Fifth Edition

STRUCTURED COMPUTER ORGANIZATION



Andrew S. Tanenbaum

CD-ROM Included

Samsung et. al. v. Techiya
IPR2024-00559
Exhibit 1025

Petitioners' Exhibit 1025
Page 01 of 21

STRUCTURED
COMPUTER ORGANIZATION

Other bestselling titles by Andrew S. Tanenbaum

Computer Networks, 4th edition

This widely-read classic, now in its fourth edition, provides the ideal introduction to today's and tomorrow's networks. It explains in detail how modern networks are structured. Starting with the physical layer and working up to the application layer, the book covers a vast number of important topics, including wireless communication, fiber optics, data link protocols, Ethernet, routing algorithms, network performance, security, DNS, electronic mail, USENET news, the World Wide Web, and multimedia. The book has especially thorough coverage of TCP/IP and the Internet.

Operating Systems: Design and Implementation, 2nd edition

This popular text on operating systems, co-authored with Albert S. Woodhull, is the only book covering both the principles of operating systems and their application to a real system. All the traditional operating systems topics are covered in detail. In addition, the principles are carefully illustrated with MINIX, a free POSIX-based UNIX-like operating system for personal computers. Each book contains a free CD-ROM containing the complete MINIX system, including all the source code. The source code is listed in an appendix to the book and explained in detail in the text.

Modern Operating Systems, 2nd edition

This comprehensive text covers the principles of modern operating systems in detail and illustrates them with numerous real-world examples. After an introductory chapter, the next five chapters deal with the basic concepts: processes and threads, deadlocks, memory management, input/output, and file systems. The next six chapters deal with more advanced material, including multimedia systems, multiple processor systems, security. Finally, two detailed case studies are given: UNIX/Linux and Windows 2000.

Distributed Systems: Principles and Paradigms

This new book, co-authored with Maarten van Steen, covers both the principles and paradigms of modern distributed systems. In the first part, it covers the principles of communication, processes, naming, synchronization, consistency and replication, fault tolerance, and security in detail. Then in the second part, it goes into different paradigms used to build distributed systems, including object-based systems, distributed file systems, document-based systems, and coordination-based systems.

STRUCTURED COMPUTER ORGANIZATION

FIFTH EDITION

ANDREW S. TANENBAUM

Vrije Universiteit

Amsterdam, The Netherlands



UPPER SADDLE RIVER, NEW JERSEY 07458

Petitioners' Exhibit 1025
Page 04 of 21

Library of Congress Cataloging-in-Publication Data

Tanenbaum, Andrew S.
Structured computer organization
/ Andrew S. Tanenbaum. —5th edition
p. cm.
Includes bibliographical references and index.
ISBN 0-13-148521-0
1. Computer programming. 2. Computer organization.
I. Title.

QA76.6.T38 2005
005.1--dc22

2005043004

Vice President and Editorial Director, ECS: Marcia Horton
Senior Acquisitions Editor: Tracy Dunkelberger
Editorial Assistant: Michael Giacobbe
Executive Managing Editor: Vince O'Brien
Managing Editor: Camille Trentacoste
Production Editor: Irwin Zucker
Director of Creative Services: Paul Belfanti
Art Director: Heather Scott
Cover Illustrator: Maraska Artistic Concept and Design
Don Martinetti, DM Graphics, Inc.
Cover Concept: Andrew S. Tanenbaum
Composition and interior design: Andrew S. Tanenbaum
Managing Editor, AV Management and Production: Patricia Burns
Art Editor: Gregory Dulles
Manufacturing Buyer: Lisa McDowell
Marketing Manager: Pamela Hersperger
Marketing Assistant: Barrie Reinhold



© 2006, 1999, 1990, 1984, 1976 by Pearson Education, Inc.
Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

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.

