Over 1,000 personal computer and word processing terms

COMPUTER DICTIONARY FOR BEGINNERS

Written for regular people—
not computer experts—
in language anyone can understand

ARTHUR NAIMAN

Ballantine/31223/$6.95 in USA • $8.95 in Canada

COMPUTER DICTIONARY FOR BEGINNERS

Arthur Naiman

Ballantine Books • New York

This book was written on a microcomputer-based word processing system put together by Tony Pietsch. Built on an S-100 bus, it has a Z80 CPU running at 4MHz, 64K of RAM, two single-density 8" drives, and a NEC Spinwriter printer. The operating system is CP/M and the word processing program is WRITE; I also use The Word+ spelling checker.

Library of Congress Catalog Card Number: 83-90068
ISBN: 0-345-31223-6

Manufactured in the United States of America

Designed by Gene Siegel

First Edition: October 1983

10 9 8 7 6 5 4 3 2 1

COMPUTER DICTIONARY FOR BEGINNERS

For R.

semper constans

Acknowledgments

It's every obsessive's dream to find someone who's as obsessive as s/he is, and about the same things. In my case, Ron Lichty answers that dream—at least as far as writing about computers is concerned. He was involved in every stage of this project and provided much crucial advice.

Like Ron, Tom Crosley read the manuscript at two different points and gave me the benefit of his expertise. Kevin Layer read a draft and critiqued it extensively; he caught several errors and provided a lot of important information. Stan Brenner also gave the manuscript a careful reading, and noticed some things everybody else missed.

I usually (but not always) followed the suggestions of these four computer experts. For that reason, and because there's so much I don't know about computers, I must place the responsibility for any errors which remain squarely on their shoulders.

Computer novices (or semi-novices) also read the manuscript and told me where it was unclear (or unfunny). Esther Wanning did that in addition to admirably performing her duties as my agent, which included endless hours of listening to my kvetching; Gloria Polanski did that in addition to helping me with the endless hassles of buying a house. Paul Glassner gave the manuscript a paricularly thorough and thoughtful going over. Meg Holmberg went through it twice and made many valuable recommendations. The diagram at **programming language,** the comparative illustrations at **motherboard** and **chip,** the special section on how computers work, and several other features of the book originated as her ideas.

Gar Smith produced more good cartoons than I could have hoped for, and put up with all my nit-picking to boot. Joelle Delbourgo showed the same exquisite taste in editing this book as she did in selecting it; I'm particularly grateful for her

open-mindedness and reasonableness in discussions on various points. Ted Johnson did a superb job of copy editing.

I also want to thank the following people for support of many different kinds: Matthew Lasar, Tony Pietsch, Clem Cole, Sandy Van Broek, Guy Steele, Rita Gibian, Albert and Nettie Naiman, Stephen McNabb, Janet Pfunder, Jonathan Ayres, Cathy Roberts, Nancy Shine, Pat Diehl, Amy Bomse, Mark McDonald, Eric Jungerman, Burt Sloane, Richard Fateman, Margaret Butler, Ed Berman, Bill Hass, Debra Dadd, Phyllis Saifer, Gail Nielsen, Alan Levin, Miki Shima, Jane Margold, Ed Kelly, Brad Bunnin, Rose Slais, Ruth Anne Martin and Tom Brockland.

COMPUTER DICTIONARY FOR BEGINNERS

Introduction

At last—a dictionary of computer terms that isn't written for engineers. If you've been frustrated by computer jargon and haven't had any easy way to find out what it means, this book is for you.

Computer Dictionary for Beginners defines the 1100 or so terms you're most likely to come across when reading about computers, shopping for them, or trying to understand the manuals that come with them, and it explains those terms in clear, everyday, conversational English. There's even some humor (but never the kind that confuses you about the real meaning of a word).

Unlike other computer dictionaries, this book is designed for beginners. So I've kept technical terms out of the definitions as much as possible. The ones I do use are all defined elsewhere in the book; if you run across one that isn't familiar to you, it's easy to look it up.

Mostly I've chosen terms that refer to personal computers—the kind individuals can afford to buy and use themselves. I assume you're interested in *using* a computer, not in designing or building one, so engineering jargon like "virtual address" and "bit slice" is left out. But I have included some "hacker jargon"—the language computer experts use when talking to

each other—not because you'll ever need to know it, but because it's interesting and fun.

Pronunciations are provided for all the terms that need them (including some you could find in a regular dictionary, if you went to the trouble). For many entries, the origins of the term are also given, as are typical sentences (in italics) to show how the term is used.

I've included several charts and diagrams, because certain information makes the most sense presented that way. Cartoons liven things up and help illustrate difficult concepts. A special section—*How Computers Work—in 500 Words or Less*—gives you a solid basic understanding that would be hard to get by looking up individual words.

If you come across a term you feel should be in this book but isn't, drop me a line at Ballantine Books, 201 East 50th Street, New York, New York 10022 and I'll try to include it in subsequent editions. But I think you'll find that just about any term you're likely to run into is already here.

How This Dictionary Works

Since I'm writing for nontechnical readers like yourself, I haven't tried to make my definitions more precise than you need them to be. Instead I've concentrated on making them clear and useful.

Some terms have applications wider than just computers, but I've only defined what they mean in the computer field. For example, the word "up" has dozens of meanings other than its computer meaning of "working."

