GUIDE TO THE IBM PERSONAL COMPUTER

WALTER SIKONOWIZ

GUIDE TO THE IBM PERSONAL COMPUTER

WALTER SIKONOWIZ

A Micro Text / McGraw-Hill Copublication
New York, N.Y.

Library of Congress Catalog Card Number: 82-062819

ISBN: 0-07-057484-7

Chapters Six and Sixteen were contributed by Daniel Lewart

Micro Text Publications, Inc.
McGraw-Hill Book Company
1221 Avenue of the Americas
New York, N.Y. 10020

Pam Edstrom of Microsoft Corporation and Janice McGowan of MicroPro International were helpful in preparing this book, and Michael Bahler assisted in cover photography.

Preface

The introduction of the IBM Personal Computer in the waning days of 1981 was a newsworthy event on two counts. Not only did it usher in a new era of low-cost 16-bit computing; it also marked the entry of America's largest and best-known mainframe computer manufacturer into the personal-computing market.

Public response to the new PC was enthusiastic, and with good reason. Here was a machine that when fully configured with two disk drives and a quarter megabyte of main memory could be had for the relatively modest sum of about $4000. Moreover, the PC was backed by a large and well respected service organization — something altogether too rare in the world of microcomputers. Is it any wonder, then, that the IBM Personal Computer, doubly blessed with an enviable pedigree and solid after-sale support, should have become an immediate success?

To even the most casual observer of the microcomputer scene, it is quite apparent that the people buying PCs today are, on the average, a different breed from those who bought Apples and Ataris in years past. A growing percentage of the people now buying computers are doing so for business or professional reasons rather than for recreation (and this applies not just to PC buyers, but to others as well). Granted, even the "serious" user may get the urge to blast a Klingon now and then, but in the main, interest has shifted from fun and games to professional advancement.

With that shift in interest has come a change in temperament on the part of the typical microcomputer purchaser. The new breed of buyer is not a hobbyist with infinite patience, or an electronics engineer with experience in digital design. He simply wants to get his computer running and keep it that way with as few problems as possible. It is to that person that this PC user's guide is dedicated.

This book should appeal to two classes of reader: the person just contemplating the purchase of a Personal Computer, and the person who already owns one. If you like the PC but wonder whether its purchase would be justified, you can find the answers to most of your questions right here. And if you already own a PC but find IBM's documentation a little above your head, the book in hand may be just what you need.

Chapter One opens with an overview of Personal Computer hardware (i.e., mechanical and electronic equipment). Then, Chapter Two introduces some of the rudiments of programming in the BASIC language. Chapter Three carries things one step farther by showing how control passes from statement to statement in the course of a program's execution. It also offers suggestions on how to write prog-

ram code that is straightforward and readable — not a twisted mass of spaghetti logic.

The Personal Computer's sophisticated yet easy-to-use operating system, which controls data transfer, storage, and retrieval, is the topic of Chapter Four. Chapter Five carries on in the same spirit by showing how the functions and capabilities of the DOS (the disk operating system) can be called up from BASIC to allow the storage of programs and data on magnetic disks. Chapter Six discusses IBM's recently released DOS 2.0, and the enhancements it adds to an already powerful operating system.

Chapter Seven provides a thorough grounding in character strings and the various ways in which they can be manipulated. Number crunching, the programmer's amusing way of referring to numerical computation, is the topic of Chapter Eight. Next, Chapter Nine provides an introduction to the writing of interactive programs that accept keyboard input from an operator during the course of their execution. Chapter Ten complements its predecessor by showing how text can be placed at will on the video screen, thus providing the operator feedback so necessary in an interactive program.

Chapter Eleven should be fun for all; it deals with the techniques of color graphics and simple animation. We get down to serious business, that of debugging and testing a program, in Chapter Twelve. It is easy to give debugging and testing short shrift because they really are tedious, and often exasperating. Nevertheless, the conscientious extermination of bugs is necessary if reliable programs are to be written.

Chapter Thirteen covers the IBM Parallel Printer Adapter and how it may be interfaced with a suitable printer to produce hard copy (i.e., printed records). In Chapter Fourteen we examine the IBM Asynchronous Communication Adapter and show how it can be used to link the PC with modems, other computers, and serially interfaced printers. Chapter Fifteen provides a lighthearted look at the production of sound and music (written, incidentally, by someone with no musical talent at all; *caveat lector*).