Pearson Prentice Hall® is a trademark of Pearson Education, Inc.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-148521-0

Pearson Education Ltd., London
Pearson Education Australia Pty. Ltd., Sydney
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., Hong Kong
Pearson Education Canada, Inc., Toronto
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education-Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., Upper Saddle River, New Jersey

To Suzan

CONTENTS

PREFACE xviii

1 INTRODUCTION 1

1.1 STRUCTURED COMPUTER ORGANIZATION 2
- 1.1.1 Languages, Levels, and Virtual Machines 2
- 1.1.2 Contemporary Multilevel Machines 5
- 1.1.3 Evolution of Multilevel Machines 8

1.2 MILESTONES IN COMPUTER ARCHITECTURE 13
- 1.2.1 The Zeroth Generation—Mechanical Computers (1642–1945) 14
- 1.2.2 The First Generation—Vacuum Tubes (1945–1955) 16
- 1.2.3 The Second Generation—Transistors (1955–1965) 19
- 1.2.4 The Third Generation—Integrated Circuits (1965–1980) 22
- 1.2.5 The Fourth Generation—Very Large Scale Integration (1980-?) 23
- 1.2.6 The Fifth Generation—Invisible Computers 26

1.3 THE COMPUTER ZOO 27
- 1.3.1 Technological and Economic Forces 27
- 1.3.2 The Computer Spectrum 29
- 1.3.3 Disposable Computers 29
- 1.3.4 Microcontrollers 31
- 1.3.5 Game Computers 33
- 1.3.6 Personal Computers 34

Petitioners' Exhibit 1025
Page 06 of 21

1.3.7 Servers 34
1.3.8 Collections of Workstations 34
1.3.9 Mainframes 36

1.4 EXAMPLE COMPUTER FAMILIES 37
1.4.1 Introduction to the Pentium 4 37
1.4.2 Introduction to the UltraSPARC III 42
1.4.3 Introduction to the 8051 44

1.5 METRIC UNITS 46

1.6 OUTLINE OF THIS BOOK 47

2 COMPUTER SYSTEMS ORGANIZATION 51

2.1 PROCESSORS 51
2.1.1 CPU Organization 52
2.1.2 Instruction Execution 54
2.1.3 RISC versus CISC 58
2.1.4 Design Principles for Modern Computers 59
2.1.5 Instruction-Level Parallelism 61
2.1.6 Processor-Level Parallelism 65

2.2 PRIMARY MEMORY 69
2.2.1 Bits 69
2.2.2 Memory Addresses 70
2.2.3 Byte Ordering 71
2.2.4 Error-Correcting Codes 73
2.2.5 Cache Memory 77
2.2.6 Memory Packaging and Types 80

2.3 SECONDARY MEMORY 81
2.3.1 Memory Hierarchies 81
2.3.2 Magnetic Disks 82
2.3.3 Floppy Disks 86
2.3.4 IDE Disks 86
2.3.5 SCSI Disks 88
2.3.6 RAID 89
2.3.7 CD-ROMs 93

Petitioners' Exhibit 1025
Page 07 of 21

2.3.8 CD-Recordables 97
2.3.9 CD-Rewritables 99
2.3.10 DVD 99
2.3.11 Blu-Ray 102

2.4 INPUT/OUTPUT 102
2.4.1 Buses 102
2.4.2 Terminals 105
2.4.3 Mice 110
2.4.4 Printers 112
2.4.5 Telecommunications Equipment 117
2.4.6 Digital Cameras 125
2.4.7 Character Codes 127

2.5 SUMMARY 131

3 THE DIGITAL LOGIC LEVEL 135

3.1 GATES AND BOOLEAN ALGEBRA 135
3.1.1 Gates 136
3.1.2 Boolean Algebra 138
3.1.3 Implementation of Boolean Functions 140
3.1.4 Circuit Equivalence 141

3.2 BASIC DIGITAL LOGIC CIRCUITS 146
3.2.1 Integrated Circuits 146
3.2.2 Combinational Circuits 147
3.2.3 Arithmetic Circuits 152
3.2.4 Clocks 157

3.3 MEMORY 159
3.3.1 Latches 159
3.3.2 Flip-Flops 161
3.3.3 Registers 163
3.3.4 Memory Organization 164
3.3.5 Memory Chips 168
3.3.6 RAMs and ROMs 171

3.4 CPU CHIPS AND BUSES 173
3.4.1 CPU Chips 174
3.4.2 Computer Buses 176
3.4.3 Bus Width 178
3.4.4 Bus Clocking 180
3.4.5 Bus Arbitration 184
3.4.6 Bus Operations 187

