A BYTE BOOK

>$16.95

MARK DAHMKE

The BYTE Guide to CP/M -86

The BYTE Guide to CP/M-86

by
Mark Dahmke

McGraw-Hill Book Company
New York St. Louis San Francisco Auckland Bogotá Hamburg
Johannesburg London Madrid Mexico Montreal New Delhi Panama
Paris São Paulo Singapore Sydney Tokyo Toronto

Library of Congress Cataloging in Publication Data

Dahmke, Mark.
The byte guide to CP/M-86.

Includes index.
1. CP/M-86 (Computer operating system) I. Title.
QA76.6.D3325 1984 001.64'25 83-19613
ISBN 0-07-015072-9

1234567890 DOCDOC 89876543

ISBN 0-07-015072-9

Printed and bound by R.R. Donnelley and Sons, Inc.

Preface

CP/M has become the de facto standard operating system for the 8080, 8085, Z80, and 8086/8088 microprocessors. "De facto" does not necessarily imply best. CP/M on the 8080 or Z80 has its particular problems. CP/M-86 (the subject of this book) carries forward some of those problems. For the most part, CP/M has been a positive influence in the world of microcomputers because it has helped to standardize software. It is now possible to buy any of hundreds of "CP/M compatible programs," load them, and realistically expect them to work.

One of the major drawbacks to CP/M-80 (the 8080, Z80 version) was the lack of good documentation. The manuals supplied by Digital Research were hopelessly inadequate, so many user's guides like this one were published to fill the information gap. CP/M-86 comes with manuals that are harder to compete with. They are well organized and cover all operational aspects of the operating system, with one exception—they neglect to explain *why* you need an operating system. Their manuals assume that you understand this. The *BYTE Guide to CP/M-86* explains each part of the operating system and and each program or utility that comes with it in terms of its use and why it needs to be discussed at all.

Such an approach makes this book more than just a reference manual. It is compatible with my previous book *Microcomputer Operating Systems*, although this book stands on its own.

An appendix is included that lists references and sources for further reading. I have also listed books that may not relate directly to CP/M-86, but that might be interesting to the reader who wants to learn more about operating systems.

Appendices III and IV contain complete listings and descriptions for several public domain programs written in the 8086 instruction set for use on any CP/M-86 system. The first is a sample program that performs some of the functions of the DIR built-in command in CP/M. The purpose of the example is to demonstrate the use of many of the BDOS function calls.

Appendix IV includes a full description, flow charts, and listings for Char-io, a set of subroutines that simulate disk-oriented character input and output. Since it is often necessary to filter a text file for special characters, it is desirable to read the file a character at a time, rather than a record at a time. The rd-char and wr-char routines allow a file to be transferred as a stream of characters, with all disk access being handled transparently. The listing for a library file including these routines is given, with two examples for its use. With the examples, readers should be able to use char-io for any purpose, without much trouble.

Acknowledgments

I would like to thank Digital Research in general, and Susan Raab in particular, for their support. Bruce Roberts, Ed Kelly, Chris Morgan, and Steve Ciarcia were the instigators behind the project, and I would like to express my appreciation for their efforts.

Table of Contents

1

Introduction: What Is CP/M-86?

1.1 INTRODUCTION

CP/M stands for "Control Program for Microcomputers." It was first developed for an Intel 8080 microcomputer system by Gary Kildall, founder of Digital Research. The acronym CP/M means little to the uninitiated user. All digital computers require programs or sequences of instructions to operate on. These programs or instructions are referred to as software. The actual computer is referred to as the hardware. The hardware may consist of the central processor, memory, disk drives, and so on. The central processor is responsible for executing the instructions that you give it. The problem is that the instructions it wants to read are not very human readable. That is where computer languages such as BASIC or Pascal come in.

A language such as BASIC reads English-like commands and converts them into commands that the computer can understand. This is referred to as compiling or interpreting a program. Even a language like BASIC needs help when it wants to read data from a disk or write characters to a video display terminal. CP/M, the operating system, is a control program that can be called upon to carry out menial chores in a generalized fashion. For example, suppose you have a BASIC interpreter and a word processor program written in some other language. If every program or every language went its own way, reading files from disk and communicating with printers or terminals as it pleased, there would be no standardization in the computer.

