UNDERSTANDING THE DIGITAL WORLD

What You Need to Know about Computers, the Internet, Privacy, and Security

BRIAN W. KERNIGHAN

Understanding the Digital World

What You Need to Know about Computers, the Internet, Privacy, and Security

Brian W. Kernighan

Princeton University Press
Princeton and Oxford

Published by Princeton University Press
41 William Street, Princeton, New Jersey 08540

In the United Kingdom: Princeton University Press
6 Oxford Street, Woodstock, Oxfordshire OX20 1TR

press.princeton.edu

ISBN 978-0-691-17654-3

British Library Cataloging-in-Publication Data is available

This book has been composed in Times, Courier and Helvetica
using groff, ghostscript, and other open source Unix tools.

The publisher would like to acknowledge the author of this volume for
providing the print-ready files from which this book was printed.

Printed on acid-free paper. ∞

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents

Preface xi

Introduction 1

Part I: Hardware 7

1. What's in a Computer? 11
1.1 Logical Construction 13
1.1.1 CPU 13
1.1.2 RAM 14
1.1.3 Disks and other secondary storage 15
1.1.4 Et cetera 17
1.2 Physical Construction 17
1.3 Moore's Law 21
1.4 Summary 22

2. Bits, Bytes, and Representation of Information 23
2.1 Analog versus Digital 23
2.2 Analog-Digital Conversion 25
2.3 Bits, Bytes, and Binary 30
2.3.1 Bits 30
2.3.2 Powers of two and powers of ten 31
2.3.3 Binary numbers 32
2.3.4 Bytes 34
2.4 Summary 36

3. Inside the CPU 37
3.1 The Toy Computer 38
3.1.1 The first Toy program 38
3.1.2 The second Toy program 40
3.1.3 Branch instructions 41
3.1.4 Representation in RAM 43
3.2 Real CPUs 43
3.3 Caching 46

3.4 Other Kinds of Computers 47
3.5 Summary 49

Wrapup on Hardware 51

Part II: Software 53

4. Algorithms 55
4.1 Linear Algorithms 56
4.2 Binary Search 58
4.3 Sorting 59
4.4 Hard Problems and Complexity 63
4.5 Summary 65

5. Programming and Programming Languages 67
5.1 Assembly Language 68
5.2 High-Level Languages 69
5.3 Software Development 75
5.3.1 Libraries, interfaces, and development kits 76
5.3.2 Bugs 77
5.4 Intellectual Property 79
5.4.1 Trade secret 80
5.4.2 Copyright 80
5.4.3 Patents 81
5.4.4 Licenses 82
5.5 Standards 84
5.6 Open Source 84
5.7 Summary 86

6. Software Systems 87
6.1 Operating Systems 88
6.2 How an Operating System Works 92
6.2.1 System calls 93
6.2.2 Device drivers 93
6.3 Other Operating Systems 94
6.4 File Systems 95
6.4.1 Disk file systems 96
6.4.2 Removing files 98
6.4.3 Other file systems 99
6.5 Applications 100
6.6 Layers of Software 102
6.7 Summary 104

7. Learning to Program 105
7.1 Programming Language Concepts 106
7.2 A First JavaScript Example 107
7.3 A Second JavaScript Example 107
7.4 Loops 110
7.5 Conditionals 111
7.6 Libraries and Interfaces 112
7.7 How JavaScript Works 114
7.8 Summary 114

Wrapup on Software 117

Part III: Communications 119

8. Networks 125
8.1 Telephones and Modems 126
8.2 Cable and DSL 126
8.3 Local Area Networks and Ethernet 128
8.4 Wireless 130
8.5 Cell Phones 131
8.6 Bandwidth 135
8.7 Compression 135
8.8 Error Detection and Correction 137
8.9 Summary 139

9. The Internet 141
9.1 An Internet Overview 142
9.2 Domain Names and Addresses 145
9.2.1 Domain Name System 145
9.2.2 IP addresses 146
9.2.3 Root servers 147
9.2.4 Registering your own domain 148
9.3 Routing 148
9.4 TCP/IP Protocols 150
9.4.1 IP, the Internet Protocol 151
9.4.2 TCP, the Transmission Control Protocol 152
9.5 Higher-Level Protocols 153
9.5.1 Telnet and SSH: remote login 154
9.5.2 SMTP: Simple Mail Transfer Protocol 154
9.5.3 File sharing and peer-to-peer protocols 156
9.6 Copyright on the Internet 157
9.7 The Internet of Things 159
9.8 Summary 159

10. The World Wide Web 163
10.1 How the Web Works 164
10.2 HTML 165
10.3 Cookies 167
10.4 Active Content in Web Pages 168
10.5 Active Content Elsewhere 170
10.6 Viruses, Worms and Trojan Horses 171
10.7 Web Security 173
10.7.1 Attacks on clients 174
10.7.2 Attacks on servers 177
10.7.3 Attacks on information in transit 179
10.8 Defending Yourself 179
10.9 Summary 181

11. Data and Information 183
11.1 Search 184
11.2 Tracking 188
11.3 Social Networks 193

11.4 Data Mining and Aggregation 195
11.5 Cloud Computing 197
11.6 Summary 202

12. Privacy and Security 203
12.1 Cryptography 204
12.1.1 Secret-key cryptography 205
12.1.2 Public-key cryptography 206
12.2 Anonymity 210
12.2.1 Tor and the Tor Browser 211
12.2.2 Bitcoin 213
12.3 Summary 215

13. Wrapping Up 217

Notes 221

Glossary 227

Index 233

Preface

Since the fall of 1999, I have been teaching a Princeton course called "Computers in Our World." The course title is embarrassingly vague, but I had to invent it in less than five minutes one day and then it became too hard to change. Teaching the course itself, however, has proven to be the most fun thing that I do, in a job that is almost all enjoyable.

The course is based on the observation that computers and computing are all around us. Some computing is highly visible: every student has a computer that is far more powerful than the single IBM 7094 computer that cost several million dollars, occupied a very large air-conditioned room, and served the whole Princeton campus when I was a graduate student there in 1964. Every student has a cell phone too, also with much more computing power than that 1964 computer. Every student has high-speed Internet access, as does a significant fraction of the world's population. Everyone searches and shops online, and uses email, texting and social networks to keep in touch with friends and family.

But this is only part of a computing iceberg, much of which lies hidden below the surface. We don't see and usually don't think about the computers that lurk within appliances, cars, airplanes and the pervasive electronic gadgets that we take for granted—cameras, DVD players, tablets, GPS navigators, games. Nor do we think much about the degree to which infrastructure depends on computing: the telephone network, cable television, air traffic control, the power grid, and banking and financial services.

Most people will not be directly involved in creating such systems, but everyone is strongly affected by them, and some will have to make important decisions about them. Wouldn't it be better if people had a better understanding of computers? An educated person ought to know at least the rudiments of computing: what computers can do and how they do it; what they can't do at all and what's merely extremely hard right now; how they talk to each other and what happens when they do; and the many ways that computing and communications influence the world around us.

The pervasive nature of computing affects us in unexpected ways. Although we are from time to time reminded of the growth of surveillance systems, incursions into our privacy, and the perils of identity theft, we perhaps do not realize the extent to which they are enabled by computing and communications.

In June 2013, Edward Snowden, a contractor at the United States National Security Agency (NSA), provided journalists with documents which revealed that the NSA had been routinely monitoring and collecting the electronic communications—phone calls, email, Internet use—of pretty much everyone in the world, but notably of American citizens living in the US who were no threat whatsoever to their country. The Snowden documents also showed that other countries were spying on their citizens, for instance the Government Communications Headquarters (GCHQ), the United Kingdom's equivalent of the NSA. Intelligence agencies routinely share information with each other, but not all of it, so it was probably a bit of a surprise in the German intelligence community to learn that the NSA was eavesdropping on the cell phone of Germany's chancellor, Angela Merkel.

Corporations also track and monitor what we do online and in the real world, and have made it hard for anyone to be anonymous. The availability of voluminous data has enabled great progress in speech understanding, image recognition and language translation, but it has come at a cost to our privacy.

Criminals have become sophisticated in their attacks on data repositories. Electronic break-ins at businesses and government agencies are frequent; information about customers and employees is stolen in large quantities, often to be used for fraud and identity theft. Attacks on individuals are common as well. It used to be that one could be fairly safe from online scams by simply ignoring mail from putative Nigerian princes or their relatives, but targeted attacks are now far more subtle and have become one of the most common ways in which corporate computers are breached.

Jurisdictional issues are difficult too. The European Union has required major search engines to provide a "right to be forgotten" mechanism so that ordinary people can have their online history excluded from search engine results. The EU also established rules that require companies that store data about EU citizens to do so on servers in the EU, not in the US. Of course these rules apply only in the EU and are different in other parts of the world.

The rapid adoption of cloud computing, where individuals and companies store their data and do their computing on servers owned by Amazon, Google, Microsoft and any number of others, adds another layer of complexity. Data is no longer held directly by its owners but rather by third parties that have different agendas, responsibilities and vulnerabilities, and may face jurisdictional requirements.