3.5 EXAMPLE CPU CHIPS 189
3.5.1 The Pentium 4 189
3.5.2 The UltraSPARC III 196
3.5.3 The 8051 200

3.6 EXAMPLE BUSES 202
3.6.1 The ISA Bus 203
3.6.2 The PCI Bus 204
3.6.3 PCI Express 212
3.6.4 The Universal Serial Bus 217

3.7 INTERFACING 221
3.7.1 I/O Chips 221
3.7.2 Address Decoding 222

3.8 SUMMARY 225

4 THE MICROARCHITECTURE LEVEL 231

4.1 AN EXAMPLE MICROARCHITECTURE 231
4.1.1 The Data Path 232
4.1.2 Microinstructions 239
4.1.3 Microinstruction Control: The Mic-1 241

4.2 AN EXAMPLE ISA: IJVM 246
4.2.1 Stacks 246
4.2.2 The IJVM Memory Model 248
4.2.3 The IJVM Instruction Set 250
4.2.4 Compiling Java to IJVM 254

Petitioners' Exhibit 1025
Page 09 of 21

4.3 AN EXAMPLE IMPLEMENTATION 255
4.3.1 Microinstructions and Notation 255
4.3.2 Implementation of IJVM Using the Mic-1 260

4.4 DESIGN OF THE MICROARCHITECTURE LEVEL 271
4.4.1 Speed versus Cost 271
4.4.2 Reducing the Execution Path Length 273
4.4.3 A Design with Prefetching: The Mic-2 281
4.4.4 A Pipelined Design: The Mic-3 281
4.4.5 A Seven-Stage Pipeline: The Mic-4 288

4.5 IMPROVING PERFORMANCE 292
4.5.1 Cache Memory 293
4.5.2 Branch Prediction 299
4.5.3 Out-of-Order Execution and Register Renaming 304
4.5.4 Speculative Execution 309

4.6 EXAMPLES OF THE MICROARCHITECTURE LEVEL 311
4.6.1 The Microarchitecture of the Pentium 4 CPU 312
4.6.2 The Microarchitecture of the UltraSPARC-III Cu CPU 317
4.6.3 The Microarchitecture of the 8051 CPU 323

4.7 COMPARISON OF THE PENTIUM, ULTRASPARC, AND 8051 325

4.8 SUMMARY 326

5 THE INSTRUCTION SET ARCHITECTURE LEVEL 331

5.1 OVERVIEW OF THE ISA LEVEL 333
5.1.1 Properties of the ISA Level 333
5.1.2 Memory Models 335
5.1.3 Registers 337
5.1.4 Instructions 339
5.1.5 Overview of the Pentium 4 ISA Level 339
5.1.6 Overview of the UltraSPARC III ISA Level 341
5.1.7 Overview of the 8051 ISA Level 345

Petitioners' Exhibit 1025
Page 10 of 21

5.2 DATA TYPES 348
5.2.1 Numeric Data Types 348
5.2.2 Nonnumeric Data Types 349
5.2.3 Data Types on the Pentium 4 350
5.2.4 Data Types on the UltraSPARC III 350
5.2.5 Data Types on the 8051 351

5.3 INSTRUCTION FORMATS 351
5.3.1 Design Criteria for Instruction Formats 352
5.3.2 Expanding Opcodes 354
5.3.3 The Pentium 4 Instruction Formats 357
5.3.4 The UltraSPARC III Instruction Formats 358
5.3.5 The 8051 Instruction Formats 359

5.4 ADDRESSING 360
5.4.1 Addressing Modes 360
5.4.2 Immediate Addressing 361
5.4.3 Direct Addressing 361
5.4.4 Register Addressing 361
5.4.5 Register Indirect Addressing 362
5.4.6 Indexed Addressing 363
5.4.7 Based-Indexed Addressing 365
5.4.8 Stack Addressing 365
5.4.9 Addressing Modes for Branch Instructions 369
5.4.10 Orthogonality of Opcodes and Addressing Modes 369
5.4.11 The Pentium 4 Addressing Modes 371
5.4.12 The UltraSPARC III Addressing Modes 373
5.4.13 The 8051 Addressing Modes 373
5.4.14 Discussion of Addressing Modes 374