Chapter Sixteen adds more to our knowledge of BASIC by discussing the special features of BASIC 2.0.

Finally, writing programs that make good use of a joystick or light pen is the topic of Chapter Seventeen. Graphics applications probably come most readily to mind, but there are other interesting things to be done as well.

So much for the preliminaries. It's now time to fire up your PC, or your imagination if no machine is at hand, and get on with the rest of this book!

CONTENTS

1. Getting Acquainted with the System

Any complex system — whether it be electronic, biological, or political — must be viewed on multiple levels in order to be understood. So it is with the IBM Personal Computer. We have to consider hardware (electronic and mechanical components) and software (programs, or sequences of instructions) to get a balanced perspective on the PC. Moreover, hardware and software are themselves composed of interesting sublevels that must also be taken into account. Fortunately, the hardware/software hierarchy of the PC is not at all difficult to understand given the proper background and incentive. That is what this introductory chapter is all about.

A Hardware Overview

Illustrated in Fig. 1-1 are the major hardware components of the IBM Personal Computer. The large oblong box with the rectangular recess in its front face is known as the System Unit. It houses most of the electronic circuits of the PC, including the central processor, main memory, and a pair of 5.25-inch disk drives, which provide mass storage of programs and data.

Fig. 1-1. Components of the IBM Personal Computer system include (from left to right) an 80 cps dot-matrix printer, a high-resolution video monitor, the System Unit containing two disk drives in addition to memory and processor, and a detached, tiltable keyboard with 83 keys. (Courtesy of IBM Corp.)

Perched atop the System Unit is the IBM high-resolution video monitor. It displays 25 lines of text with a maximum line length of 80 characters. The IBM monitor is limited to monochrome reproduction of text and simple line graphics, but by adding a suitable adapter card and a color video monitor, it is possible to display high-resolution color graphics in addition to text.

Also visible in Fig. 1-1 is the 83-key detached keyboard that constitutes a user's primary interface with the PC. A six-foot coiled cord links the keyboard with the System Unit and allows considerable flexibility in the keyboard's placement — on a desk, in a lap, or wherever else might be convenient. As with all IBM keyboards, this one has the tactile and audible feedback and careful design that make typing a pleasure.

To the left of the keyboard in Fig. 1-1 is the optional IBM 80 cps (characters-per-second) dot-matrix printer. This is a thinly disguised version of the familiar Epson MX-80, the most popular printer in personal computing. Most applications will be well served by this low-cost machine, but for those applications such as word processing that demand printed copy of a higher caliber, there are a variety of daisy-wheel printers on the market that will interface readily with the PC.

Figure 1-2 provides a close-up view of the System Unit with its top cover removed. In the upper-right-hand corner of the photograph we find a 63.5-watt switch-mode power supply that furnishes all the voltages required by the system. Below the power supply is a pair of 5.25-inch disk drives. At his option, a user can have one, two, three, or four disk drives, although no more than two standard-height drives will fit inside the System Unit. The remainder must be mounted externally in a suitable chassis.

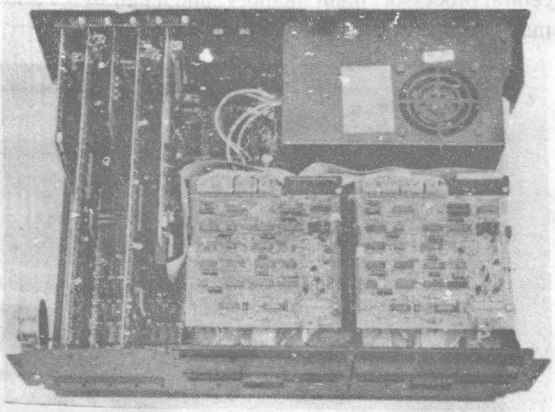

Fig. 1-2. Interior view of the System Unit with its top removed. Two disk drives are visible in the lower right-hand corner of the photo, while the upper right-hand corner is occupied by a fan-cooled power supply. The PC's System Board mounts horizontally in the left half of the chassis, and into this circuit board five auxiliary adapters have been plugged. (Courtesy of IBM Corp.)

While it may be difficult to see in Fig. 1-2, the floor of the left half of the System Unit is covered by an 8 × 10 inch printed-circuit board, the so-called System Board, on which is mounted a considerable portion of the PC's electronics. This includes the CPU (central processing unit), the random-access memory, and the read-only memory containing the BASIC interpreter. (RAM and ROM are forms of semiconductor

memory. Do not despair; these and other strange words that keep creeping into the text will be discussed at length in the next couple of sections.)