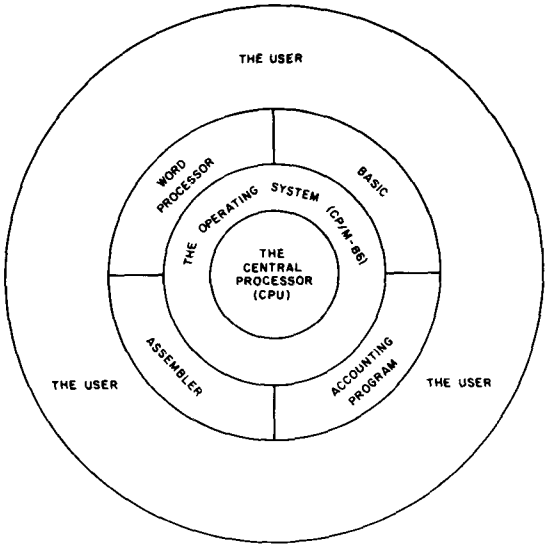

Figure 1.1: The relationship between the microprocessor, operating system, applications, and users.

The control program or operating system acts like a traffic cop who makes sure everyone obeys the conventions of the system, or, if you like, the "rules of the road."

Note: CP/M-86 is one of the control programs developed for the 8086 and 8088 microprocessors. It is derived from the 8080 version of CP/M (now referred to as CP/M-80, or CP/M 2.0). CP/M for the 8086 or 8088 is always referred to as CP/M-86. In this book, we will assume that all references to "CP/M" refer to CP/M-86, CP/M for the 8080, 8085 and Z80 microprocessors will be referred to as CP/M-80.

1.2 FEATURES AND FACILITIES

The purpose of an operating system is to provide a common environment for programs to run in. This can mean many things. All microcomputer systems have the following characteristics: input and output devices, main memory, a central processor, and mass storage. (See Figure 1.2.) The problem is that many manufacturers have their own ideas about how these building blocks should go together and have built products that can run CP/M, but are all different in some respects. For example, the disk interface may use different circuitry or may store information on the disk in a different fashion. Alternately, the video display terminal might be connected to the central processor differently.

To take all these minor variations into account, CP/M was de-

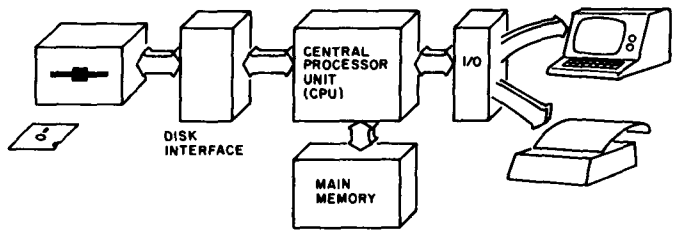

Figure 1.2: Microcomputer system organization.

signed with a customized section that must be set up by the manufacturer of the computer. This section is called the basic input/output system (BIOS) and is the lowest level of the CP/M operating system. CP/M calls upon this section to accomplish its needs.

Because the BIOS is the only portion of CP/M that changes from one manufacturer's computer to the next, the user need not be aware of any differences in the actual operation of CP/M. To get around the problem of incompatible software, CP/M supports logical devices that are indifferent to the actual hardware. Such devices include a console or video display terminal, a printer, disk drives, and perhaps a modem or telephone line interface. All software, whether it be BASIC, a word processor, or a program written by the user, must use the logical devices provided by CP/M. If this is done, any program should be able to run on any CP/M system, regardless of who wrote it or what system it was developed on. This portability of software implies that anyone running CP/M can trade software with anyone else running CP/M.

Note that CP/M-80 programs should run on any CP/M-80 microcomputer, and CP/M-86 programs should run on any CP/M-86 microcomputer. However, as stated earlier, due to differences in instruction sets, a CP/M-80 program will not run directly on a CP/M-86 system, and vice versa.

Another feature of CP/M is that it provides built-in functions that would be tedious to recreate if the programmer were forced to supply them. Functions such as open disk file, search disk directory, and compute file size are easy for CP/M to do, but would take a programmer a long time to figure out.