5.5 INSTRUCTION TYPES 375
5.5.1 Data Movement Instructions 375
5.5.2 Dyadic Operations 376
5.5.3 Monadic Operations 377
5.5.4 Comparisons and Conditional Branches 379
5.5.5 Procedure Call Instructions 381
5.5.6 Loop Control 382
5.5.7 Input/Output 383
5.5.8 The Pentium 4 Instructions 386
5.5.9 The UltraSPARC III Instructions 389
5.5.10 The 8051 Instructions 392
5.5.11 Comparison of Instruction Sets 392

Petitioners' Exhibit 1025
Page 11 of 21

5.6 FLOW OF CONTROL 395
5.6.1 Sequential Flow of Control and Branches 395
5.6.2 Procedures 396
5.6.3 Coroutines 401
5.6.5 Traps 404
5.6.5 Interrupts 404

5.7 A DETAILED EXAMPLE: THE TOWERS OF HANOI 408
5.7.1 The Towers of Hanoi in Pentium 4 Assembly Language 409
5.7.2 The Towers of Hanoi in UltraSPARC III Assembly Language 409

5.8 THE IA-64 ARCHITECTURE AND THE ITANIUM 2 411
5.8.1 The Problem with the Pentium 4 413
5.8.2 The IA-64 Model: Explicitly Parallel Instruction Computing 414
5.8.3 Reducing Memory References 415
5.8.4 Instruction Scheduling 416
5.8.5 Reducing Conditional Branches: Predication 418
5.8.6 Speculative Loads 420

5.9 SUMMARY 421

6 THE OPERATING SYSTEM MACHINE LEVEL 427

6.1 VIRTUAL MEMORY 428
6.1.1 Paging 429
6.1.2 Implementation of Paging 431
6.1.3 Demand Paging and the Working Set Model 433
6.1.4 Page Replacement Policy 436
6.1.5 Page Size and Fragmentation 438
6.1.6 Segmentation 439
6.1.7 Implementation of Segmentation 442
6.1.8 Virtual Memory on the Pentium 4 445
6.1.9 Virtual Memory on the UltraSPARC III 450
6.1.10 Virtual Memory and Caching 452

6.2 VIRTUAL I/O INSTRUCTIONS 453
6.2.1 Files 454
6.2.2 Implementation of Virtual I/O Instructions 455
6.2.3 Directory Management Instructions 459

Petitioners' Exhibit 1025
Page 12 of 21

6.3 VIRTUAL INSTRUCTIONS FOR PARALLEL PROCESSING 460
6.3.1 Process Creation 461
6.3.2 Race Conditions 462
6.3.3 Process Synchronization Using Semaphores 466

6.4 EXAMPLE OPERATING SYSTEMS 470
6.4.1 Introduction 470
6.4.2 Examples of Virtual Memory 479
6.4.3 Examples of Virtual I/O 482
6.4.4 Examples of Process Management 493

6.5 SUMMARY 500

7 THE ASSEMBLY LANGUAGE LEVEL 507

7.1 INTRODUCTION TO ASSEMBLY LANGUAGE 508
7.1.1 What Is an Assembly Language? 508
7.1.2 Why Use Assembly Language? 509
7.1.3 Format of an Assembly Language Statement 512
7.1.4 Pseudoinstructions 515

7.2 MACROS 517
7.2.1 Macro Definition, Call, and Expansion 518
7.2.2 Macros with Parameters 520
7.2.3 Advanced Features 521
7.2.4 Implementation of a Macro Facility in an Assembler 521

7.3 THE ASSEMBLY PROCESS 522
7.3.1 Two-Pass Assemblers 522
7.3.2 Pass One 523
7.3.3 Pass Two 527
7.3.4 The Symbol Table 529

7.4 LINKING AND LOADING 530
7.4.1 Tasks Performed by the Linker 532
7.4.2 Structure of an Object Module 535
7.4.3 Binding Time and Dynamic Relocation 536
7.4.4 Dynamic Linking 539

7.5 SUMMARY 543

Petitioners' Exhibit 1025
Page 13 of 21

8 PARALLEL COMPUTER ARCHITECTURES 547

8.1 ON-CHIP PARALELLISM 548
- 8.1.1 Instruction-Level Parallelism 549
- 8.1.2 On-Chip Multithreading 556
- 8.1.3 Single-Chip Multiprocessors 562