There's a rapidly growing "Internet of Things" in which all kinds of devices connect to the Internet. Cell phones are an obvious instance, of course, but it's also cars, security cameras, home appliances and controls, medical equipment, and a great deal of infrastructure like air traffic control and power grids. This trend towards connecting everything in sight to the Internet will continue, because the benefits of connection are compelling. Unfortunately, there are many risks, since security for such devices is much weaker than for more mature systems.

Cryptography is one of the few effective defenses against all of this, since it provides ways to keep communications and data storage private. But strong cryptography is under continuous attack. Governments don't like the idea that individuals or companies or terrorists could have truly private communications, so there are frequent proposals to require backdoors into cryptographic mechanisms that would allow government agencies to break the encryption, though of course with "proper safeguards" and only "in the interests of national security." However well-intentioned, this is a bad idea, because weak cryptography helps your adversaries as well as your friends.

These are some of the problems and issues that ordinary people like the students in my course or the proverbial man or woman on the street have to worry about, no matter what their background and training.

The students in my course are not technical—no engineers, physicists or mathematicians. Instead they are English and politics majors, historians, classicists, economists, musicians and artists, a wonderful slice through the humanities and social sciences. By the end of the course these bright people should be able to read and understand a newspaper article about computing, to learn more from it, and perhaps to spot places where it might not be accurate. More broadly, I want my students and my readers to be intelligently skeptical about technology, to know that it is often a good thing but not a panacea; conversely, though it sometimes has bad effects, technology is not an unmitigated evil.

A fine book by Richard Muller called *Physics for Future Presidents* attempts to explain the scientific and technical background underlying major issues that leaders have to grapple with—nuclear threats, terrorists, energy, global warming, and the like. Well-informed citizens without aspirations to be president should know something of these topics as well. Muller's approach is a good metaphor for what I would like to accomplish: "Computing for Future Presidents."

What should a future president know about computing? What should a well-informed person know about computing? Everyone will have their own ideas; here are mine.

There are three core technical areas—hardware, software, and communications—and the book is organized around them.

Hardware is the tangible part, the computers we can see and touch, that sit in our homes and offices, and that we carry around in our phones. What's inside a computer, how does it work, how is it built? How does it store and process information? What are bits and bytes, and how do we use them to represent music, movies, and everything else?

Software, the instructions that tell computers what to do, is by contrast hardly tangible at all. What can we compute, and how fast can we compute it? How do we tell computers what to do? Why is it so hard to make them work right? Why are they so often hard to use?

Communications means computers, phones, and other devices talking to each other on our behalf and letting us talk to each other: the Internet, the Web, email and social networks. How do these work? The rewards are obvious, but what are the risks, especially to our privacy and security, and how can they be mitigated?

To this trio we should add *data*, which is all the information that hardware and software collect, store and process, and which communications systems send round the world. Some of this is data we contribute voluntarily, whether prudently or not, by uploading our words, pictures and videos. Some is personal information about us, usually gathered and shared without our knowledge, let alone agreement.

President or not, you should know about the world of computing because it affects you personally. No matter how non-technical your life and work, you're going to have to interact with technology and technical people. Knowing something of how devices and systems operate is a big advantage, even something as simple as recognizing when a salesperson or a help line is not telling you the whole truth. Indeed, ignorance can be directly harmful. If you don't understand viruses, phishing and similar threats, you become more susceptible to them. If you don't know how social networks leak, or even broadcast, information that you thought was private, you're likely to reveal much more than you realize. If you're not aware of the headlong rush by commercial interests to exploit what they have learned about your life, you're giving up privacy for little benefit. If you don't know why it's risky to do your personal banking in a coffee shop or an airport, you're vulnerable to theft of money and identity. And we ignore government encroachment on our personal privacy at our peril.

The book is meant to be read from front to back but you might prefer to skip ahead to topics of personal interest and come back later. For example, you could begin by reading about networks, cell phones, the Internet, the web and privacy issues starting in Chapter 8; you might have to look back at earlier chapters to understand a few parts, but mostly it will be accessible. You can skip anything quantitative, for instance how binary numbers work in Chapter 2, and ignore details of programming languages in a couple of chapters. The notes at the end list some books that I particularly like, and include links to sources and helpful supplements. A glossary gives brief definitions and explanations of important technical terms and acronyms.

Any book about computing can become dated quickly, and this one is no exception. The first edition was published well before we learned about the extent of NSA spying on individuals. I've updated the book with some of these important new stories, many of which relate to personal privacy and security, since that issue has changed significantly in the last few years. I've also tried to clarify explanations that were murky, and some dated material has been deleted or replaced. Nevertheless, some details will be wrong or out of date when you read this. As with the first edition, I've tried to ensure that things of lasting value are clearly identified; for the rest, check out the book's web site at `kernighan.com` for updates, corrections, extra material, and the like.

My goal for this book is that you will come away with some appreciation for an amazing technology and a real understanding of how it works, where it came from, and where it might be going in the future. Along the way, perhaps you'll pick up a helpful way of thinking about the world. I hope so.

Acknowledgments

I am again deeply indebted to friends and colleagues for their generous help and advice. As he did with the first edition, Jon Bentley read several drafts with meticulous care, providing helpful comments on every page; the book is much the better for his contributions. I also received valuable suggestions, criticisms and corrections on the whole manuscript from Swati Bhatt, Giovanni De Ferrari, Peter Grabowski, Gerard Holzmann, Vickie Kearn, Paul Kernighan, Eren Kursun, David Malan, David

Mauskop, Deepa Muralidhar, Madeleine Planeix-Crocker, Arnold Robbins, Howard Trickey, Janet Vertesi and John Wait. I have also benefited from helpful advice from David Dobkin, Alan Donovan, Andrew Judkis, Mark Kernighan, Elizabeth Linder, Jacqueline Mislow, Arvind Narayanan, Jonah Sinowitz, Peter Weinberger and Tony Wirth. The production team at Princeton University Press—Mark Bellis, Lorraine Doneker, Dimitri Karetnikov and Vickie Kearn—has been a pleasure to work with. My thanks to all of them.

I am also grateful to Princeton's Center for Information Technology Policy for good company, conversation, and weekly free lunches. And to the wonderful students of COS 109, whose talent and enthusiasm continue to amaze and inspire me, thank you.

Acknowledgments for the First Edition

I am deeply indebted to friends and colleagues for generous help and advice. In particular, Jon Bentley provided detailed comments on almost every page of several drafts. Clay Bavor, Dan Bentley, Hildo Biersma, Stu Feldman, Gerard Holzmann, Joshua Katz, Mark Kernighan, Meg Kernighan, Paul Kernighan, David Malan, Tali Moreshet, Jon Riecke, Mike Shih, Bjarne Stroustrup, Howard Trickey, and John Wait read complete drafts with great care, made many helpful suggestions, and saved me from some major gaffes. I also thank Jennifer Chen, Doug Clark, Steve Elgersma, Avi Flamholz, Henry Leitner, Michael Li, Hugh Lynch, Patrick McCormick, Jacqueline Mislow, Jonathan Rochelle, Corey Thompson, and Chris Van Wyk for valuable comments. I hope that they will recognize the many places where I took their advice, but not notice the few where I did not.

David Brailsford offered a great deal of helpful advice on self-publishing and text formatting, based on his own hard-won experience. Greg Doench and Greg Wilson were generous with advice about publishing. I am indebted to Gerard Holzmann and John Wait for photographs.

Harry Lewis was my host at Harvard during the 2010–2011 academic year, when the first few drafts of the book were written. Harry's advice and his experience teaching an analogous course were of great value, as were his comments on multiple drafts. Harvard's School of Engineering and Applied Sciences and the Berkman Center for Internet and Society provided office space and facilities, a friendly and stimulating environment, and (yes, there is such a thing!) regular free lunches.

I am especially grateful to the many hundreds of students who have taken COS 109, "Computers in our World." Their interest, enthusiasm and friendship have been a continual source of inspiration. I hope that when they are running the world a few years from now, they will have profited in some way from the course.

Understanding the Digital World

Introduction

"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke, Report on Planet 3, *Technology and the Future*, 1972.

My wife and I took a long vacation in the summer of 2015, nearly three months in England and France. We rented cars, bought train tickets, and reserved hotels in big cities and cottages in the middle of nowhere, all entirely through web sites. We used online maps and Google Street View to scope out neighborhoods and attractions before finalizing our lodgings. While we were away, we used mobile phones to navigate unfamiliar places, we kept in touch with friends and family by email and Skype, we sent frequent pictures and occasional movies, and I worked for a few hours almost every day on a book with my co-author in New York. I even checked mail once or twice a day while we were on a ship in the middle of the Atlantic.

To which you are probably thinking, "So what? Doesn't everybody?" And except maybe for the unusually long vacation and the ship, you're right. This is standard operating procedure in today's world. It's almost magical how easy and convenient it is to make arrangements without intermediaries and to keep in touch even when far away from home. These technological systems are so commonplace that we tend not to think about them, even though they have changed our lives remarkably quickly.

My wife and I didn't use Airbnb to find places to stay, though we could have. Airbnb was founded in 2008; it now operates in 190 countries and has one and a half million listings. Airbnb has had a major impact on the hotel industry in many cities—its prices are often lower and its use of technology sidesteps an established regulatory environment that has been slow to adapt.