1.3 THE ENVIRONMENT

CP/M-86 requires a certain environment to operate in. The most important element of the environment is the microprocessor. As the name implies, CP/M-86 was designed to operate on an Intel 8086 16-bit microprocessor. The 8088 will also work since it uses the same instruction set as the 8086 and operates identically.

The type of microprocessor is important because many different microprocessors have different instruction sets. This means that the instruction code for "call subroutine" on one microprocessor may mean "store data into memory" on another. Different microprocessors are developed by different manufacturers for different purposes, and the instruction sets are often tailored to meet those needs. The 8086 and 8088 are newer, more powerful components, and their instruction sets have been enhanced from previous generation microprocessors. Specifically, the 8086 is an outgrowth of the incredibly successful Intel 8080. The 8080 was introduced in the early 1970's and has been the most popular of the microprocessors. It addresses memory in 8-bit wide chunks called *bytes*.

A *bit* is a *b*inary dig*it* and can be either a one or a zero. A byte is a group of eight bits. Since each bit can be either a one or a zero, the range of numbers possible with an 8-bit microprocessor is: 00000000 to 11111111 or, in decimal, 0 to 255. For many applications, an 8-bit computer is more than adequate. However, the 8080 (and most 8-bit machines for that matter) can only address memory with a 16-bit address, allowing for 64K bytes (65535 in decimal). The *K* after the 64 is short for *kilo*. One kilobyte is not 1,000 bytes but 1,024 bytes. Since computers use a base 2 or binary number system, 2^{10} or 1024 is more convenient than 1,000, so 64K is $64 \times 1{,}024$ or 65,535 instead of 64,000.

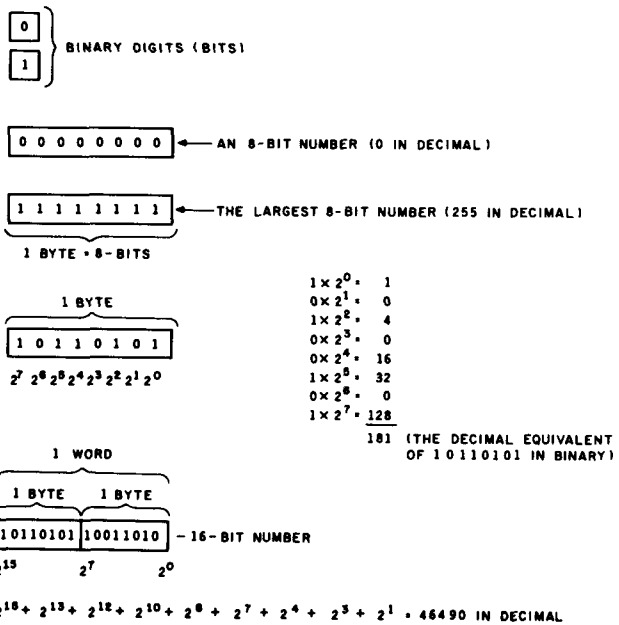

Figure 1.3: 8-bit and 16-bit binary numbers.

The 8086 is called a 16-bit microprocessor because it addresses memory and operates on data in 16-bit chunks called *words*. A word is two bytes long. When the 8086 reads data from a memory location, it gets two bytes at a time instead of one. This means that the 8086 runs faster and is more efficient in handling data.

In order to run CP/M-86, the microcomputer system must have at least 56K bytes of main memory. The memory, or the address space, of the 8086 is a continuous sequence of words, in which each word is 16 bits long. Although the 8086 can operate on 16-bit words, it is still convenient to speak of the amount of memory in bytes. If a particular system has 64K words of memory, it would be correct to say that it has 128K bytes of memory. The 8088 microprocessor is functionally identical to the 8086, but addresses memory in bytes instead of words. Thus, for those who wish to upgrade their 8080 or other 8-bit computer to an 8086, it is sometimes possible to replace the old circuit card with one that has an 8088 processor. Note that the 8088 addresses memory in bytes, but it operates on 16-bit words just like the 8086. Instead of loading a word of memory in one operation, the 8088 requires two loads of one byte each to accomplish the same thing.