8.2 COPROCESSORS 567
- 8.2.1 Network Processors 568
- 8.2.2 Media Processors 576
- 8.2.3 Cryptoprocessors 581

8.3 SHARED-MEMORY MULTIPROCESSORS 582
- 8.3.1 Multiprocessors vs. Multicomputers 582
- 8.3.2 Memory Semantics 590
- 8.3.3 UMA Symmetric Multiprocessor Architectures 594
- 8.3.4 NUMA Multiprocessors 602
- 8.3.5 COMA Multiprocessors 611

8.4 MESSAGE-PASSING MULTICOMPUTERS 612
- 8.4.1 Interconnection Networks 614
- 8.4.2 MPPs—Massively Parallel Processors 617
- 8.4.3 Cluster Computing 627
- 8.4.4 Communication Software for Multicomputers 632
- 8.4.5 Scheduling 635
- 8.4.6 Application-Level Shared Memory 636
- 8.4.7 Performance 643

8.5 GRID COMPUTING 649

8.6 SUMMARY 651

9 READING LIST AND BIBLIOGRAPHY 655

9.1 SUGGESTIONS FOR FURTHER READING 655
- 9,1,1 Introduction and General Works 655
- 9.1.2 Computer Systems Organization 657
- 9.1.3 The Digital Logic Level 658

9.1.4 The Microarchitecture Level 659
9.1.5 The Instruction Set Architecture Level 659
9.1.6 The Operating System Machine Level 660
9.1.7 The Assembly Language Level 661
9.1.8 Parallel Computer Architectures 661
9.1.9 Binary and Floating-Point Numbers 663
9.1.10 Assembly Language Programming 664

9.2 ALPHABETICAL BIBLIOGRAPHY 664

A BINARY NUMBERS 679

A.1 FINITE-PRECISION NUMBERS 679

A.2 RADIX NUMBER SYSTEMS 681

A.3 CONVERSION FROM ONE RADIX TO ANOTHER 683

A.4 NEGATIVE BINARY NUMBERS 685

A.5 BINARY ARITHMETIC 688

B FLOATING-POINT NUMBERS 691

B.1 PRINCIPLES OF FLOATING POINT 692

B.2 IEEE FLOATING-POINT STANDARD 754 694

C ASSEMBLY LANGUAGE PROGRAMMING 701

C.1 OVERVIEW 702
C.1.1 Assembly Language 702
C.1.2 A Small Assembly Language Program 703

Petitioners' Exhibit 1025
Page 15 of 21

C.2 THE 8088 PROCESSOR 704
C.2.1 The Processor Cycle 705
C.2.2 The General Registers 705
C.2.3 Pointer Registers 708

C.3 MEMORY AND ADDRESSING 709
C.3.1 Memory Organization and Segments 709
C.3.2 Addressing 711

C.4 THE 8088 INSTRUCTION SET 715
C.4.1 Move, Copy and Arithmetic 715
C.4.2 Logical, Bit and Shift Operations 718
C.4.3 Loop and Repetitive String Operations 718
C.4.4 Jump and Call Instructions 719
C.4.5 Subroutine Calls 721
C.4.6 System Calls and System Subroutines 723
C.4.7 Final Remarks on the Instruction Set 725

C.5 THE ASSEMBLER 725
C.5.1 Introduction 726
C.5.2 The ACK-Based Tutorial Assembler as88 727
C.5.3 Some Differences with Other 8088 Assemblers 730

C.6 THE TRACER 732
C.6.1 Tracer Commands 734

C.7 GETTING STARTED 735

C.8 EXAMPLES 736
C.8.1 Hello World Example 736
C.8.2 General Registers Example 740
C.8.3 Call Command and Pointer Registers 742
C.8.4 Debugging an Array Print Program 744
C.8.5 String Manipulation and String Instructions 748
C.8.6 Dispatch Tables 750
C.8.7 Buffered and Random File Access 752

INDEX **757**

Petitioners' Exhibit 1025
Page 16 of 21

2.3.11 Blu-Ray

Nothing stands still in the computer business, certainly not storage technology. DVD was barely introduced before its successor threatened to make it obsolete. The successor to DVD is **Blu-Ray**, so called because it uses a blue laser instead of the red one used by DVDs. A blue laser has a shorter wavelength than a red one, which allows it to focus more accurately and thus support smaller pits and lands. Single-sided Blu-Ray disks hold about 25 GB of data; double-sided ones hold about 50 GB. The data rate is about 4.5 MB/sec, which is good for an optical disk, but still insignificant compared to magnetic disks (cf. ATAPI-6 at 100 MB/sec and wide Ultra4 SCSI at 320 MB/sec). It is expected that Blu-Ray will eventually replace CD-ROMs and DVDs, but this transition will take some years.