The System Board also bears five sockets, into which various auxiliary adapter boards can be plugged to extend the capabilities of the PC. If Fig. 1-2 is examined carefully, five adapter boards mounted perpendicular to the System Board can be seen at the extreme left of the photo. Also visible in the lower left-hand corner is a small speaker, through which the PC beeps and generates musical sounds.

Figure 1-3 shows the back side of the System Unit. Notice the five slots on the right side of the panel. Connectors mounted on the various plug-in adapters can protrude through these slots, thus allowing electrical connections to be made between the boards and external equipment like a video monitor or printer. If a slot is unoccupied, it can be covered by a metal plate to prevent the escape of RF radiation (which can interfere with radio and television reception) and keep dust out of the cabinet. The grille at the center of the System Unit's rear panel hides a cooling fan.

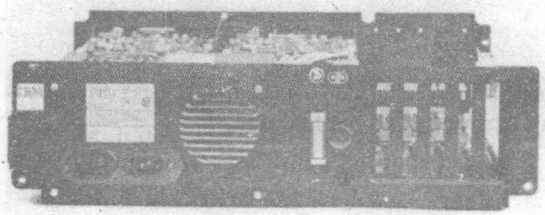

Fig. 1-3. Rear view of the System Unit. Slots on the right allow sockets mounted on auxiliary adapter boards to protrude through the steel chassis. If not in use, an access slot may be covered to prevent the escape of RF radiation. (Courtesy of IBM Corp.)

Bits, Bytes and Words

Let us now examine the PC, and computers in general, on a more abstract level — that of data representation. The smallest unit of data in any digital computer is the bit, or binary digit. A bit can assume one of two values, 0 or 1, and no other. If we want to represent larger numbers, we must combine bits into words. For example, a 2-bit data word can have four possible values — 00, 01, 10, or 11 — while a 3-bit word can have eight: 000, 001, 010, 011, 100, 101, 110, or 111. In general, the number of unique values that can be assumed by a word of *n* bits is given by 2 to the *n*th power, or 2 multiplied by itself *n* times.

A word of eight bits is a special entity in computing, and in recognition of its importance it bears a unique name: the byte. Even though the IBM PC is a 16-bit computer, it uses byte-size data, which makes it an interesting hybrid. In an upcoming discussion of the 8088 microprocessor, around which the PC was designed, we will have occasion to examine some of the advantages and disadvantages of using byte-size data in a supposedly 16-bit machine. For the time being, however, let us just note that the byte is a very important entity, one that will crop up again and again in the pages ahead, and that it would be wise to commit to memory here and now the byte's definition as a word consisting of eight bits.

NUMBERS

Binary words, which are in truth the currency of information exchange within a computer, may represent any conceivable form of data. For example, numbers are obviously essential to a computer, and they can be encoded in a variety of ways, as we shall see in Chapter Eight.

For the moment, let us look at just one method of representing numbers, the pure binary code. A number expressed in binary form can easily be made comprehensible to a decimal-oriented mind. One multiplies the status of each bit (1 or 0) by the significance of the column it occupies and sums the values obtained to arrive at a decimal (base 10) result. For example, consider the conversion of 10011 to decimal form:

Decimal Significance:	16	8	4	2	1
Binary Bit Value:	1	0	0	1	1

Since multiplying by 0 yields 0, we need only sum the decimal values corresponding to the columns in which a 1 appears. Thus we find that:

16+2+1 = 19

Once they become familiar, small binary numbers are almost as easy to read as decimal. The large ones, of course, are so long that they cannot be deciphered at a glance, and for that reason it is common for programmers to use octal (base 8) and hexadecimal (base 16) numbers in preference to binary. We will have occasion to talk of the octal and hexadecimal number systems later on. For the present, let us emphasize that inside the computer all numbers are represented in binary form.

ALPHANUMERIC CHARACTERS

Alphanumeric characters (letters and numbers) of the sort that make up everyday English text are another important class of information that can be encoded as digital words. The code used for this purpose is a universally accepted one known as ASCII (American Standard Code for Information Interchange). Seven bits are used to encode 96 printable characters and 32 non-printing control characters that function as commands within a computer. The control codes have values from 0 to 31, while the printable characters (a, b, c...1, 2, 3...etc.) have ASCII codes from 32 to 127. (It is convenient to speak of ASCII codes in decimal terms, but remember that the computer must deal with them in binary form.)