The 8086 microprocessor allows the user to address up to one megabyte of memory because it generates a 20-bit memory address. Note that 2^{20} is 1,048,576. If this number is divided by 1024 to convert to kilobytes, we find that a megabyte is equal to 1024 Kilobytes. Since all the 8086 internal registers are 16 bits long, the 20-bit memory address must be generated indirectly. The 8086 uses a memory addressing technique called segment addressing, as shown in Figure 1.4. When a program is loaded, it is placed into memory at an address designated by CP/M-86 at load time. The program is treated as if it were located at address zero in memory, but is actually at a higher address. In the example in Figure 1.4, the program is loaded at 34FD0H (the H stands for hexadecimal notation or base 16) in absolute memory. The code segment register is loaded with the top 16 bits of the start address. The program counter currently holds 0045H, the address of the next instruction to be executed. The 8086 shifts the code segment address to the left 4 bits, making it into a 20-bit number. Notice that only the top 16 bits were stored in the first place. This means that programs can only start on 16-byte boundaries. The shifted code segment is added to the program counter, and the result is used to access the instruction in the program.

With the segment scheme, it would appear that programs cannot be larger than 64K bytes (!) and that no more than 64K bytes of data may be addressed. Actually, the programmer has complete control of the segment registers and can set up as many data or code areas as desired. Several machine instructions are provided to make this easy.

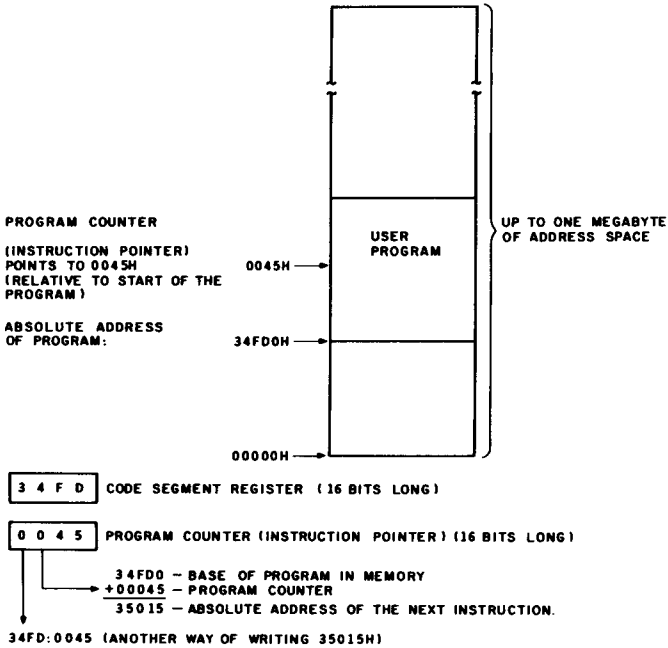

Figure 1.4: Addressing memory with the 8086.

Since the program and data areas can be separately addressed in the 8086, the user has several options for organizing a program in memory. Three memory models are provided in CP/M-86: the Small model, the Compact model, and the 8080 model. Normally, the user does not need to be aware of these memory organizations, but they are mentioned here to show that memory must be managed and that CP/M-86 performs that function. Chapter 5 discusses memory management in detail.

1.4 DISKS AND DISK FILES

Almost all modern microcomputer systems have floppy disk drives or a hard disk drive. Floppy disks have become the industry standard for software storage and distribution.

Floppy disks are circular, flexible sheets of mylar with a magnetic coating. They come in two sizes: 8 inches and 5.25 inches in diameter. A typical 8-inch disk can hold 243,000 bytes of information in single-density mode, 486,000 bytes in double-density mode, and 1,200,000 bytes in double-density, double-sided operation.

Information is recorded on a floppy disk in tracks and sectors (see

Figure 1.5). A track, or concentric circle, is divided into sectors. Some disks are hard sectored (not to be confused with hard disk). Hard-sectored disks have ten or more holes cut around the inner part, each hole indicating the start of a sector. Most disks in use now are soft sectored—meaning that the sector layout is determined by software, not hardware (holes in the disk).