2.4 INPUT/OUTPUT

As we mentioned at the start of this chapter, a computer system has three major components: the CPU, the memories (primary and secondary), and the **I/O** (**Input/Output**) equipment such as printers, scanners, and modems. So far we have looked at the CPU and the memories. Now it is time to examine the I/O equipment and how it is connected to the rest of the system.

2.4.1 Buses

Physically, most personal computers and workstations have a structure similar to the one shown in Fig. 2-28. The usual arrangement is a metal box with a large printed circuit board at the bottom, called the **motherboard** (parentboard, for the politically correct). The motherboard contains the CPU chip, some slots into which DIMM modules can be clicked, and various support chips. It also contains a bus etched along its length, and sockets into which the edge connectors of I/O boards can be inserted (the PCI bus). Older PCs also have a second bus (the ISA bus), for legacy I/O boards, but modern computers usually lack it and it is rapidly dying off.

The logical structure of a simple low-end personal computer is shown in Fig. 2-29. This one has a single bus used to connect the CPU, memory, and I/O devices; most systems have two or more buses. Each I/O device consists of two parts: one containing most of the electronics, called the **controller**, and one containing the I/O device itself, such as a disk drive. The controller is usually contained on a board plugged into a free slot, except for those controllers that are not optional (such as the keyboard), which are sometimes located on the motherboard. Even though the display (monitor) is not an option, the video controller is sometimes located on a plug-in board to allow the user to choose between boards with

Petitioners' Exhibit 1025
Page 17 of 21

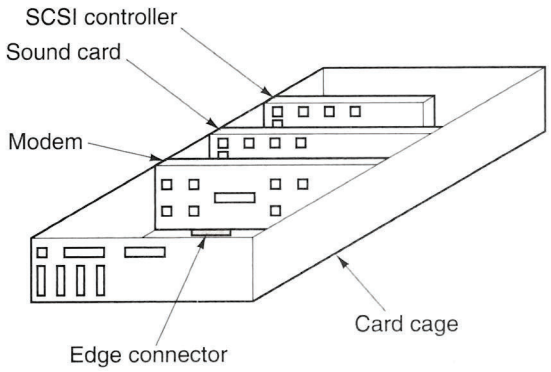


Figure 2-28. Physical structure of a personal computer.

or without graphics accelerators, extra memory, and so on. The controller connects to its device by a cable attached to a connector on the back of the box.

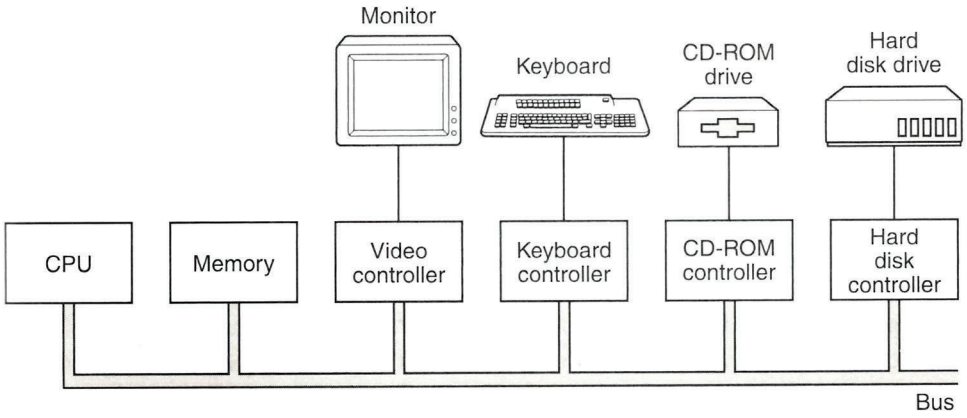


Figure 2-29. Logical structure of a simple personal computer.