We also didn't use Uber since we only took a couple of cab rides, but we could have (and our London cabbie moonlighted as an Uber driver). Uber was founded in 2009; it operates in over 60 countries. Uber is having a significant impact on the taxi industry in many cities—as with Airbnb, its prices are often lower, and its use of technology also skirts a regulatory environment that has been slow to adapt.

We didn't use WhatsApp to keep in touch because Skype was better for us, but we could have. WhatsApp was also founded in 2009, and was acquired by Facebook in 2014 for $19 billion. It's the largest instant messaging system for phones, with over 900 million users. Late in 2015 and again in May and July 2016, a judge in Brazil ordered WhatsApp to suspend its service, because it had refused to comply with a court order to turn over data that was part of a criminal investigation. An appeals court quickly reversed the order each time and 100 million Brazilian users went back to using WhatsApp instead of the established phone companies.

These stories, which are far from unique, remind us of the range of technology, how quickly it changes, how disruptive it can be, and how deeply it has become part of our lives, making them better in all kinds of ways.

But there's a darker side to the story, not so cheery and optimistic. For every kind of interaction I just mentioned, countless computer systems were quietly watching and remembering—who you and I dealt with, how much we paid, and where we were at the time. A large part of this data gathering is for commercial purposes, since the more that companies know about us, the more accurately they can target us for advertising. Most readers know that such data is collected, but I expect that many would be surprised by how much and how detailed it is.

And as we learned not long ago, companies are not the only observers.

The NSA emails, internal reports, and PowerPoint presentations disclosed by Edward Snowden revealed much about spying in the digital era. The gist is that the NSA spies on everyone on a grand scale. Fearing for his own safety, Snowden provided material very cautiously to a small number of journalists in Hong Kong, then fled to Moscow, where he remains under the protection of the Russian government. Variously called a traitor and a hero, he's likely to be there for a long time. The story of how he got the information and safely revealed it is told in Glenn Greenwald's 2014 book *No Place to Hide* and Laura Poitras's movie *Citizenfour*, which won the 2015 Oscar for best documentary film.

Snowden's revelations were stunning. It was widely suspected that the NSA spied on more people than it admitted, but the extent surpassed everyone's imagination. The NSA routinely recorded metadata about all telephone calls made in the US—who called whom, when they talked, and for how long—and may have recorded the content of these calls as well. It had recorded my Skype conversations and email contacts, and probably the mail contents as well. (Yours too, of course.) It was tapping the cell phones of world leaders. It was intercepting huge amounts of Internet traffic by placing recording devices on equipment at various sites. It had enlisted or coerced the major telecommunications and Internet companies to gather and hand over information about their users. It was storing great amounts of data for extended periods of time, sharing some of it with spy agencies in other countries.

Meanwhile, back on the commercial front, hardly a day goes by when we don't learn of another breach of security at some company or institution, in which shadowy hackers steal information like names, addresses, and credit card numbers of millions of people. Usually this is criminal theft, but sometimes it's espionage by other countries, looking for valuable information. Sometimes foolish behavior by whoever maintains the information accidentally exposes private data. No matter what the mechanism, data that has been collected about us is all too often exposed or stolen, potentially to be used against us.

So it's not all as wonderful and magical as one might think.

The purpose of this book is to explain the computing and communications technology that lies behind all of this, so you understand how such systems operate. How can pictures, music, movies, and intimate details of your personal life be sent around the world in no time at all? How do email and texting work, and how private are they? Why is spam so easy to send and so hard to get rid of? Do cell phones really report where you are all the time? How do iPhones and Android phones differ and why are they fundamentally exactly the same? Who is tracking you online and on your phone, and why does that matter? Can hackers take over your car? How about self-driving cars? Can you defend your privacy and security at all? By the end of the book, you should have a decent idea of how computer and communications systems work, how they affect you, and how you can strike a balance between using helpful services and protecting your privacy.

There are only a handful of fundamental ideas, which we will discuss in much more detail in the rest of the book.

First is the *universal digital representation of information*. Complex and sophisticated mechanical systems like those that stored documents, pictures, music and movies for much of the 20th century have been replaced by a single uniform mechanism. This is possible because information is represented digitally rather than in specialized forms like colored dyes embedded in plastic film or magnetic patterns on vinyl tape. Paper mail gives way to digital mail. Paper maps yield to digital ones. Paper documents are replaced by online databases. Different analog representations of information are replaced by a single digital representation.

Second is the *universal digital processor*. All of this information can be processed by a single general-purpose device, the digital computer. Digital computers that process the uniform digital representation of information have replaced complicated mechanical devices that process analog representations of information. As we'll see, computers are all equally capable in what they can compute, differing only in how fast they operate and how much data they can store. A smartphone is a computer of considerable sophistication, with as much computing power as a laptop from only a few years ago. Thus more and more of what might have been limited to desktop or laptop computers has found its way onto phones, and this process of convergence is accelerating.

Third is the *universal digital network*. The Internet connects the digital computers that process the digital representation; it connects computers and phones to mail, search, social networks, shopping, banking, news, entertainment, and everything else. You can exchange email with anyone, regardless of where they might be or how they choose to access their mail. You can search, comparison shop, and purchase from your phone, laptop, or tablet. Social networks keep you in touch with friends and family, again from phone or computer. There is great deal of infrastructure that makes all these services work.

An immense amount of *digital data* is also continuously being collected and analyzed. Maps, aerial photographs, and street-level views of much of the world are freely available. Search engines tirelessly scan the Internet so they can answer queries efficiently. Millions of books are available in digital form. Social networks and sharing sites maintain enormous amounts of data for and about us. Online stores and services provide access to their wares while quietly recording everything we do when we visit them, aided and abetted by search engines and social networks. For all of our online interactions, Internet service providers log the connections we make, and perhaps more. Governments spy on us all the time, to an extent that would have been unbelievable only a decade or two ago.

All of this is changing rapidly because digital technological systems are getting smaller, faster, and cheaper at an exponential rate: every year or two, things are twice as powerful for the same price, or cost half as much as they did. New phones with fancier features, better screens, and more interesting applications arrive continuously. New gadgets appear all the time; the most useful ones often find their functionality subsumed into phones over time. This is a natural by-product of digital technology, in which any technological development leads to improvement across the board for digital devices: if some change makes it possible to handle data cheaper, faster or in larger quantity, all devices will benefit. As a result, digital systems are pervasive, an integral part of our lives both visibly and behind the scenes.

This progress must surely seem wonderful, and indeed in most ways it is. But there are clouds around the silver lining. One of the most obvious and perhaps the most worrying to individuals is the impact of technology on personal privacy. When you use your phone to search for some product and then visit store web sites, all parties keep records of what you visited and what you clicked on. They know who you are because your phone identifies you uniquely. They know where you are because phones report their locations *all the time*. With GPS, the Global Positioning System, the phone company can locate you to within five to ten meters; even without GPS, they know your location to within a hundred meters or so. And they can sell that information. Physical stores are increasingly watching you as well. Face recognition technology may well be able to identify you on the street or in a store. Traffic cameras scan your license plates and know where your car is. The tracking that we permit today without even thinking about it makes the monitoring in George Orwell's *1984* look casual and superficial.

The records of what we do and where we do it may well live forever. Digital storage is so cheap and data is so valuable that information is rarely discarded. If you post something embarrassing online or send mail that you subsequently regret, it's too late. Information about you can be combined from multiple sources to create a detailed picture of your life, and is available to commercial, government and criminal interests without your knowledge or permission. It is likely to remain available indefinitely and could surface to embarrass you at any time in the future.

The universal network and its universal digital information have made us vulnerable to strangers to a degree never imagined a decade or two ago. As Bruce Schneier says in his excellent 2015 book, *Data and Goliath*, "Our privacy is under assault from constant surveillance. Understanding how this occurs is critical to understanding what's at stake."

The societal mechanisms that protect our privacy and our property have not kept up with the rapid advances in technology. Thirty years ago, I dealt with my local bank and other financial institutions by physical mail and occasional personal visits. Accessing my money took time and it left an extensive paper trail; it would have been difficult for someone to steal from me. Today, I deal with financial institutions mostly through the Internet. I can easily access my accounts, but unfortunately it's quite possible that through some blunder on my part or a screwup by one of these institutions, someone on the far side of the world could clean out my account, steal my identity, ruin my credit rating, and who knows what else, in no time at all and with little chance of recourse.

This book is about understanding how these systems work and how they are changing our lives. Of necessity it's a snapshot, so you can be certain that ten years from now, today's systems will seem clunky and antiquated. Technological change is not an isolated event but an ongoing process—rapid, continuous, and accelerating. Fortunately, the basic ideas of digital systems will remain the same, so if you understand those, you'll understand tomorrow's systems too, and you'll be in a better position to deal with the challenges and the opportunities that they present.

