Evaluation and Measurement Techniques for Digital Computer Systems*  
ECKHOUSE, *Minicomputer Systems: Organization and Programming (PDP-11)*  
FIKE, *Computer Evaluation of Mathematical Functions*  
FIKE, *PL/I for Scientific Programmers*  
FORSYTHE AND MOLER, *Computer Solution of Linear Algebraic Systems*  
GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*  
GORDON, *System Simulation*  
GRISWOLD, *String and List Processing in SNOBOL4: Techniques and Applications*  
HANSEN, *A Table of Series and Products*  
HARTMANIS AND STEARNS, *Algebraic Structure Theory of Sequential Machines*  
JACOBY, et al., *Iterative Methods for Nonlinear Optimization Problems*  
JOHNSON, *System Structure in Data, Programs, and Computers*  
KIVIAT, et al., *The SIMSCRIPT II Programming Language*  
LAWSON AND HANSON, *Solving Least Squares Problems*  
LORIN, *Parallelism in Hardware and Software: Real and Apparent Concurrency*  
LOUDEN AND LEDIN, *Programming the IBM 1130*, 2nd ed.  
MARTIN, *Computer Data-Base Organization*  
MARTIN, *Design of Man-Computer Dialogues*  
MARTIN, *Design of Real-Time Computer Systems*  
MARTIN, *Future Developments in Telecommunications*  
MARTIN, *Programming Real-Time Computing Systems*  
MARTIN, *Security, Accuracy, and Privacy in Computer Systems*  
MARTIN, *Systems Analysis for Data Transmission*  
MARTIN, *Telecommunications and the Computer*

MARTIN, *Teleprocessing Network Organization*  
 MARTIN AND NORMAN, *The Computerized Society*  
 MCKEEMAN, et al., *A Compiler Generator*  
 MEYERS, *Time-Sharing Computation in the Social Sciences*  
 MINSKY, *Computation: Finite and Infinite Machines*  
 NIEVERGELT, et al., *Computer Approaches to Mathematical Problems*  
 PLANE AND MCMILLAN, *Discrete Optimization:*  
     *Integer Programming and Network Analysis for Management Decisions*  
 POLIVKA AND PAKIN, *APL: The Language and Its Usage*  
 PRITSKER AND KIVIAT, *Simulation with GASP II:*  
     *A FORTRAN-based Simulation Language*  
 PYLYSHYN, ed., *Perspectives on the Computer Revolution*  
 RICH, *Internal Sorting Methods Illustrated with PL/I Programs*  
 SACKMAN AND CITRENBBAUM, eds., *On-Line Planning:*  
     *Towards Creative Problem-Solving*  
 SALTON, ed., *The SMART Retrieval System:*  
     *Experiments in Automatic Document Processing*  
 SAMMET, *Programming Languages: History and Fundamentals*  
 SCHAEFER, *A Mathematical Theory of Global Program Optimization*  
 SCHULTZ, *Spline Analysis*  
 SCHWARZ, et al., *Numerical Analysis of Symmetric Matrices*  
 SHAH, *Engineering Simulation Using Small Scientific Computers*  
 SHAW, *The Logical Design of Operating Systems*  
 SHERMAN, *Techniques in Computer Programming*  
 SIMON AND SIKLOSSY, eds., *Representation and Meaning:*  
     *Experiments with Information Processing Systems*  
 STERBENZ, *Floating-Point Computation*  
 STOUTEMYER, *PL/I Programming for Engineering and Science*  
 STRANG AND FIX, *An Analysis of the Finite Element Method*  
 STROUD, *Approximate Calculation of Multiple Integrals*  
 TANENBAUM, *Structured Computer Organization*  
 TAVISS, ed., *The Computer Impact*  
 UHR, *Pattern Recognition, Learning, and Thought:*  
     *Computer-Programmed Models of Higher Mental Processes*  
 VAN TASSEL, *Computer Security Management*  
 VARGA, *Matrix Iterative Analysis*  
 WAITE, *Implementing Software for Non-Numeric Application*  
 WILKINSON, *Rounding Errors in Algebraic Processes*  
 WIRTH, *Systematic Programming: An Introduction*  
 YEH, ed., *Applied Computation Theory: Analysis, Design, Modeling*

*To Suzanne, Pluis, and Koe*

## PREFACE

Once upon a time computers were very simple. These early machines executed a small number of elementary instructions, and the user wrote his programs directly using these primitive instructions. Those days are long gone. A modern computer is a far more complicated entity, often consisting of a half dozen or more distinct levels at which it can be programmed and studied. In fact, nowadays it is often difficult to tell where the "machine" ends and where the software begins.