The job of a controller is to control its I/O device and handle bus access for it. When a program wants data from the disk for example, it gives a command to the disk controller, which then issues seeks and other commands to the drive. When the proper track and sector have been located, the drive begins outputting the data as a serial bit stream to the controller. It is the job of the controller to break the bit stream up into units, and write each unit into memory, as it is assembled. A unit is typically one or more words. A controller that reads or writes data to or from memory without CPU intervention is said to be performing **Direct Memory Access**, better known by its acronym **DMA**. When the transfer is completed, the controller normally causes an **interrupt**, forcing the CPU to immediately suspend running its current program and start running a special procedure, called an

Petitioners' Exhibit 1025
Page 18 of 21

interrupt handler, to check for errors, take any special action needed, and inform the operating system that the I/O is now finished. When the interrupt handler is finished, the CPU continues with the program that was suspended when the interrupt occurred.

The bus is not only used by the I/O controllers, but also by the CPU for fetching instructions and data. What happens if the CPU and an I/O controller want to use the bus at the same time? The answer is that a chip called a **bus arbiter** decides who goes next. In general, I/O devices are given preference over the CPU, because disks and other moving devices cannot be stopped, and forcing them to wait would result in lost data. When no I/O is in progress, the CPU can have all the bus cycles for itself to reference memory. However, when some I/O device is also running, that device will request and be granted the bus when it needs it. This process is called **cycle stealing** and it slows down the computer.

This design worked fine for the first personal computers, since all the components were roughly in balance. However, as the CPUs, memories, and I/O devices got faster, a problem arose: the bus could no longer handle the load presented. On a closed system, such as an engineering workstation, the solution was to design a new and faster bus for the next model. Because nobody ever moved I/O devices from an old model to a new one, this approached worked fine.

However, in the PC world, people often upgraded their CPU but wanted to move their printer, scanner, and modem to the new system. Also, a huge industry had grown up around providing a vast range of I/O devices for the IBM PC bus, and this industry had exceedingly little interest in throwing out its entire investment and starting over. IBM learned this the hard way when it brought out the successor to the IBM PC, the PS/2 range. The PS/2 had a new, and faster bus, but most clone makers continued to use the old PC bus, now called the **ISA (Industry Standard Architecture)** bus. Most disk and I/O device makers also continued to make controllers for it, so IBM found itself in the peculiar situation of being the only PC maker that was no longer IBM compatible. Eventually, it was forced back to supporting the ISA bus. As an aside, please note that ISA stands for Instruction Set Architecture in the context of machine levels whereas it stands for Industry Standard Architecture in the context of buses.

Nevertheless, despite the market pressure not to change anything, the old bus really was too slow, so something had to be done. This situation led to other companies developing machines with multiple buses, one of which was the old ISA bus, or its backward-compatible successor, the **EISA (Extended ISA)** bus. The most popular of these now is the **PCI (Peripheral Component Interconnect)** bus. It was designed by Intel, but Intel decided to put all the patents in the public domain, to encourage the entire industry (including its competitors) to adopt it.

The PCI bus can be used in many configurations, but a typical one is illustrated in Fig. 2-30. Here the CPU talks to a memory controller over a dedicated high-speed connection. The controller talks to the memory and to the PCI bus directly, so CPU-memory traffic does not go over the PCI bus. However, high-

Petitioners' Exhibit 1025
Page 19 of 21

bandwidth (i.e., high data rate) peripherals can connect to the PCI bus directly. In addition, here the PCI bus has a bridge to the ISA bus, so that ISA controllers and their devices can still be used, although as mentioned earlier, the ISA bus is being phased out. A machine of this design would typically contain three or four empty PCI slots and one or two ISA slots, to allow customers to plug in both old ISA I/O cards (usually for slow devices) and new PCI I/O cards (usually for fast devices).