Part I

Hardware

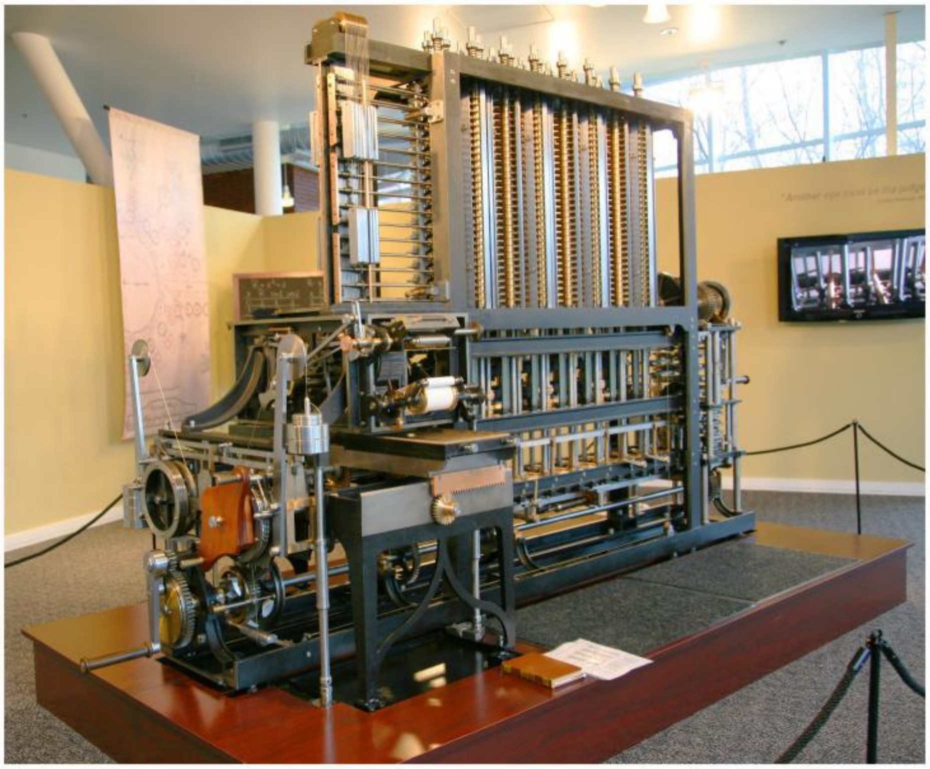
"I wish to God that this computation had been executed by steam."

Charles Babbage, 1821, quoted in Harry Wilmot Buxton,
Memoir of the Life and Labours of the Late Charles Babbage, 1872.

Hardware is the solid, visible part of computing: devices and equipment that you can see and put your hands on. The history of computing devices is interesting, but I will only mention a little of it here. Some trends are worth noting, however, especially the exponential increase in how much circuitry and how many devices can be packed into a given amount of space, often for a fixed price. As digital equipment has become more powerful and cheaper, widely disparate mechanical systems have been superseded by much more uniform electronic ones.

Computing machinery has a long history, though most early computational devices were specialized, often for predicting astronomical events and positions. For example, one theory holds that Stonehenge was an astronomical observatory, though the theory remains unproven. The Antikythera mechanism, from about 100 BCE, is an astronomical computer of remarkable mechanical sophistication. Arithmetic devices like the abacus have been used for millennia, especially in Asia. The slide rule was invented in the early 1600s, not long after John Napier's description of logarithms. I used one as an undergraduate engineer in the 1960s, but slide rules are now curiosities, replaced by calculators and computers, and my painfully acquired expertise is useless.

The most relevant precursor to today's computers is the Jacquard loom, which was invented in France by Joseph Marie Jacquard around 1800. The Jacquard loom used rectangular cards with multiple rows of holes that specified weaving patterns. The Jacquard loom thus could be "programmed" to weave a wide variety of different patterns under the control of instructions that were provided on punched cards; changing the cards caused a different pattern to be woven. The creation of labor-saving machines for weaving led to social disruption as weavers were put out of work; the Luddite movement in England in 1811–1816 was a violent protest against mechanization. Modern computing technology has similarly led to disruption.



Modern implementation of Babbage's Difference Engine.

Computing in today's sense began in England in the mid-19th century with the work of Charles Babbage. Babbage was a scientist who was interested in navigation and astronomy, both of which required tables of numeric values for computing positions. Babbage spent much of his life trying to build computing devices that would mechanize the tedious and error-prone arithmetic calculations that were performed by hand. You can sense his exasperation in the quotation above. For a variety of reasons, including alienating his financial backers, he never succeeded in his ambitions, but his designs were sound. Modern implementations of some of his machines, built with tools and materials from his time, can be seen in the Science Museum in London and the Computer History Museum in Mountain View, California (in the figure).

Babbage encouraged a young woman, Augusta Ada Byron, the daughter of the poet George Gordon, Lord Byron, and later Countess of Lovelace, in her interests in mathematics and his computational devices. Lovelace wrote detailed descriptions of how to use Babbage's Analytical Engine (the most advanced of his planned devices) for scientific computation and speculated that machines could do non-numeric computation as well, such as composing music. "Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." Ada Lovelace is often called the world's first programmer, and the Ada programming language is named in her honor.

Ada Lovelace. Detail from a portrait by Margaret Sarah Carpenter (1836).

Herman Hollerith, working with the US Census Bureau in the late 1800s, designed and built machines that could tabulate census information far more rapidly than could be done by hand. Using ideas from the Jacquard loom, Hollerith used holes punched in paper cards to encode census data in a form that could be processed by his machines; famously, the 1880 census had taken eight years to tabulate, but with Hollerith's punch cards and tabulating machines, the 1890 census took only one year. Hollerith founded a company that in 1924 became, through mergers and acquisitions, International Business Machines, which we know today as IBM.

Babbage's machines were complex mechanical assemblies of gears, wheels, levers and rods. The development of electronics in the 20th century made it possible to imagine computers that did not rely on mechanical components. The first significant one of these all-electronic machines was ENIAC, the Electronic Numerical Integrator and Computer, which was built during the 1940s at the University of Pennsylvania in Philadelphia, by Presper Eckert and John Mauchly. ENIAC occupied a large room and required a large amount of electric power; it could do about 5,000 additions in a second. It was intended to be used for ballistics computations and the like, but it was not completed until 1946 when World War II was long over. (Parts of ENIAC are on display in the Moore School of Engineering at the University of Pennsylvania.)

Babbage saw clearly that a computing device could store its operating instructions and its data in the same form, but ENIAC did not store instructions in memory along with data; instead it was programmed by setting up connections through switches and re-cabling. The first computers that truly stored programs and data together were built in England, notably EDSAC, the Electronic Delay Storage Automatic Calculator, in 1949.

Early electronic computers used vacuum tubes as computing elements. Vacuum tubes are electronic devices roughly the size and shape of a cylindrical light bulb (see Figure 1.6); they were expensive, fragile, bulky, and power hungry. With the invention of the transistor in 1947, and then of integrated circuits in 1958, the modern era of computing really began. These technologies have allowed electronic systems to become smaller, cheaper and faster.

The next three chapters describe computer hardware, focusing on the logical architecture of computing systems more than on the physical details of how they are built. The architecture has been largely unchanged for decades, while the hardware has changed to an astonishing degree. The first chapter is an overview of the components of a computer. The second chapter shows how computers represent information with bits, bytes and binary numbers. The third chapter explains how computers actually compute: how they process the bits and bytes to make things happen.

1

What's in a Computer?

"Inasmuch as the completed device will be a general-purpose computing machine it should contain certain main organs relating to arithmetic, memory-storage, control and connection with the human operator."

Arthur W. Burks, Herman H. Goldstine, John von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.

Let's begin our discussion of hardware with an overview of what's inside a computer. We can look at a computer from at least two viewpoints: the logical or functional organization—what the pieces are, what they do and how they are connected—and the physical structure—what the pieces look like and how they are built. The goal of this chapter is to see what's inside, learn roughly what each part does, and get some sense of what the myriad acronyms and numbers mean.

Think about your own computing devices. Many readers will have some kind of "PC," that is, a laptop or desktop computer descended from the Personal Computer that IBM first sold in 1981, running some version of the Windows operating system from Microsoft. Others will have an Apple Macintosh that runs a version of the Mac OS X operating system. Still others might have a Chromebook or similar laptop that relies on the Internet for storage and computation. More specialized devices like smartphones, tablets and ebook readers are also powerful computers. These all look different and when you use them they feel different as well, but underneath the skin, they are fundamentally the same. We'll talk about why.

There's a loose analogy to cars. Functionally, cars have been the same for over a hundred years. A car has an engine that uses some kind of fuel to make the engine run and the car move. It has a steering wheel that the driver uses to control the car. There are places to store the fuel and places to store the passengers and their goods. Physically, however, cars have changed greatly over a century: they are made of different materials, and they are faster, safer, and much more reliable and comfortable. There's a world of difference between my first car, a well-used 1959 Volkswagen Beetle, and a Ferrari, but either one will carry me and my groceries home from the store or across the country, and in that sense they are functionally the same. (For the record, I have never even sat in a Ferrari, let alone owned one, so I'm speculating about whether there's room for the groceries.)

The same is true of computers. Logically, today's computers are very similar to those of the 1950s, but the physical differences go far beyond the kinds of changes that have occurred with the automobile. Today's computers are much smaller, cheaper, faster and more reliable than those of 50 years ago, literally a million times better in some properties. Such improvements are the fundamental reason why computers are so pervasive.