IBM has long used its own proprietary code (known as EBCDIC) for representing

alphanumeric characters, but in building the PC they broke with tradition and adopted ASCII. However, they extended it to 8 bits (a whole byte) to allow an extra 128 symbols to be encoded. These include Greek letters, mathematical symbols, and assorted pictograms (hearts, diamonds, little smiling faces, etc.). This gives the PC user a 256-character symbol set from which to construct imaginative displays of all sorts.

TRUE AND FALSE

In addition to alphanumeric characters, digital words may also be used to encode truth and falsity. As the reader may know, it is common in mathematical logic for a 1 to represent a TRUE condition, and for a 0 to represent a FALSE one. Within a computer, whole bytes are typically used to encode truth and falsity, since data are accessed (i.e., stored and retrieved) in byte-size chunks.

For instance, a computer might interpret a byte whose value was greater than or equal to a binary 128 (10000000) as TRUE, and one whose value was less than or equal to 127 (01111111) as FALSE. Note that it is the status of the high-order bit in this example that determines truth or falsity; the rest are irrelevant. In practice, any one of the bits of a byte might be singled out, though the high-order bit is the one commonly used. Computers make decisions on the basis of TRUE/FALSE data, and the results of those decisions may likewise be represented in TRUE/FALSE form.

INSTRUCTIONS

Instruction codes are yet another form of data that can be represented by a digital word. Such codes are essential in a computer because they tell the CPU (central processing unit) what to do and when to do it. A sequence of instruction codes representing the step-by-step solution of a problem is known as a program.

It is the job of a computer programmer to specify useful sequences of instruction codes and to see that they are correctly stored in the memory of a computer. Unfortunately, the boring sameness of binary codes makes programming in machine language (the computer's binary instruction set) a difficult chore. To make programming faster and simpler, high-level languages like BASIC, Fortran and PL/1 have been developed. These languages allow programs to be written as sequences of easily understood English-like statements and mathematical expressions. Eventually, however, all programs written in a high-level language must be translated to machine code, which is the only language a computer understands.

DIFFERENTIATING BETWEEN DATA TYPES

Thus far we have examined four different types of data likely to be found in a computer: numbers, alphanumeric symbols, TRUE/FALSE data, and instruction

codes. Given a particular byte of data, is it possible to tell which of the four categories it fits into simply by looking at it? The answer is no; there are no telltale signs that identify a data word as belonging to a particular class of information.

The central processing unit of a computer will interpret a data word in a manner consistent with what it expects that data word to be at a given instant. Sequence is everything. If the CPU fetches what it presumes to be an instruction, it will interpret that word as an instruction, regardless of what the programmer may have intended. One of the obligations of programming at the machine-code level, therefore, is seeing to it that data are stored in the proper order in a computer's memory. Using a high-level language like BASIC frees the programmer from such seemingly trivial details and allows him to concentrate on the larger issues before him.

PHYSICAL REPRESENTATION OF DATA

The ones and zeros we have been discussing are convenient abstractions, and nothing more. Inside the computer, data must be represented by some physical quantity in order for the computer to do any work. Most of the time, ones and zeros are represented by voltage levels, with the higher voltage corresponding to a 1, and the lower voltage to a 0. (Sometimes the situation is reversed. As long as a convention is agreed upon and maintained, either system is practical.)

Voltage levels are not the only means of encoding data. On magnetic disks or tapes, bits of information are stored as reversals in the orientation of a magnetic field within a thin layer of iron oxide. Optical-disc storage systems, which should be commercially available by the end of 1984, will store data in the form of tiny pits burned by a laser into a thin film of tellurium. We could go on, but the point has been made: like butterflies, our ones and zeros may metamorphose repeatedly on their journey through an information-processing system.

System Organization

Having gained an appreciation for the fine structure of computer organization, we are now ready to move up to the system level. Complex though the PC's hardware may be, it yields readily enough to analysis in terms of a small number of semi-autonomous function blocks. Figure 1-4 illustrates the important modules that make up the PC in schematic form. The rest of this section will discuss each module in turn and show how it contributes to the PC's overall operation.