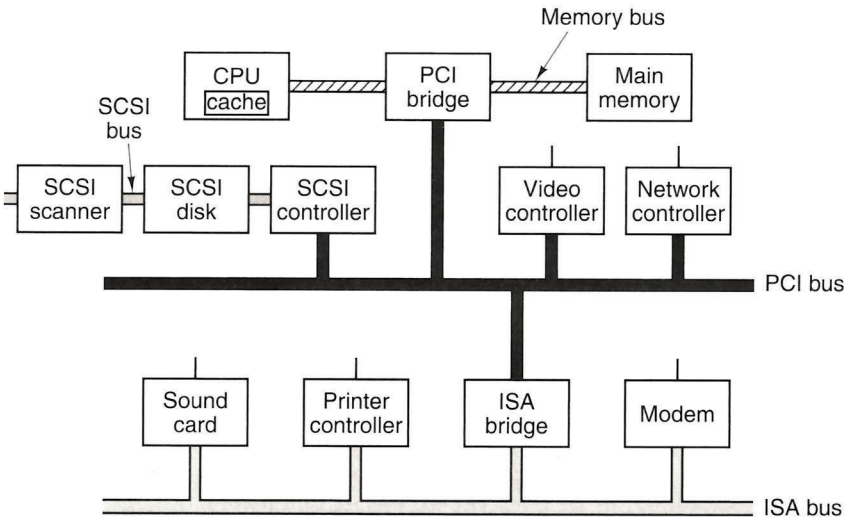


Figure 2-30. A typical modern PC with a PCI bus and an ISA bus. The modem and sound card are ISA devices; the SCSI controller is a PCI device.

Many kinds of I/O devices are available today. A few of the more common ones are discussed below.

2.4.2 Terminals

Computer terminals consist of two parts: a keyboard and a monitor. In the mainframe world, these parts are often integrated into a single device and attached to the main computer by a serial line or over a telephone line. In the airline reservation, banking, and other mainframe-oriented industries, these devices are still in widespread use. In the personal computer world, the keyboard and monitor are independent devices. Either way, the technology of the two parts is the same.

Keyboards

Keyboards come in several varieties. The original IBM PC came with a keyboard that had a snap-action switch under each key that gave tactile feedback and made a click when the key was depressed far enough. Nowadays, the cheaper

Petitioners' Exhibit 1025
Page 20 of 21

UNICODE uses the same code point for characters that look almost identical but have different meanings or are written slightly differently in Japanese and Chinese (as though English word processors always spelled "blue" as blew" because they sound the same). Some people view this as an optimization to save scarce code points; others see it as Anglo-Saxon cultural imperialism (and you thought assigning 16-bit values to characters was not highly political?). To make matters worse, a full Japanese dictionary has 50,000 kanji (excluding names), so with only 20,992 code points available for the Han ideographs, choices had to be made. Not all Japanese people think that a consortium of computer companies, even if a few of them are Japanese, is the ideal forum to make these choices.

2.5 SUMMARY

Computer systems are built up from three types of components: processors, memories, and I/O devices. The task of a processor is to fetch instructions one at a time from a memory, decode them, and execute them. The fetch-decode-execute cycle can always be described as an algorithm and, in fact, is sometimes carried out by a software interpreter running at a lower level. To gain speed, many computers now have one or more pipelines or have a superscalar design with multiple functional units that operate in parallel.

Systems with multiple processors are increasingly common. Parallel computers include array processors, on which the same operation is performed on multiple data sets at the same time, multiprocessors, in which multiple CPUs share a common memory, and multicomputers, in which multiple computers each have their own memories but communicate by message passing.

Memories can be categorized as primary or secondary. The primary memory is used to hold the program currently being executed. Its access time is short—a few tens of nanoseconds at most—and independent of the address being accessed. Caches reduce this access time even more. Some memories are equipped with error-correcting codes to enhance reliability.

Secondary memories, in contrast, have access times that are much longer (milliseconds or more) and dependent on the location of the data being read or written. Tapes, magnetic disks and optical disks are the most common secondary memories. Magnetic disks come in many varieties, including floppy disks, Winchester disks, IDE disks, SCSI disks, and RAIDs. Optical disks include CD-ROMs, CD-Rs, and DVDs.

I/O devices are used to transfer information into and out of the computer. They are connected to the processor and memory by one or more buses. Examples are terminals, mice, printers, and modems. Most I/O devices use the ASCII character code, although UNICODE is rapidly gaining acceptance as the computer industry goes global.

Petitioners' Exhibit 1025
Page 21 of 21