Sometimes I've had to use technical terms in the definitions. If I hadn't, each definition would be ten pages long. But when I use a term that I think will be unfamiliar to most readers, I boldface it—so you know you can look it up (although, in fact, *all* the technical terms I use are defined elsewhere in the dictionary, whether they're boldfaced or not).

I also boldface synonyms for the word being defined, as well as additional words or phrases I'm defining at the same time (like ***the* field,** which means something different from the word **field** by itself).

Unless otherwise stated, abbreviations are pronounced letter by letter (for example, "EMI" is pronounced "ee-em-eye"). In the pronunciations, "g" is always hard (as in "go"); if I want to

indicate soft "g," I use "j." By the same token, "s" always equals "ss," never "zz."

I haven't included brand names of popular programs or pieces of hardware, for two reasons:

1) what gets included and what gets left out of such a list is always subjective, and unfair to many excellent products an author simply doesn't know about; and

2) mentioning a brand name implies a certain sort of approval, and there are several popular products I wouldn't want anyone to think I endorse.

The exceptions to this rule are brand names that you're likely to come across in an ad for some other product, without any explanation as to what they are (like CP/M, SoftCard, and the names of various programming languages and CPU chips).

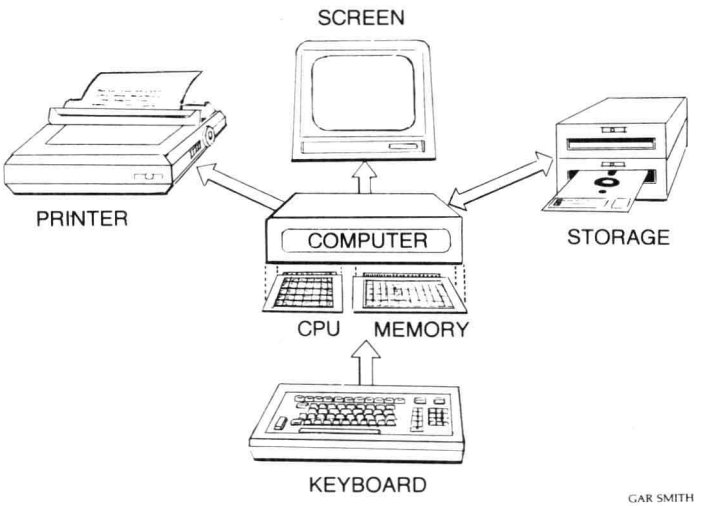
Some computer terms are numbers; for example, "8086" is the name of a type of chip. I list numbers in alphabetical order, mixed in with the other terms. Thus "8086" is under "e," because "eighty" begins with "e." But if there are several number terms right in a row, I put them in ascending numerical order, rather than in strict alphabetical order.

I alphabetize as if each entry were one long word; spaces and hyphens don't count. Symbols are listed at the end of the book. Terms that begin with a symbol but also include letters (like **.TXT** and **/bin**) are listed by their first letter.

How Computers Work—in 500 Words or Less

I won't waste any words telling you that 500 words isn't going to allow for a very detailed picture of how . . . oh my God! I've just wasted 28 . . . no, 30 . . . uh . . .

Look at the drawing below. All computers (and word processors) are made up of these basic pieces of **hardware** (some



GAR SMITH

include other components too, of course). When you type a character (any letter, number or symbol) on the keyboard, a coded electronic impulse travels over a cable to the computer, where it's stored in **memory** (this kind of memory is called **RAM**).

A computer's memory is basically a whole slew of little switches. Since a switch can only be on or off, computers reduce all information to a series of ons and offs, ones and zeros (in much the same way that Morse Code reduces all communications to a series of dots and dashes).

Today's computer technology allows thousands of electronic switches to be squeezed onto a piece of silicon the size of a baby's fingernail (called a **chip**). These chips are mounted and wired together on thin slabs of fiberglass called **boards.**

Not only are the switches used in computers very small, they're also very fast; some computers can throw millions of switches a second. (Primitive computers used mechanical switches, which were relatively big, clumsy and slow.)

Once a character is lodged in RAM, it's transmitted to a screen much like the one on your TV, which is also called a **monitor** or **display.** (Did you notice how I saved a word back there by using "it's" instead of "it is"?)

To actually *do* something to the characters you've placed in RAM—that is, to process your data—you need the **CPU** (the "central processing unit"). The CPU—made up of one or more chips—is capable of doing thousands of calculations a second; whatever a computer does, from word processing to Donkey Kong, it reduces to numerical calculations (to switch-throwing, really).

When you turn your computer off, RAM goes blank, so you need some way to permanently store your processed data (whether it's a letter, a game you've written, or a list of your accounts receivable). This is done on a **storage device** which puts your information on **media** like **floppy disks, hard disks** or cassette tapes; these store data in the form of magnetic impulses—basically the same way recording tapes store sound.

Another use for storage devices is to "load" **software** (instructions to the computer—also called **programs**) into the computer. These instructions can also be given to a computer by a special kind of hard-to-change memory called **ROM,** but programs on disks are usually more convenient (because you can change them much more quickly).

Since shipping your whole computer system to someone is a little inconvenient, **printers** were invented to produce **hard**

copies (on paper) of letters or other documents. It's also convenient to have hard copies for your own use, because it gets to be annoying to always have to look at things on the screen.

498 . . . 499 . . . I did it!!! . . . whoops.