SEMICONDUCTOR MEMORY

The Personal Computer uses two types of semiconductor memory, known respectively as ROM (read-only memory) and RAM (random-access memory). We will consider RAM first.

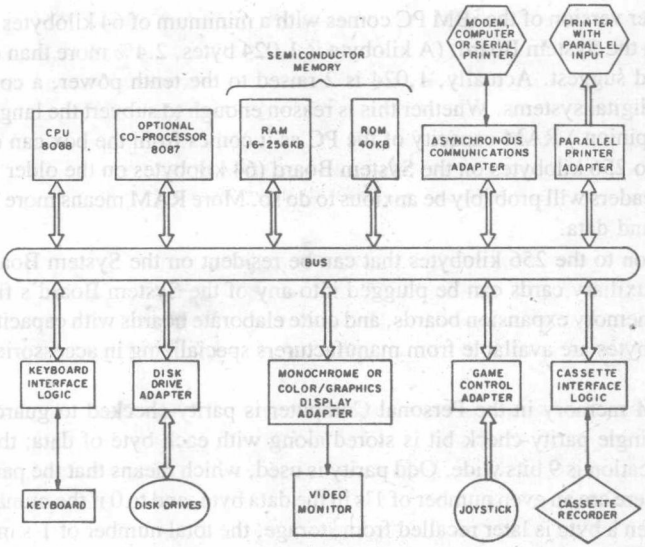

Fig. 1-4. Logical organization of the Personal Computer. A bus consisting of numerous, parallel electrical conductors shuttles digital information back and forth between the various modules that make up the system. The operation of these modules is coordinated and controlled by the CPU, or central processing unit.

Each individual storage site or address in RAM can be accessed independently of all other sites — hence the name random-access memory. It may be convenient to think of RAM as a cluster of mailboxes such as might be found in a hotel or office. Most likely each box will have a unique number (address) on it for the sake of identification. The letters contained in these mailboxes are analogous to the data stored in RAM. Obviously, we can stuff a letter into any box or remove it without concerning ourselves in the least with the contents of neighboring boxes. So it is with data access in RAM.

The foregoing observations probably sound unremarkable, but they are significant nevertheless. To see why, consider a sequential-access storage system like magnetic tape. The amount of time needed to retrieve a given parcel of data stored on tape depends on the separation between the section of tape initially under the tape head and the section that contains the data desired. Thus, data placement and access time are inextricably entwined. Of course, it really is unfair to compare a slow mechanical system like tape with electronic RAM. But even when sequential access is implemented in silicon, yielding an integrated circuit known as a shift register, the accessing of particular bits of data is still clumsy. For that reason, the main memory of a computer is always constructed of RAM.

RAM has one notable shortcoming, its volatility. Once power to the computer has been shut off, all data stored in RAM are lost. Thus, RAM must be supplemented by a more permanent form of storage if data are to be saved for future use. In the Personal Computer, long-term storage is provided by magnetic disks.

The newer version of the IBM PC comes with a minimum of 64 kilobytes of RAM installed on the System Board. (A kilobyte is 1,024 bytes, 2.4% more than common sense would suggest. Actually, 1,024 is 2 raised to the tenth power, a convenient number in digital systems. Whether this is reason enough to subvert the language is a matter of opinion.) RAM capacity of the PC as it comes from the box can easily be expanded to 256 kilobytes on the System Board (64 kilobytes on the older models), and most readers will probably be anxious to do so. More RAM means more space for programs and data.

In addition to the 256 kilobytes that can be resident on the System Board, extra RAM on auxiliary cards can be plugged into any of the System Board's five slots. IBM sells memory expansion boards, and quite elaborate boards with capacities of up to 512 kilobytes are available from manufacturers specializing in accessories for the PC.

All RAM memory in the Personal Computer is parity-checked to guard against errors. A single parity-check bit is stored along with each byte of data; thus, each memory location is 9 bits wide. Odd parity is used, which means that the parity bit is set to 1 if there are an even number of 1's in the data byte, and to 0 if the number of 1's is odd. When a byte is later recalled from storage, the total number of 1's in the data and parity bits is checked. If the number is odd, the data byte is assumed to be valid. But, if the total number of 1's turns out to be even, an error is flagged and the system shuts down, displaying a "PARITY ERROR" message on the video screen.

Parity errors are the result of defective memory chips, which must be located and replaced (usually by a service technician). Note that parity checking will not detect an error if *two* bits of a byte are defective, since reversing the status of two bits restores the original parity. However, double-bit errors have such a low probability of occurrence that they can be ignored.