Many universities have a course in assembly language programming and computer organization early in the curriculum. There was a time when teaching the students how to program one specific computer in assembly language was considered adequate. Those days are also long gone. These courses must keep pace with a rapidly changing subject, and must now provide an introduction to a wide range of topics related to computer organization, many of which were barely known outside the research laboratories only a few years ago. Segmented virtual memories, parallel processing, race conditions, microprogramming, variable architecture machines, store and forward networks, self-virtualizing machines, and cache memories are just a few examples of these varied topics.

This book is intended as a textbook for such an introductory course on assembly language programming and computer organization. The only prerequisite is an introductory course in computer science (or equivalent practical experience) including a little knowledge of programming in a high-level language such as FORTRAN, COBOL, or PL/I. No mathematical or engineering background is needed, making the book suitable for use even at the sophomore level. Nearly all of the material of the B2 course in the ACM Curriculum 68 is covered, as well as similar material that has become important since that proposal was written. The book can also be used for the proposed COSINE course on "Machine Structure and Machine Language Programming." The chapters are self-contained, allowing them to be used as references on specific subjects.

The structured organization of computers, as a hierarchy of levels, is the theme of this book. At the bottom level is the true hardware, whose function is to execute



interpreters called microprograms. The language interpreted by the microprograms is the one most people think of as the "machine language" and is the one described in the manufacturer's machine reference manual. Thus, this "machine language" is already one level removed from the language directly executed by the electronic circuits. We call the lowest level "the microprogramming level" (Chapter 4) and the level it supports "the conventional machine level" (Chapter 3).

Most computers have an operating system that runs on the conventional machine level. The operating system provides its users with an "extended" or "virtual" machine that has instructions and facilities not present at the conventional machine level. These include file manipulation instructions, instructions for parallel processing (multi-tasking) and virtual memory. The set of features and instructions available to the user of the operating system can be regarded as defining a new level, "the operating system machine level" (Chapter 5).

The fourth level is the symbolic assembly language level. Unlike levels 2 and 3, which are supported by interpretation, this level is implemented by a program called an assembler (Chapter 6), which translates level 4 programs to one of the levels below it. Still higher levels, not covered in this book, are the levels defined by problem-oriented languages.

Chapter 7 examines the construction and applications of multilevel computers as a whole, in contrast to Chapters 3-6, each of which deals only with a single level. Chapter 8 is a guide to further study.

The IBM 370, CDC Cyber 70, and DEC PDP-11 computers are used as running examples throughout the text. These machines are used for illustrative purposes only, and no prior experience with any of them is expected.

There are a few algorithms written in a simple subset of PL/I, but these should be immediately understandable to anyone familiar with FORTRAN or ALGOL. PL/I was chosen as a compromise, since it is both widely used and suitable for structured programming. My own first choice was ALGOL 68, but few sophomores know it, unfortunately.

Many people have made contributions to this book. At times I have felt more like an editor than an author. Two people stand out above the others. Jack Alanen read the first handwritten draft in its entirety, and kept many bad ideas from even getting as far as the typewriter. He also made numerous suggestions for improving both the content and presentation. I especially thank Jack for teaching me a lot about teaching. Kim Gostelow went over a later version with a fine-tooth comb, commenting extensively, correcting errors, and trying to turn my prose into something resembling English. Nearly every single paragraph in the entire manuscript was marked with some suggestion for improvement. I am deeply indebted to both of them.

Reind van de Riet and Mitchell Tanenbaum also read and criticized the complete manuscript, providing me with different points of view. John W. Carr III, Arnie Falick, Dick Grune, Jim van Keulen, Bob Rosin, Carel Stillebroer and Wayne Wilner each offered suggestions and help with specific sections. I would also like to thank my students, especially Arie de Bruin, Wim Harmsen, Ad König, Sape Mullender, Johan

Stevenson, and Hans van Vliet for providing feedback. Mrs. Homburg-Knieper did a fine job of typing several versions of the manuscript.

