Notes on

The IBM Personal Computers

by

Brice Carnahan

James O. Wilkes

College of Engineering
The University of Michigan
Ann Arbor

1987

PREFACE

This book brings together in one volume information about three IBM Personal Computers (IBM/PC, PC-XT, and PC-AT)

. Emphasis is on use of the IBM Personal Computer as a stand-alone machine (not connected to a network) .

p2.25

The material is oriented toward:

(1) the IBM/PC machine configuration and software used in

,

(2) the IBM/PC-XT and PC-AT machine configuration and software used in

,

(3)

.

The topics covered should be of interest to anyone who has access to an IBM/PC, PC-XT, PC-AT .

Chapter 1 covers background material on the functional organization of digital computers, the most important features of the IBM/PC, PC-XT, and PC-AT hardware.

.

Chapter 2 describes the use of PC/DOS 3.1 (the most-used version of the IBM/PC disk operating system), including the most important of the DOS features and commands. PC/DOS allows the IBM Personal Computers to be used as stand-alone microcomputers.

.

Chapter 4 summarizes the procedures for compiling and executing FORTRAN-77 programs on the IBM/PCs using MicroSoft's MS FORTRAN compiler. Those interested in the FORTRAN-77 programming language should refer to our companion text, FORTRAN-77, with MTS and the IBM/PC.

The first appendix is a Quick Reference Guide to the use of DOS

Brice Carnahan
James O. Wilkes

September 1, 1987

TABLE OF CONTENTS

Chapter 4: FORTRAN-77 PROGRAMS ON THE IBM/PC

APPENDIX A: QUICK REFERENCE GUIDE FOR PC/DOS FILES, DEVICES AND COMMANDS

CHAPTER 1

THE IBM PERSONAL COMPUTERS

1.1 Introduction

The IBM Personal Computer, usually called the IBM PC or simply the PC, was introduced in the Fall of 1981 amid considerable fanfare. Apple Computer, Inc. had carved a niche for its Apple II computer as a personal computer with a significant educational market (particularly in the secondary schools) in the early part of the decade. However, IBM's entry into the microcomputer field, an area it had ignored before 1981, had the effect of "legitimizing" personal computers as tools for serious business and scientific applications. Software companies and vendors immediately began to focus attention and effort on developing a substantial body of business and educational application programs specifically for use on the new computer.

IBM's PC sales soon climbed past a million machines, far exceeding IBM's original sales forecasts. In early 1983 an improved version of the PC called the PC-XT (PC-eXTra) became available. In late 1983 the PC-Jr, intended for the home computer market, and a portable version of the original PC were introduced (both were later discontinued because of poor sales). In August 1984 a yet more powerful IBM personal computer, called the PC-AT (PC-Advanced Technology), was introduced. In 1985 a faster but completely compatible version of the PC-AT also joined the PC family.

Early in 1986, two new members of the PC family were announced by IBM, (1) a light-weight battery-powered portable (lap-top) version of the standard PC called the PC-Convertible, and (2) a much more powerful machine called the PC-RT (Reduced Instruction Set Technology). The latter machine is quite expensive by PC standards, and is intended for use as an engineering workstation, particularly for CAD/CAM (Computer-Aided Design/Manufacturing) applications. An updated version of the PC-XT, called the PC-XT286, was introduced later in 1986.

Except for the PC-RT, all of these IBM computers conform to what is called the IBM PC standard. The PC standard is an open machine architecture, in that the hardware performance specifications and characteristics are made available to other manufacturers by IBM. In particular, all details of

the hardware bus (the principal communication channel in the machine) are supplied to other manufacturers, who are encouraged by IBM to design hardware "add-on" circuit boards for performing a wide variety of machine enhancements (for example, for connecting special devices such as plotters and laboratory data acquisition equipment). Several special sockets called slots are incorporated into the principal IBM PC hardware board (the "motherboard"), to accommodate attachment of these non-IBM boards to the IBM PC bus.

One byproduct of this publicly available open architecture has been the proliferation of IBM PC look-alikes called "PC compatibles" or "PC clones" that support the IBM PC standard. These personal computers are made by many manufacturers and are essentially identical to, and sometimes faster than, their IBM counterparts, and in most cases are even made to look like the IBM products (similar appearance and colors for cases, keyboards, etc.). These computers are almost always less expensive than the comparable IBM products.

More than twelve million IBM PCs and compatibles have been sold since 1981. Currently, the standard IBM PC, XT, and AT and their clones account for about seventy-five percent of all desktop computer sales in the United States. However, units manufactured by IBM itself account for only about a third of these "IBM PC" sales; the more than twenty clone manufacturers (sales leaders are Tandy Corp., Compaq, Inc., and Zenith Corp.) account for the balance. Most of the IBM PC and compatible machines are being purchased by large corporate, governmental, and academic organizations, although many of the less expensive IBM models and clones are now being purchased for personal or home use.

In April 1987 IBM announced an entirely new family of personal computers called Personal System/2 (PS/2). This family of machines consists of four models (Model 30, 50, 60, and 80) of increasing power and capability. The highest performance of these computers (in particular, several versions of the model 80) will not be in full production until sometime in 1988, but most of the other models will be available in the latter half of 1987. The "IBM PS/2 standard" architecture will almost certainly become a new personal computer standard, though the IBM PC standard will still be important for many years to come (two other important standards are those for the Apple II and Macintosh personal computers from Apple Computer, Inc.).