We turn now to ROM, the second form of semiconductor memory. Like RAM, ROM is designed for random access. However, the data in ROM are stored there permanently at the time of the ROM's manufacture, and though this stored information can later be retrieved at will, it can never be altered. Furthermore, ROM storage is non-volatile; pulling the plug leaves ROM data intact.

Non-volatile read-only memory is useful as a means of storing programs essential to the operation of a computer system. The IBM Personal Computer comes with 40 kilobytes of ROM containing the BASIC language and a small portion of the computer's operating system. Both high-level languages and operating systems are important examples of system software, programs that give a computer its "personality." Without system software in its memory, a computer simply sits there, unresponsive and useless to its owner. We shall have occasion to say considerably more about system software later in this chapter.

While RAM allows data to be both written (stored) and read (retrieved), ROM allows only reading. RAM storage is volatile, so data vanish when system power is removed. In contrast, ROM storage is non-volatile and permanent.

THE BUS

As illustrated in Fig. 1-4, ROM and RAM are connected with the PC's central processing unit through a bus, or set of parallel electrical conductors. On some computers, the bus is a separate physical entity, a circuit board into which the other modules of the system are plugged. The Personal Computer, in contrast, was conceived as an integrated single-board design, so the bus together with many of the important modules of the system are co-resident on one large printed circuit board, the so-called System Board. The lines of the bus are routed to five 62-pin sockets on the System Board and are thus available to any auxiliary adapters that are plugged in.

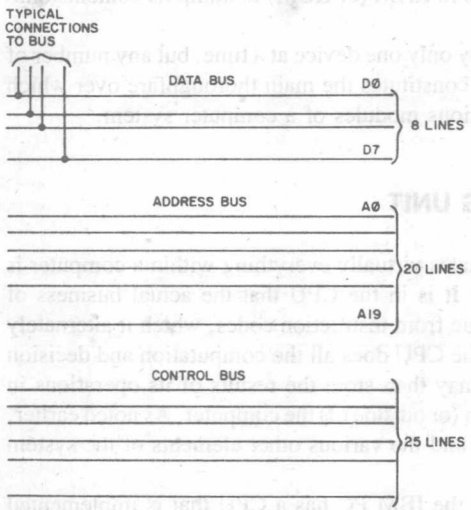

Fig. 1-5. The bus of the Personal Computer consists of three sub-buses that convey data, address, and control signals. Although the 8088 is a 16-bit microprocessor, it works with byte-size data words — hence, the 8-line data bus.

Figure 1-5 reveals that the bus is in fact composed of three separate subbuses: the data bus, the address bus, and the control bus. Data flowing into or out of memory does so by way of the data bus, which comprises eight lines, D0–D7. From this we can surmise (correctly) that memory in the PC is organized to accommodate byte-size data words.

The address bus is used by the CPU to select which location in memory will be read from or written to. Note that the address bus comprises 20 lines (A0–A19), which means that a maximum of one megabyte of memory can be addressed by the CPU. (We arrive at this figure by raising 2 to the 20-th power, which yields 1,048,576, or one megabyte. The same logic that causes 1,024 to be called a kilobyte prevails here.) Most PC owners will be content with far less than a megabyte of main memory, 64–256 kilobytes being more typical.

The remaining 25 lines of the bus are referred to collectively as the control bus. It is through these lines that the CPU exerts its control over the other elements of the system. For example, one line controls writing to memory, another reading from memory, a third writing to I/O ports, and a fourth reading from I/O ports. (As used

here, an I/O (input/output) port is an abstraction that refers to an addressable location which, when read from or written to, allows the CPU to communicate with devices external to the computer. More on this in a subsequent chapter.) Also comprised in the control bus are clock lines, which carry timing signals used by the entire system, and various lines that allow feedback to the CPU from other modules.

All three subbuses work in concert. For example, when the CPU wishes to store data in memory, it puts data on the data bus, puts the desired address on the address bus, and sends one line of the control bus LOW to cause the data to be stored at the designated location in RAM. Similarly, during a memory-read operation the CPU puts an address on the address bus, and then sends the proper line of the control bus LOW, causing the designated location in RAM (or ROM) to dump its contents onto the data bus.