The distinction between the functional behavior of something and its physical properties—the difference between what it does and how it's built or works inside—is an important idea. For computers, the "how it's built" part changes at an amazing rate, as does how fast it runs, but the "how it does what it does" part is quite stable. This distinction between an abstract description and a concrete implementation will come up repeatedly in what follows.

I sometimes do a survey in my class in the first lecture. How many have a PC? How many have a Mac? The ratio was fairly constant at 10 to 1 in favor of PCs in the first half of the 2000s, but changed rapidly over a few years, to the point where Macs now account for well over three quarters of the computers. This is not typical of the world at large, however, where PCs dominate by a wide margin.

Is the ratio unbalanced because one is superior to the other? If so, what changed so dramatically in such a short time? I ask my students which kind is better, and for objective criteria on which to base that opinion. What led you to your choice when you bought your computer?

Naturally, price is one answer. PCs tend to be cheaper, the result of fierce competition in a marketplace with many suppliers. A wider range of hardware add-ons, more software, and more expertise are all readily available. This is an example of what economists call a *network effect*: the more other people use something, the more useful it will be for you, roughly in proportion to how many others there are.

On the Mac side are perceived reliability, quality, esthetics, and a sense that "things just work," for which many consumers are willing to pay a premium.

The debate goes on, with neither side convincing the other, but it raises some good questions and helps to get people thinking about what is different between different kinds of computing devices and what is really the same.

There's an analogous debate about phones. Almost everyone has a "smart phone" that can run programs ("apps") downloaded from Apple's App Store or Google's Play Store. The phone serves as a browser, a mail system, a watch, a camera, a music and video player, a comparison shopping tool, and even occasionally a device for conversation. Typically about three quarters of the students have an iPhone from Apple; almost all the rest have an Android phone from one of many suppliers. A tiny fraction might have a Windows phone, and (rarely) someone admits to having only a "feature phone," which is defined as a phone that has no features beyond the ability to make phone calls. My sample is for the US and a comparatively affluent environment; in other parts of the world, Android phones would be much more common.

Again, people have good reasons—functional, economic, esthetic—for choosing one kind of phone over others but underneath, just as for PCs versus Macs, the hardware that does the computing is very similar. Let's look at why.

1.1 Logical Construction

If we were to draw an abstract picture of what's in a simple generic computer—its logical or functional architecture—it would look like the diagram in Figure 1.1 for both Mac and PC: a processor (the CPU), some primary memory (RAM), some secondary storage (a disk) and a variety of other components, all connected by a set of wires called a bus that transmits information between them.

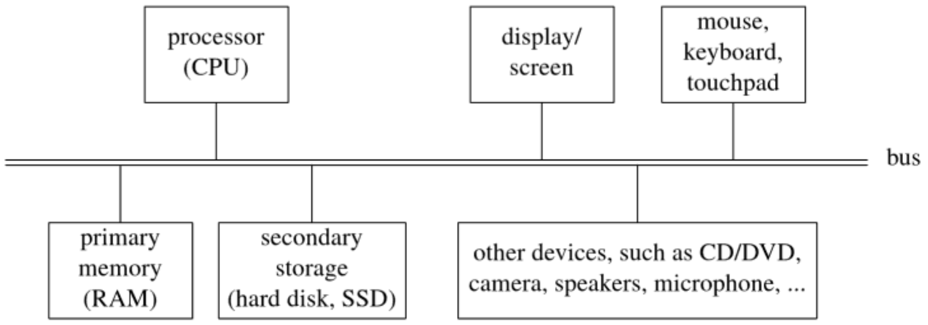

processor (CPU)
display/ screen
mouse, keyboard, touchpad
bus
primary memory (RAM)
secondary storage (hard disk, SSD)
other devices, such as CD/DVD, camera, speakers, microphone, ...

Figure 1.1: Architectural diagram of a simple idealized computer.

If instead we drew this picture for a phone or tablet, it would be similar, though mouse, keyboard and display are combined into one component, the screen. There's certainly no CD or DVD, but there are hidden components like a compass, an accelerometer, and a GPS receiver for determining your physical location.

The basic organization—with a processor, storage for instructions and data, and input and output devices—has been standard since the 1940s. It's often called the *von Neumann architecture*, after John von Neumann, who described it in the 1946 paper quoted above. Though there is still debate over whether von Neumann gets too much credit for work done by others, the paper is so clear and insightful that it is well worth reading even today. For example, the quotation at the beginning of this chapter is the first sentence of the paper. Translated into today's terminology, the CPU provides arithmetic and control, the RAM and disk are memory storage, and the keyboard, mouse and display interact with the human operator.

1.1.1 CPU

The processor or *central processing unit* (*CPU*) is the brain, if a computer could be said to have such a thing. The CPU does arithmetic, moves data around, and controls the operation of the other components. The CPU has a limited repertoire of basic operations that it can perform but it performs them blazingly fast, billions per second. It can decide what operations to perform next based on the results of previous computations, so it is to a considerable degree independent of its human users. We will spend more time on this component in Chapter 3 because it's so important.

If you go to a store or shop online to buy a computer, you'll find most of these components mentioned, usually accompanied by mysterious acronyms and equally mysterious numbers. For example, you might see a CPU described as a "2.2 GHz dual-core Intel Core i7 processor," as it is for one of my computers. What's that?

Intel makes the CPU and "Core i7" is just a marketing term. This particular processor actually has two processing units in a single package; in this context, lower-case "core" has become a synonym for "processor." For most purposes, it's sufficient to think of the combination as "the CPU," no matter how many cores it has.

"2.2 GHz" is the more interesting part. CPU speed is measured, at least approximately, in terms of the number of operations or instructions or parts thereof that it can do in a second. The CPU uses an internal clock, rather like a heartbeat or the ticking of a clock, to step through its basic operations. One measure of speed is the number of such ticks per second. One beat or tick per second is called one *hertz* (abbreviated *Hz*), after the German engineer Heinrich Hertz, whose discovery of how to produce electromagnetic radiation in 1888 led directly to radio and other wireless systems. Radio stations give their broadcast frequencies in megahertz (millions of hertz), like 102.3 MHz. Computers today typically run in the billions of hertz, or gigahertz, or GHz; my quite ordinary 2.2 GHz processor is zipping along at 2,200,000,000 ticks per second. The human heartbeat is about 1 Hz or almost 100,000 beats per day, which is around 30 million per year, so my CPU does in 1 second the number of beats my heart would do in 70 years.

This is our first encounter with some of the numerical prefixes like mega and giga that are so common in computing. "Mega" is one million, or 10^6; "giga" is one billion, or 10^9, and usually pronounced with a hard "g" as in "gig." We'll see more units soon enough, and there is a complete table in the glossary.

1.1.2 RAM

The primary memory or *random access memory* (*RAM*) stores information that is in active use by the processor and other parts of the computer; its contents can be changed by the CPU. The RAM stores not only the data that the CPU is currently working on, but also the instructions that tell the CPU what to do with the data. This is a crucially important point: by loading different instructions into memory, we can make the CPU do a different computation. This makes the stored-program computer a general-purpose device; the same computer can run a word processor and a spreadsheet, surf the web, send and receive email, keep up with friends on Facebook, do my taxes, and play music, all by placing suitable instructions in the RAM. The importance of the stored-program idea cannot be overstated.

RAM provides a place to store information while the computer is running. It stores the instructions of programs that are currently active, like Word, Photoshop or a browser. It stores their data—the pictures on the screen, the documents being edited, the music that's currently playing. It also stores the instructions of the operating system—Windows, Mac OS X or something else—that operates behind the scenes to let you run multiple applications at the same time. We'll talk about operating systems in Chapter 6.

RAM is called *random access* because the CPU can access the information stored at any place within it as quickly as in any other; to over-simplify a little, there's no speed penalty for accessing memory locations in a random order. Compare this to an old VCR tape, where to look at the final scenes of a movie, you have to fast forward (slowly!) over everything from the beginning; that's called *sequential access*.

Most RAM is *volatile*, that is, its contents disappear if the power is turned off, and all this currently active information is lost. That's why it's prudent to save your work often, especially on a desktop machine, where tripping over the power cord could be a real disaster.

Your computer has a fixed amount of RAM. Capacity is measured in bytes, where a *byte* is an amount of memory that's big enough to hold a single character like `W` or `@`, or a small number like 42, or a part of a larger value. Chapter 2 will show how information is represented in memory and other parts of a computer, since it's one of the fundamental issues in computing. But for now, you can think of the RAM as a large collection of identical little boxes, numbered from 1 up to a few billion, each of which can hold a small amount of information.

What is the capacity? The laptop I'm using right now has 4 billion bytes or 4 gigabytes or 4 GB of RAM, which many people would deem too small. The reason is that more RAM usually translates into faster computing, since there's never enough for all the programs that want to use it at the same time, and it takes time to move parts of an unused program out to make room for something new. If you want your computer to run faster, buying extra RAM is likely to be the best strategy.

1.1.3 Disks and other secondary storage

The RAM has a large but limited capacity to store information and its contents disappear when the power is turned off. *Secondary storage* holds information even when the power is turned off. There are two main kinds of secondary storage, the magnetic disk, usually called the *hard disk* or *hard drive*, and flash memory, often called *solid state disk*. Both kinds of disk store much more information than RAM and it's not volatile: information on disk stays there indefinitely, power or not. Data, instructions, and everything else is stored on the disk for the long term and brought into RAM only transiently.