I would like to express my appreciation to the International Business Machines Corporation, the Control Data Corporation, the Digital Equipment Corporation and the Burroughs Corporation for permission to adapt material from their copyrighted publications as follows: IBM—"IBM System/370 Principles of Operation Manual" and engineering drawings for the IBM 3125 CPU; CDC—"Control Data Cyber 70 Models 72, 73, 74 Computer Systems Reference Manual"; DEC—"PDP-11/45 Processor Handbook" and PDP-11/40 engineering drawings; Burroughs—"B1700 Systems Reference Manual." Any errors in the descriptions of these computers are my responsibility.

Finally, I want to thank Suzanne for encouragement, support, help, and especially patience throughout my long period of self-imposed exile from the human race during the preparation of this book. I also would like to thank her for breaking up the long hours with an *uitgeperste sinaasappel*.

ANDREW S. TANENBAUM

# CONTENTS

## PREFACE

xv

## INSTRUCTOR'S PREFACE

xviii

## 1 INTRODUCTION

1

- 1.1 Languages, Levels, and Virtual Machines 3
- 1.2 Contemporary Multilevel Machines 4
- 1.3 Historical Evolution of Multilevel Machines 7
- 1.4 Hardware, Software, and Multilevel Machines 10
- 1.5 Processes 12
- 1.6 Outline of this Book 14

## 2 COMPUTER SYSTEMS ORGANIZATION

18

- 2.1 Processors 18
  - 2.1.1 Instruction execution 19
  - 2.1.2 Parallel instruction execution 22
- 2.2 Memory 24
  - 2.2.1 Bits 25
  - 2.2.2 Memory addresses 25
  - 2.2.3 Metabits 28
  - 2.2.4 Secondary memory 29
    - Magnetic tapes 29
    - Magnetic disks 30
    - Magnetic drums 32
    - Optical memories 32
- 2.3 Input/Output 33
  - 2.3.1 I/O devices 33
  - 2.3.2 I/O processors 33
  - 2.3.3 Character codes 34
  - 2.3.4 Error-correcting codes 34
  - 2.3.5 Frequency-dependent codes 38
- 2.4 Transfer of Information 41
  - 2.4.1 Data paths 41
  - 2.4.2 Telecommunications 42

	Modulation	43
	Asynchronous and synchronous transmission	45
	Simplex, half-duplex, and full-duplex transmission	46
2.5	Computer Networks	47
2.6	Distributed Computers	49

### 3 THE CONVENTIONAL MACHINE LEVEL

3.1	Examples of the Conventional Machine Level	56
3.1.1	IBM System/360 and System/370	57
3.1.2	CDC 6000, Cyber 70, and Cyber 170	62
3.1.3	DEC PDP-11	65
3.2	Instruction Formats	70
3.2.1	Design criteria for instruction formats	71
3.2.2	Expanding opcodes	73
3.2.3	Examples of instruction formats	75
3.3	Addressing	79
3.3.1	Immediate addressing	80
3.3.2	Direct addressing	81
3.3.3	Register addressing	82
3.3.4	Indirect addressing	82
3.3.5	Indexing	84
3.3.6	Base registers	85
3.3.7	Stack addressing	87
	Reverse Polish	89
	Evaluation of reverse Polish formulas	90
3.3.8	Addressing on the PDP-11	93
3.3.9	Discussion of addressing modes	97
3.4	Instruction Types	98
3.4.1	Data movement instructions	99
3.4.2	Dyadic operations	100
3.4.3	Monadic operations	102
3.4.4	Comparisons and conditional jumps	104
3.4.5	Procedure call instructions	106
3.4.6	Loop control	107
3.4.7	Input/output	108
3.5	Representation of Data	111
3.5.1	Integers	112
3.5.2	Floating-point numbers	112
3.5.3	Booleans	112
3.5.4	Characters	113
3.5.5	Strings	113
3.5.6	Arrays	116
	Dope vectors	116
	Marginal indexing	117
3.6	Flow of Control	119
3.6.1	Sequential flow of control and jumps	119
3.6.2	Procedures	120
3.6.3	Coroutines	125
3.6.4	Traps	130
3.6.5	Interrupts	131