Each line of the bus can be driven by only one device at a time, but any number of devices can be "listening." The bus constitutes the main thoroughfare over which data traffic is routed between the various modules of a computer system.

THE CENTRAL PROCESSING UNIT

The module that controls and coordinates virtually everything within a computer is the central processing unit, or CPU. It is in the CPU that the actual business of computing is performed. Taking its cue from instruction codes, which it alternately fetches from memory and executes, the CPU does all the computation and decision making specified by a program,. It may then store the results of its operations in memory or send them elsewhere within (or outside of) the computer. As noted earlier, all communication between the CPU and the various other elements of the system takes place via the bus.

Like virtually all microcomputers, the IBM PC has a CPU that is implemented almost entirely on a single integrated circuit known as a microprocessor. In building the Personal Computer, IBM engineers chose the 16-bit Intel 8088 microprocessor, which operates internally on 16-bit words but shuttles data back and forth over the bus in byte-size chunks. This may seem strangely inefficient, in comparison with 16-bit data transfers, and indeed it is. Compared to the 8086, which is identical to the 8088 in all respects except that it uses a full 16-bit-wide data bus, the 8088 is only about half as fast. Compared to an 8-bit microprocessor like the 8080, however, the 8088 is of course significantly faster.

In spite of the speed penalty it exacts, restricting the width of the 8088's data path to 8 bits does have certain advantages. For one, it allowed the PC to be developed using cheap and readily available 8-bit support circuits. These perform assorted useful functions like establishing an interface between the microprocessor and its data bus, or between the data bus and a device external to the CPU.

An even more compelling advantage is rooted in the relationship between the 8088 and its venerable 8-bit ancestor, the 8080. The assembly language instruction set of the 8088 is a superset of that used in the 8080. In other words, the 8088 can execute programs written for the 8080 if the assembly-language instructions (source code) are

re-assembled for the 8088. What this means is that after suitable translation, much of the vast array of software written, tested and debugged on the 8080 can now be run on the 8088. Do not assume, though, that *any* program that will run on an 8080-based computer will work on the PC. It must have been re-assembled for the 8088.

This, however, explains why only a year or so after the PC's introduction we see such a wide variety of familiar software (MicroPro's "WordStar," Ashton-Tate's "dBase II," etc.) being offered for the PC. Of course, a lot of brand new programs have been and will be written for the Personal Computer, but the quick availability of an established program base was certainly an important factor in IBM's decision to use the 8088.

The 16-bit 8088 is an improvement over the older 8-bit 8080 in several areas. Probably the most obvious difference between the two is that the 8088 can address a full megabyte of memory, while the 8080 was limited to just 64 kilobytes. (Only a few years ago, 64 kilobytes seemed like more than enough. How times change!) In addition, the 8088 boasts of improved error checking and higher speed. The latter is made possible in part by on-chip hardware for multiplication and division, two procedures that ordinarily are quite sluggish when implemented strictly in software.

The 8088 is capable of addressing a huge number of I/O ports — 65,536 to be exact — although only 512 of these are available for use in the PC. I/O devices and memory share the same address and data buses, but are activated by separate control lines. We shall have occasion shortly to discuss two important I/O devices, the parallel printer port and the asynchronous communications adapter.

THE 8087 CO-PROCESSOR

The computational power of the 8088 microprocessor can be extended by the addition of an optional 8087 Numeric Data Processor, which plugs into a vacant socket reserved for it on the System Board. Once in place, the 8087 functions as a co-processor that shares the same instruction stream as the 8088. Each processor responds only to the commands appropriate to it, and ignores all the rest. In effect, then, the 8087 and 8088 work in tandem, each one performing the tasks for which it is best suited.

Basically, the 8087 is a high-powered number cruncher capable of increasing the speed of floating-point numerical computation by a factor of as much as 100. Inside its ceramic case, the 8087 contains the hardware necessary for the multiplication, division, addition, and subtraction of 80-bit floating-point numbers. Also contained are assorted look-up tables for the standard logarithmic and trigonometric functions. Because all routines are implemented in hardware, extremely large numbers can be processed in a fraction of the time required by a 16-bit 8088 working alone.

Speed is not the 8087's only virtue. Its 80-bit internal data format allows numbers to be represented with extremely high precision. This can be especially important during a lengthy series of calculations, since numerical precision is degraded every step of the way because of the conversion from the decimal to the binary system. To end up with a reliable answer, it is necessary to carry out the intermediate steps of a