Magnetic disks store information by setting the direction of magnetization of tiny regions of magnetic material on rotating metallic surfaces. Data is stored in concentric tracks that are read and written by a sensor that moves from track to track. The whirring and clicking that you hear when a computer is doing something is the disk in action, moving the sensor to the right places on the surface. You can see the surface and sensor in the picture of a standard laptop disk in Figure 1.2; the platter is 2½ inches (6¼ cm) in diameter.

Disk space is about 100 times cheaper per byte than RAM, but accessing information is slower. It takes about ten milliseconds for the disk drive to access any particular track on the surface; data is then transferred at roughly 100 MB per second.

Increasingly, laptops have solid state disks, which use flash memory instead of rotating machinery. Flash memory is non-volatile; information is stored as electric charges in circuitry that maintains the charge in individual circuit elements without using any power. Stored charges can be read to see what their values are, and they can be erased and overwritten with new values. Flash memory is faster, lighter, more reliable, won't break if dropped, and requires less power than conventional disk storage, so it's used in cell phones, cameras, and the like. Right now it's more expensive per byte but prices are coming down fast and it seems likely to take over from

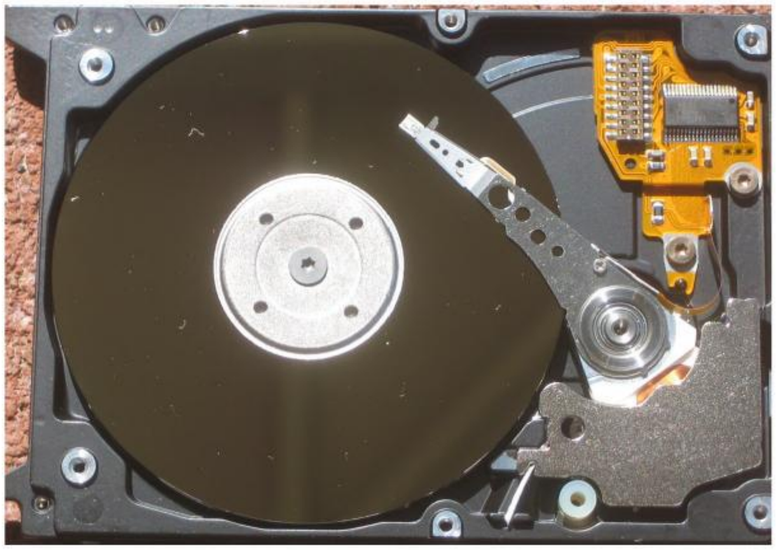


Figure 1.2: Inside a hard disk drive.

mechanical disks in laptops.

A typical laptop disk today holds perhaps 500 gigabytes, and external drives that can be plugged in to a USB socket have capacities in the multi-terabyte (TB) range. "Tera" is one trillion, or 10^{12}, another unit that you'll see more and more often.

How big is a terabyte, or even a gigabyte for that matter? One byte holds one alphabetic character in the most common representation of English text. *Pride and Prejudice*, about 250 pages on paper, has about 550,000 characters, so 1 GB could hold nearly 2,000 copies of it. More likely, I would store one copy and then include some music. Music in MP3 or AAC format is about 1 MB per minute, so an MP3 version of one of my favorite audio CDs, *The Jane Austen Songbook*, is about 60 MB, and there would still be room for another 15 hours of music in 1 GB. The two-disk DVD of the 1995 BBC production of *Pride and Prejudice* with Jennifer Ehle and Colin Firth is less than 10 GB, so I could store it and a hundred similar movies on a 1 TB disk.

A disk is a good example of the difference between logical structure and physical implementation. When we run a program like Explorer in Windows or Finder in Mac OS X, we see disk contents organized as a hierarchy of folders and files. But the data could be stored on rotating machinery, integrated circuits with no moving parts, or something else entirely. The particular kind of "disk" in a computer doesn't matter. Hardware in the disk itself and software in the operating system, called the file system, create the organizational structure. We will return to this in Chapter 6.

The logical organization is so well matched to people (or, more likely, by now we're so completely used to it) that other devices provide the same organization even though they use completely different physical means to achieve it. For example, the software that gives you access to information from a CD-ROM or DVD makes it look like this information is stored in a file hierarchy, regardless of how it is actually stored. So do USB devices, cameras and other gadgets that use removable memory cards. Even the venerable floppy disk, now totally obsolete, looked the same at the logical level. This is a good example of *abstraction*, a pervasive idea in computing: physical implementation details are hidden. In the file system case, no matter how the different technologies work, they are presented to users as a hierarchy of organized information.

1.1.4 Et cetera

Myriad other devices serve special functions. Mice, keyboards, touch screens, microphones, cameras and scanners all allow users to provide input. Displays, printers and speakers output to users. Networking components like Wi-Fi or Bluetooth communicate with other computers.

The architecture drawing shows these as if they were all connected by a set of wires called a *bus*, a term borrowed from electrical engineering. In reality, there are multiple buses inside a computer, with properties appropriate to their function—short, fast, but expensive between CPU and RAM; long, slow, but cheap to the headphone jack. Some of the buses make an appearance outside as well, like the ubiquitous Universal Serial Bus or *USB* for plugging devices into a computer.

We won't spend much time on other devices at the moment, though we'll occasionally mention them in some specific context. For now, try to list the different devices that might accompany your computer or be attached to it: mice, keyboards, touch pads and touch screens, displays, printers, scanners, game controllers, music players, headphones, speakers, microphones, cameras, phones, connections to other computers. The list goes on and on. All of these have gone through the same evolution as processors, memory, and disks: the physical properties have changed rapidly, often towards more capabilities in a smaller package at a lower price.

It's also worth noting how these devices are converging into a single one. Cell phones now serve as watches, calculators, still and video cameras, music and movie players, game consoles, barcode readers, navigators, and even flashlights. A smartphone has the same abstract architecture as a laptop, though with major implementation differences due to size and power constraints. Phones don't have hard disks like the one shown in Figure 1.2, but they do have flash memory to store information—address books, pictures, apps, and the like—when the phone is turned off. They don't have many external devices either, though there's likely Bluetooth, a socket for headphones and an external microphone, and a USB connector. Tiny cameras are so cheap that most phones have one on each side. Tablets like the iPad and its competitors occupy another position in the space of possibilities; they too are computers with the same general architecture and similar components.

1.2 Physical Construction

In class, I pass around a variety of hardware devices (the legacy of several decades of dumpster diving), typically with their innards exposed. So many things in computing are abstract that it's helpful to be able to see and touch disks, integrated circuit chips, the wafers on which they are manufactured, and so on. It's also interesting to see the evolution of some of these devices. For example, a laptop disk today is indistinguishable from one a decade old; the newer one is likely to have 10 or 100 times the capacity but the improvement is invisible. The same is true of SD cards like those used in digital cameras. Today's packages are identical to those of a few years ago (Figure 1.3), but the capacity is much higher and the price is lower; that 32 GB card costs less than 10 dollars.

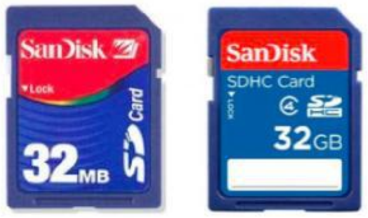


Figure 1.3: SD cards of very different capacities.

On the other hand, there's a clear progression in the circuit boards that hold the components of a computer; there are fewer components today because more of the circuitry is inside them, the wiring is finer, and the connecting pins are more numerous and much closer together than they were 20 years ago.

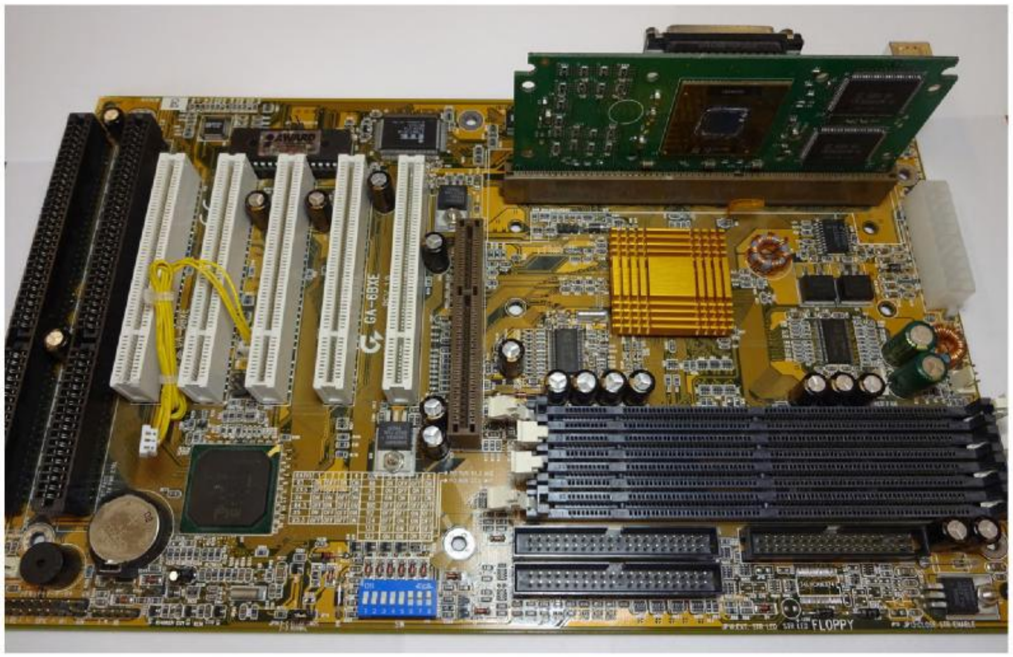


Figure 1.4: PC circuit board, circa 1998; 12 x 7.5 inches (30 x 19 cm).