## THE MICROPROGRAMMING LEVEL

141

4.1	Processor Components	142
4.1.1	Registers	142
4.1.2	Buses	143
4.1.3	Gates	143
4.1.4	Clocks	144
4.1.5	Memory ports	145
4.1.6	Arithmetic and logical units	146
4.1.7	Packaging of processor components	147
4.2	Basic Operations	148
4.2.1	Register transfer	148
4.2.2	Memory read/write	150
4.2.3	Bit testing	150
4.3	A Hypothetical Target Level	150
4.4	A Hypothetical Host Level	153
4.4.1	The host level's registers	153
4.4.2	The host level's ALU	155
4.4.3	The host level's gates and data paths	155
4.5	Gate Sequences	157
4.5.1	Subcycles	157
4.5.2	Gate sequences for the ADD instruction	158
4.6	Microprogrammed Gate Control	160
4.6.1	Microinstructions	161
4.6.2	Execution of microprograms	162
4.6.3	A two-level machine	164
4.7	A Language for Microprogramming	165
4.7.1	Notation for GATE microinstructions	165
4.7.2	Notation for TEST microinstructions	166
4.8	The Interpreter for the Target Machine	167
4.8.1	Interpretation of the multiplication instruction	171
4.8.2	Interpretation of the division instruction	172
4.8.3	Perspective	175
4.9	Design of the Microprogramming Level	175
4.9.1	Encoded fields	176
4.9.2	Horizontal versus vertical organization	177
4.9.3	Memory cycle ratios and overlapped execution	180
4.9.4	Nanomemories	183
4.9.5	Universal versus specific microprogramming levels	185
4.9.6	Review of microprogramming level organization	186
4.10	Advantages and Disadvantages of Microprogramming	188
4.11	The IBM 370/125's Microprogramming Level	190
4.11.1	Architecture of the IBM 370/125 microprogramming level	190
4.11.2	IBM 3125 microinstructions	194
4.12	The PDP-11/40 microprogramming level	196
4.12.1	Architecture of the PDP-11/40 microprogramming level	196
4.12.2	UNIBUS operation	199
4.12.3	PDP-11/40 microinstructions	201
4.13	The Burroughs B1700	204
4.13.1	Architecture of the B1700	206
4.13.2	The B1700 instruction set	209

## 5 THE OPERATING SYSTEM MACHINE LEVEL

5.1	Implementation of the Operating System Machine Level	215
5.2	Virtual I/O Instructions	218
5.2.1	Sequential files	219
5.2.2	Random access files	220
5.2.3	Implementation of virtual i/o instructions	221
5.2.4	IBM 370 virtual i/o	226
5.2.5	Cyber 70 virtual i/o	229
5.2.6	PDP-11 virtual i/o	235
5.2.7	Comparison of level 3 i/o	236
5.3	Virtual Instructions Used in Parallel Processing	237
5.3.1	Process creation and destruction	238
5.3.2	Race conditions	239
5.3.3	Process synchronization using semaphores	242
5.3.4	Instructions for interprocess communication	245
5.4	Other Level 3 Instructions	246
5.4.1	Directory management instructions	246
5.4.2	Reconfiguring the level 3 machine	247
5.5	Virtual Memory	249
5.5.1	Paging	250
5.5.2	Implementation of paging	252
5.5.3	Demand paging and the working set model	256
5.5.4	Page replacement policy	259
5.5.5	The dirty bit	261
5.5.6	The hardware map	261
5.5.7	Page size and fragmentation	262
5.5.8	Cache memory	263
5.5.9	Segmentation	264
5.5.10	Virtual memory on the PDP-11	268
	Checkerboarding	271
5.5.11	The MULTICS virtual memory	273
5.5.12	Virtual memory on the IBM 370	277
5.5.13	Segmented virtual memory and file i/o	278
5.6	Job Control Languages	279

## 6 THE ASSEMBLY LANGUAGE LEVEL

6.1	Introduction to Assembly Language	289
6.1.1	What is an assembly language?	289
6.1.2	Format of an assembly language statement	290
6.1.3	Comparison of assembly language and PL/I	293
6.1.4	Program tuning	293
6.2	The Assembly Process	296
6.2.1	Two-pass assemblers	296
6.2.2	Pass one	297
6.2.3	Pass two	301
6.3	Searching and Sorting	303
6.3.1	Searching	303
6.3.2	Linear searching	304