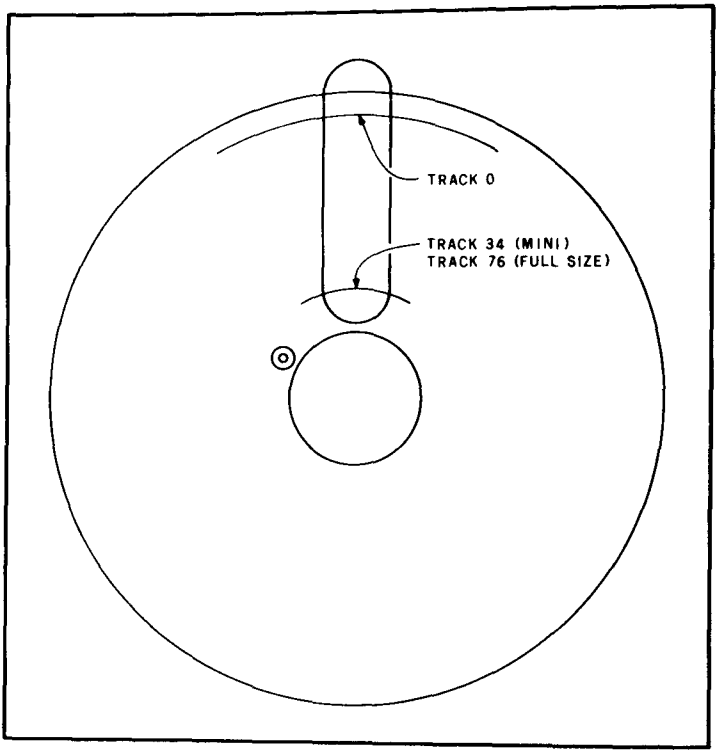

Figure 1.5: Floppy disk geography.

Standard 8-inch single density disks have 77 tracks, each divided into 26 sectors. The CP/M standard (following the IBM 3740 disk format standard) stores 128 bytes per sector. Therefore, each track can hold 26 × 128 bytes or 3328 bytes. Since there are 77 tracks, a total of 256,256 bytes or characters can be stored on a disk. CP/M normally reserves two tracks for the operating system, or loader program, and the directory. A normal single-density disk directory has room for 64 entries. Each directory entry contains the file name, file type, and location information of each file on a disk. The directory for a given disk is unique and restricted to that disk. Only files on a particular disk may have directory entries on that disk.

1.5 COMPATIBILITY WITH CP/M-80

CP/M-80, developed for the 8080 microprocessor will also run on the 8085 and the Z80 microprocessor. As mentioned earlier, the instruction set of the 8080 is not compatible with that of the 8086 or 8088. However, the 8086 was designed to be upward compatible with the 8080. This means that it is possible to translate an 8080 program (at the instruction mnemonic level) into the 8086 instruction set. A program exists that can do this for standard Intel mnemonic assembler programs.

For programs that have been translated from CP/M-80, CP/M-86 offers an 8080 memory model in which all segment registers are set to the same starting address, giving the effect of all program and data areas overlapping. This environment most accurately duplicates the architecture of the 8080. For more information on the 8080 model, consult Chapter 5, Section 5.11.

If you are transferring a BASIC or Pascal (or other language) program, you would normally recompile it on the CP/M-86 system. This is not always the case, but under most circumstances, it is possible.

All disk files and disk formats from CP/M-80 are compatible with CP/M-86, although program files have a file type of .COM in CP/M-80 and .CMD in CP/M-86. Also, command (.CMD) files have a header record in CP/M-86 that contains allocation information for the four segments (among other things). Since command files from CP/M-80 are not executable on CP/M-86, this change is not a problem. Source text files and data files are all directly transferrable to CP/M-86.

The CP/M-86 operating environment was designed to be as compatible with CP/M-80 as possible. If you are an experienced CP/M-80 programmer, most functions of the operating system will already be familiar to you.

1.6 OPERATIONS

When you start up your microcomputer system for the first time, it is necessary to take some safety precautions. The most important one is to make a back-up copy of all the diskettes supplied with the system. Disks don't last forever, and you may need to go back to the original disks if one of your daily working copies is damaged or loses data.

The COPYDISK program supplied on the master disk will make the back-up copy for you, but first, you will probably need to format the blank disk you intend to use. Most manufacturers supply a program called FORMAT or FMT or INIT with their computer. It will clear off a