Figure 1.4 shows a desktop PC circuit board from the late 1990s. The components like CPU and RAM are mounted on or plugged into this board, and are connected by wires printed on the other side. Figure 1.5 shows part of the back side of the circuit board in Figure 1.4; the parallel printed wires are buses of various sorts.

for networking. All of these have had dramatic improvements in size, capacity and cost over the past 50 years.

1.3 Moore's Law

In 1965, Gordon Moore, later the co-founder and long-time CEO of Intel, published a short article entitled "Cramming more components onto integrated circuits." Extrapolating from a very few data points, Moore observed that as technology improved, the number of transistors that could be manufactured on an integrated circuit of a particular size was doubling approximately every year, a rate that he later revised to every two years, and others have set at 18 months. Since the number of transistors is a rough surrogate for computing power, this meant that computing power was doubling every two years, if not faster. In 20 years there would be ten doublings and the number of devices would have increased by a factor of 2^{10}, that is, about one thousand. In forty years, the factor is a million or more.

This exponential growth, now known as *Moore's Law*, has been going on for over fifty years, so integrated circuits now have well over a million times as many transistors as they did in 1965. Graphs of Moore's Law in action, especially for processor chips, show the number of transistors rising from a couple of thousand in Intel's 8008 CPU in the early 1970s to a billion or more in the processors in inexpensive consumer laptops today.

The number that characterizes the scale of circuitry is the size of individual features on an integrated circuit, for example, the width of a wire or the active part of a transistor. This number has been shrinking steadily for many years. The first (and only) integrated circuit I designed in 1980 used 3.5 micron (3.5 micrometer) features. For many ICs today, that minimum feature size is 14 nanometers, that is, 14 billionths of a meter, and the next step will be 10 nanometers. "Micro" is one millionth, or 10^{-6}. "Milli" is one thousandth, or 10^{-3}. "Nano" is one billionth, or 10^{-9}, and nanometer is abbreviated nm. For comparison, a sheet of paper or a human hair is about 100 micrometers or 1/10th of a millimeter thick.

The design and manufacture of integrated circuits is an extremely sophisticated business, and highly competitive. Manufacturing operations ("fabrication lines") are expensive as well; a new factory can easily cost several billion dollars. A company that can't keep up technically and financially is at a serious competitive disadvantage, and a country that doesn't have such resources must depend on others for its technology, potentially a serious strategic problem.

It should be noted that Moore's Law is not a law of nature, but a guideline that the semiconductor industry has used to set targets. At some point the law will stop working. Its limits have often been predicted in the past, though ways around them have been found so far. We are getting to the point where there are only a handful of individual atoms in some circuits, however, and that's too small to control. CPU speeds no longer double every couple of years, in part because faster chips generate too much heat, but RAM capacity still does increase; meanwhile, processors can use more transistors by placing more than one CPU on a chip and often have multiple processor chips.

It's striking to compare a personal computer of today to the original IBM PC, which appeared in 1981. That PC had a 4.77 MHz processor; the clock rate in a 2.2 GHz CPU is nearly 500 times faster. It had 64 kilobytes of RAM; today's 8 GB computers have 125,000 times as much. ("Kilo" is one thousand, abbreviated "K".) The original had at most 750 KB of floppy disk storage and no hard disk; today's laptops are creeping up on a million times as much disk space. It had an 11-inch screen that could only display 24 rows of 80 green letters on a black background; I wrote a lot of this book on a 24-inch screen with 16 million colors. A PC with 64 KB of memory and a single 160 KB floppy disk cost $3,000 in 1981 dollars, which now might be equivalent to $5,000 to $10,000; today a laptop with a 2 GHz processor, 4 GB of RAM, and a 500 GB disk costs two or three hundred dollars.

1.4 Summary

Computer hardware, indeed digital hardware of all sorts, has been improving exponentially for over fifty years, starting with the invention of the integrated circuit. The word "exponential" is often misunderstood and misused, but in this case it's accurate; over every fixed period of time, circuits have gotten smaller or cheaper or more capable by a given percentage. The simplest version is Moore's Law: every 18 months or so the number of devices that can be put on an integrated circuit of a given size approximately doubles. This tremendous growth in capabilities is at the heart of the digital revolution that has changed our lives so much.

The architecture of a computer—what the pieces are, what they do, and how they are connected to each other—has not changed since the 1940s. If von Neumann were to come back and examine one of today's computers, I conjecture that he would be stunned by modern hardware but would find the architecture completely familiar.

That similarity of structure applies even more broadly. One of the great insights of 20th century computer science is that the logical or functional properties of today's digital computers, the original PC, its physically much bigger but less powerful ancestors, and our ubiquitous phones are all the same. If we ignore practicalities like speed and storage capacity, they all can compute exactly the same things. Thus, improvements in hardware make a great practical difference in what we can compute, but surprisingly do not of themselves make any fundamental change in what could be computed in principle. We'll talk more about this in Chapter 3.

2

Bits, Bytes, and Representation of Information

"If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey."

Claude Shannon, *A Mathematical Theory of Information*, 1948.

In this chapter we're going to discuss three fundamental ideas about how computers represent information.

First, *computers are digital processors*: they store and process information that comes in discrete chunks and takes on discrete values—basically just numbers. By contrast, analog information implies smoothly varying values.

Second, *computers represent information in bits*. A *bit* is a binary digit, that is, a number that is either 0 or 1. Everything inside the computer is represented with bits instead of the familiar decimal numbers that people use.

Third, *groups of bits represent larger things*. Numbers, letters, words, names, sounds, pictures, movies, and the instructions that make up the programs that process them—all of these are represented as groups of bits.

You can safely skip the numeric details in this chapter, but the ideas are important.

2.1 Analog versus Digital

Let's distinguish between analog and digital. "Analog" comes from the same root as "analogous," and is meant to convey the idea of values that change smoothly as something else changes. Much of what we deal with in the real world is analog, like a water tap or the steering wheel of a car. If you want to turn the car a little, you turn the wheel a little; you can make as small an adjustment as you like. Compare this to the turn signal, which is either on or off; there's no middle ground. In an analog device, something (how much the car turns) varies smoothly and continuously in proportion to a change in something else (how much you turn the steering wheel). There are no discrete steps; a small change in one thing implies a small change in another.

Digital systems deal with discrete values, so there are only a fixed number of possible values: the turn signal is either off or it's on in one direction or the other. A small change in something results either in no change or in a sudden change in something else, from one of its discrete values to another.

Think about a watch. Analog watches have an hour hand, a minute hand, and a second hand that goes around once a minute. Although modern watches are controlled by digital circuitry inside, the hour and minute hands move smoothly as time passes and go through every possible position. By contrast, a digital watch or a cell phone clock displays time with digits. The display changes every second, a new minute value appears every minute, and there's never a fractional second.

Think about a car speedometer. My car has a traditional analog speedometer, where a needle moves smoothly in direct proportion to the car's speed. The transitions from one speed to another are smooth and there's no break. But it also has a digital display that shows speed to the nearest mile or kilometer per hour. Go a tiny bit faster and the display goes from 65 to 66; go a tiny bit slower and it drops back to 65. There's never a display of 65.5.

Think about a thermometer. The kind with a column of red liquid (colored alcohol, usually) or mercury is analog: the liquid expands or contracts in direct proportion to temperature changes, so a small change in temperature will produce a similarly small change in the height of the column. But the sign that flashes `37` outside a building is digital: the display is numeric, and it shows 37 for all temperatures between 36½ and 37½.

This can lead to some odd situations. Some years ago, I was listening to my car radio on a US highway within reception distance of Canada, which uses the metric system. The announcer, trying to be helpful to everyone in his audience, said "the Fahrenheit temperature has gone up one degree in the last hour; the Celsius temperature is unchanged."

Why digital instead of analog? After all, our world is analog, and analog devices like watches and speedometers are easier to interpret at a glance. Nevertheless, much modern technology is digital; in many ways, that's the story in this book. Data from the external world—sound, images, movement, temperature, and everything else—is converted as soon as possible to a digital form on the input side, and is converted back to analog form as late as possible on the output side. The reason is that digital data is easy for computers to work with. It can be stored, transported, and processed in many ways regardless of its original source. As we'll see in Chapter 8, digital information can be compressed by squeezing out redundant or unimportant information. It can be encrypted for security and privacy, it can be merged with other data, it can be copied exactly, it can be shipped anywhere via the Internet, and it can be stored in an endless variety of devices. Most of this is infeasible or even impossible with analog information.