6.3.3	Binary searching	305
6.3.4	Sorting	307
6.3.5	Hash coding	308
6.3.6	Hashing functions and collisions	312
6.3.7	Comparison of association techniques	315
6.4	Macros	316
6.4.1	Macro definition, call, and expansion	316
6.4.2	Macros with parameters	319
6.4.3	Conditional macro expansion	320
6.4.4	Nested macro calls	322
6.4.5	Recursive macro calls	323
6.4.6	Nested macro definitions	325
6.4.7	Implementation of a macro facility in an assembler	326
6.5	Linking and Loading	328
6.5.1	Tasks performed by the linker	329
6.5.2	Structure of an object module	332
6.5.3	Binding time and dynamic relocation	333
6.5.4	Dynamic linking	336

## 7 MULTILEVEL MACHINES

344

7.1	Methods of Implementing New Levels	344
7.1.1	Interpretation	344
7.1.2	Translation	346
	General-purpose macro processors	347
7.1.3	Procedural extension	349
7.2	Design Strategies for Multilevel Machines	349
7.2.1	Top-down design	350
7.2.2	Bottom-up design	352
7.2.3	Middle-out design	354
7.3	Program Portability	354
7.3.1	A universal programming language	355
7.3.2	The brute force approach	356
7.3.3	UNCOL	357
7.3.4	Do-it-yourself virtual machines	360
7.3.5	Emulation	363
7.3.6	Networks	364
7.4	Self-Virtualizing Machines	364
7.4.1	IBM VM/370 system	365
7.4.2	Goals of self-virtualizing machines	367
	Self-virtualizing machines and time sharing	368
	Operating system testing	368
	Protection of confidential data	369
7.4.3	Implementation of a self-virtualizing machine	370
	Exceptions and virtual machine faults	370
	Simulation of virtual machine i/o	372
	Self-modifying channel programs	372
	Shadow page tables	373
7.5	High-Level Machine Architecture	375
7.5.1	Addressing and descriptors	377
7.5.2	High-level machine instructions	381
7.5.3	Advantages and disadvantages of high-level machines	383

## 8 SUGGESTIONS FOR FURTHER READING AND BIBLIOGRAPHY

387

- 8.1 Suggestions for Further Reading 387
  - 8.1.1 Addressing and instructions 387
  - 8.1.2 Assemblers and assembly language programming 388
  - 8.1.3 Binary numbers and arithmetic 389
  - 8.1.4 Character codes, redundant and nonredundant 390
  - 8.1.5 Computer organization 390
  - 8.1.6 The conventional machine level 391
  - 8.1.7 Deadlocks 393
  - 8.1.8 File systems 393
  - 8.1.9 High-level machines 394
  - 8.1.10 Input/output 395
  - 8.1.11 Linkers and loaders 395
  - 8.1.12 Macros 396
  - 8.1.13 The microprogramming level 397
  - 8.1.14 Multilevel computers 398
  - 8.1.15 Networks 399
  - 8.1.16 Operating systems 400
  - 8.1.17 Parallel processing 401
  - 8.1.18 Self-virtualizing machines 402
  - 8.1.19 Symbol tables 403
  - 8.1.20 Telecommunications 405
  - 8.1.21 Virtual memory 405
- 8.2 Alphabetical Bibliography 407

## APPENDIX

### A FINITE-PRECISION ARITHMETIC AND BINARY NUMBERS

413

- A.1 Finite-Precision Numbers 413
- A.2 Radix Number Systems 415
- A.3 Conversion from One Radix to Another 416
- A.4 Negative Binary Numbers 420
- A.5 Binary Arithmetic 421

## APPENDIX

### B FLOATING-POINT NUMBERS

424

## APPENDIX

### C BOOLEAN ALGEBRA

432

## INDEX

436



# 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 two numbers

check a number to see if it is zero

move 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.

There are two principal ways to attack this problem; both involve designing a new set of instructions that is more convenient for people to use than the set of built-in machine instructions. Taken together, these new instructions also form a