Digital systems have another advantage over analog: they are much more easily extended. In stopwatch mode, my digital watch can display elapsed times to a hundredth of a second; adding that capability to an analog watch would be challenging. On the other hand, analog systems sometimes have the advantage: old media like clay tablets, stone carvings, parchment, paper and photographic film have all stood the test of time in a way that digital forms may not.

2.2 Analog-Digital Conversion

How do we convert analog information into digital form? Let's look at some of the basic examples, beginning with pictures and music, which between them illustrate the most important ideas.

Conversion of images to digital form is probably the easiest way to visualize the process. Suppose we take a picture of the family cat (Figure 2.1).

Figure 2.1: The family cat.

An analog camera creates an image by exposing a light-sensitive area of chemical-coated plastic film to light from the object being photographed. Different areas receive different amounts of light of different colors, and that affects dyes in the film. The film is developed and printed on paper through a complicated sequence of chemical processes; the colors are displayed as varying amounts of colored dyes.

In a digital camera, the lens focuses the image onto a rectangular array of tiny light-sensitive detectors that lie behind red, green and blue filters. Each detector stores an amount of electric charge that is proportional to the amount of light that falls on it. These charges are converted into numeric values and the digital representation of the picture is the sequence of resulting numbers that represent the light intensities. If the detectors are smaller and more numerous and the charges are measured more precisely, then the digitized image will capture the original more accurately.

Each element of the sensor array is a set of detectors that measure the amount of red, green and blue light; each group is called a *pixel*, for picture element. If the image is 3,000 by 2,000 pixels, that's six million picture elements, or six megapixels, small for today's digital cameras. The color of a pixel is usually represented by three values that record the intensities of red, green and blue that it contains, so a six

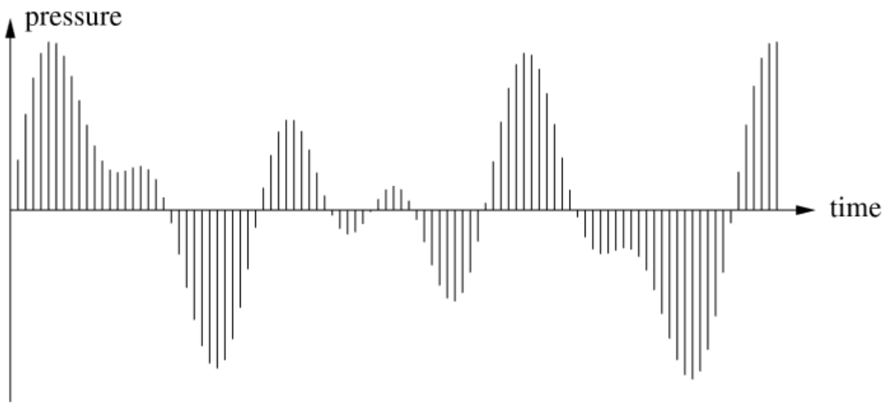


Figure 2.5: Sampling a sound waveform.

CD sound quality is not as good as LPs.

The audio compact disc or CD appeared around 1982, and was the first consumer example of digital sound. Rather than the analog groove of an LP record, a CD records *numbers* in a long spiral track on one side of the disk. The surface at each point along the track either is smooth or has a tiny pit. These pitted or smooth spots are used to encode the numeric values of the wave; each spot is a single bit, and a sequence of bits represents the numeric value in a binary encoding, as we will discuss in the next section. As the disk rotates, a laser shines on the track and a photoelectric sensor detects changes in how much light is reflected. If there's not much light, there was a pit; if there's a lot of reflected light, there was no pit. The standard encoding for CDs uses 44,100 samples per second; each sample is two values (left side and right side, for stereo) of amplitudes measured to an accuracy of one part in 65,536, which is 2^{16}. The pits are so small that they can only be seen with a microscope. DVDs are similar; smaller spots and a shorter wavelength laser allow them to store nearly 5 GB, compared to about 700 MB for a CD.

The audio CD almost drove the LP out of existence because it was so much better in most ways—not subject to wear because there is no physical contact from the laser, not much bothered by dirt or scratches, not fragile, and definitely compact. LPs periodically enjoy a modest renaissance, while CDs of popular music are in serious decline because it's easier and cheaper to download music from the Internet. CDs also had a second career as a storage and distribution medium for software and data, but that was superseded by DVDs, which in turn have largely been replaced by downloading. To many readers, audio CDs may already seem as antique as LPs, though I personally am happy that my music collection is entirely on CDs. I own them outright, which is not true of music collections "in the cloud." And manufactured CDs will outlast me, though copied ones may not, because they rely on a chemical change in a light-sensitive dye whose properties may change over time.

Because they contain more detail than humans can actually perceive, sound and images can be *compressed*. For music, this is typically done with compression techniques like MP3 and AAC (Advanced Audio Compression), which reduce the size by a factor of 10 with very little perceptible degradation. For images, the most common compression technique is called JPEG, named after the Joint Photographic Experts Group, the organization that defined it; it also shrinks an image by a factor of 10 or more. Compression is an example of the kind of processing that can be done on digital information but would be extremely difficult if not impossible with analog. We'll discuss it further in Chapter 9.

What about movies? In the 1870s, the English photographer Eadweard Muybridge showed how to create the illusion of motion by displaying a sequence of still images one after another in rapid succession. Today, a motion picture displays images at 24 frames per second, and TV displays at 25 to 30 frames per second, which is fast enough that the human eye perceives the sequence as continuous motion. Old movies used only a dozen frames per second so they had noticeable flicker; that artifact lives on in the old word "flicks" for movies and today in the name Netflix.

A digital representation of a movie combines and synchronizes the sound and picture components. Compression can be used to reduce the amount of space required as in standard movie representations like MPEG ("Moving Picture Experts Group"). In practice, video representation is more complicated than audio, in part because it's intrinsically harder, but also because much of it is based on standards for broadcast television, which for most of its lifetime has been analog. Analog television is being phased out in most parts of the world. In the US, television broadcasting switched to digital in 2009; other countries are in various stages of the process.

Movies and television shows are a combination of pictures and sound, and commercial ones cost much more to produce than music does. Yet it's just as easy to make perfect digital copies and send them around the world for free. So the copyright stakes are higher than for music, and the entertainment industry continues its campaign against copying.

Some kinds of information are easy to represent in digital form, since no transformation is needed beyond agreement on what the representation is. Consider ordinary text, like the letters, numbers and punctuation in this book. We could assign a unique number to each different letter—`A` is 1, `B` is 2, and so on—and that would be a fine digital representation. In fact, that's exactly what is done, except that in the standard representation, `A` through `Z` are 65 through 90, `a` through `z` are 97 through 122, the digits `0` through `9` are 48 through 57, and other characters like punctuation take other values. This representation is called *ASCII*, which stands for the American Standard Code for Information Interchange.

Figure 2.6 shows part of ASCII; I've omitted the first four rows, which contain tab, backspace and other non-printing characters.

There are multiple character-set standards in different geographic or linguistic regions, but the world has more or less converged on a single standard called Unicode, which specifies a unique numeric value for every character in every language. This is a big collection, since humans have been endlessly inventive but rarely systematic in their creation of writing systems. Unicode has over 120,000 characters at this point and the number rises steadily. As might be imagined, Asian character sets like Chinese account for a big part of Unicode, but by no means all. The Unicode web site, `unicode.org`, has charts of all the characters; it's fascinating and well worth a detour.

32	space	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

Figure 2.6: ASCII characters and their numeric values.

The bottom line: a digital representation can represent all of these kinds of information and indeed anything that can be converted into numeric values. Since it is just numbers, it can be processed by digital computers; as we will see in Chapter 9 it can be copied to any other computer by the universal digital network, the Internet.

2.3 Bits, Bytes, and Binary

"There are only 10 kinds of people in the world—those who understand binary numbers and those who don't."

Digital systems represent information of all types as numeric values, but perhaps surprisingly, they do not use the familiar base ten (decimal) number system internally. Instead they use binary numbers, that is, numbers in base two.

Although everyone is more or less comfortable with arithmetic, in my experience their understanding of what a number means is sometimes shaky, at least when it comes to drawing the analogy between base ten (totally familiar) and base two (not familiar to most). I'll try to remedy this problem in this section, but if things seem confused or confusing, keep repeating to yourself, "It's just like ordinary numbers, but with two instead of ten."

2.3.1 Bits

The most elemental way to represent digital information is with bits. As noted in the quotation at the beginning of this chapter, the word *bit* is a contraction of *binary digit* that was coined by the statistician John Tukey in the mid-1940s. (Tukey also coined the word *software* in 1958.) It is said that Edward Teller, best known as the father of the hydrogen bomb, preferred "bigit," a term that mercifully didn't catch on. The word "binary" suggests something with two values (the prefix "bi" means two, of course), and that is indeed the case: a bit is a digit that takes on either the value 0 or the value 1, with no other possibilities. This can be contrasted with the 10 possible values of the decimal digits 0 through 9.

With a single bit, we can encode or represent any choice that involves selecting one of two values. Such binary choices abound: on/off, true/false, yes/no, high/low, in/out, up/down, left/right, north/south, east/west, and so on. A single bit is sufficient to identify which one of the pair was selected. For example, we could assign 0 to off