While the PS/2 architecture is still "open" for board development by other manufacturers, IBM has designed the internals of the PS/2 machine to contain many "clone-proof" IBM-proprietary design features and hardware elements, some of which have been patented or copyrighted. This effort will probably discourage some of the technically weaker

companies from developing PS/2 clones, but is unlikely in the long run to stop successful copying of the PS/2 machines. The future market will almost certainly offer "PS/2 compatible" personal computers, and probably sooner rather than later.

This text is written primarily for engineering students . However, the topics covered should be of interest to any computer user with access to an IBM PC, PC-XT, or PC-AT for instruction and research. [Henceforth, we will use **IBM/PC** as a generic label for the IBM PC, PC-XT, and PC-AT microcomputers; when there is a need, the specific terms **PC**, **PC-XT** (or just **XT**) and **PC-AT** (or just **AT**) will be used.]

The text focuses on six major topics:

1. The IBM/PCs as general-purpose microcomputers.

2. PC/DOS 3.1. DOS (Disk Operating System) allows you to operate the IBM/PCs as stand-alone computers. IBM-compatible microcomputers normally use the essentially identical MS/DOS (available from MicroSoft, Inc.) operating system instead.

3.

4.

5.

6. Compilation and execution of FORTRAN-77 programs using: (1) Microsoft's MS FORTRAN compiler on the IBM personal computers,
. The FORTRAN-77 programming language is not described here, but is covered in detail in our companion text, FORTRAN-77, (with MTS and the IBM PC.

The versior of PC/DOS, described in Chapter 2 is widely used

The remainder of this chapter is devoted to a short description of the organization of digital computers in general and of the IBM/PC microcomputers in particular

1.2 Digital Computer Organization

Although digital computers are often viewed as very fast calculators, they are, in fact, rather general devices for manipulating symbolic information. In most applications, the symbols being manipulated are numbers or digits (hence the name digital computer), and the operations being performed on these symbols are the standard arithmetical operations such as addition and division. However, the symbols might also have nonnumerical values and the operations be nonnumerical in nature. For example, the symbols might be characters such as letters and punctuation marks, and the operations might result in the parsing of sentences for word and phrase structure.

A general-purpose computer can be instructed to accept, store, manipulate, and display virtually any kind of information that has been encoded in appropriate symbolic form. General-purpose computers are designed to solve essentially any "computable" problem. Computability has a rigorous meaning to computer scientists, but an intuitive understanding of the term is adequate for most programmers. A computable problem is one that can be stated unambiguously and for which an unambiguous terminating solution procedure or algorithm can be outlined, step by step. If the algorithm is in graphical form it is usually called a flow diagram; if it is in the form of a list of commands that can be processed directly by a computer, the algorithm is called a program.

Although we tend to view a digital computer as a unit, that is, as a single problem-solving machine, every computer is in fact a collection of inter-connected electronic (no moving parts) and electro-mechanical (some moving parts) devices, all directed by a central control unit. Fortunately, an understanding of computer operation and the ability to use a computer does not require detailed knowledge either of electronics or of hardware (the physical equipment) construction. An overall view of the organization of the computer with emphasis on function rather than electrical or mechanical details is sufficient.

Viewed functionally, all items of equipment associated with a digital computer can be grouped into four general categories:

1. Memory or Store
2. Input/Output units
3. Arithmetic (or Arithmetic and Logic) Unit
4. Control Unit

The overview of machine organization shown in Fig. 1.1 is typical of most currently available digital computers. Specific hardware interconnections and operating details vary from one machine to the next.

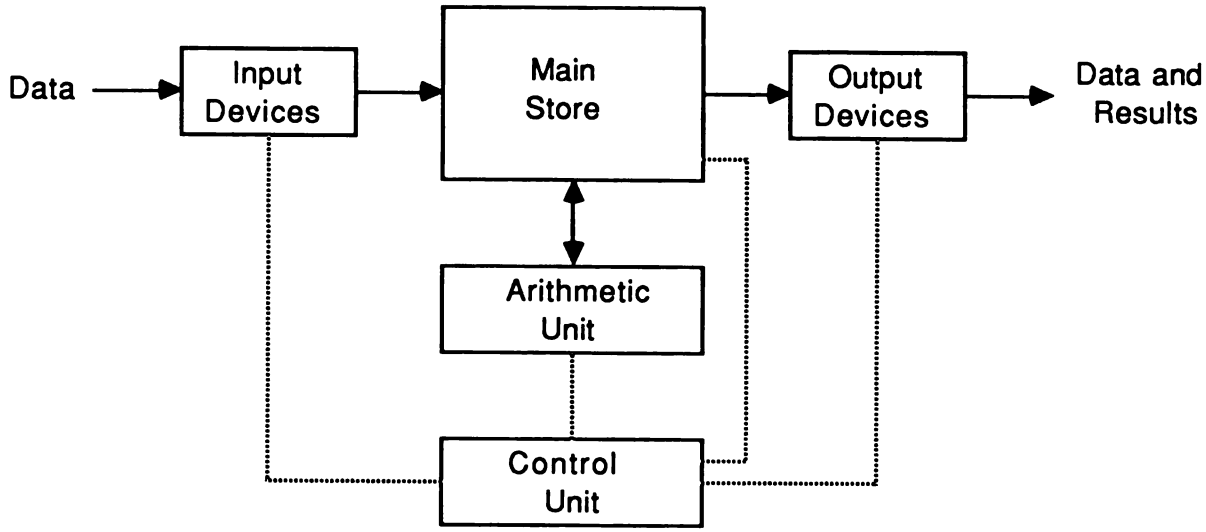

Figure 1.1

Organization of a Digital Computer
(lines between units show flow of information)

The functional roles of the major machine components in the operation of a digital computer are described briefly in the sections that follow, with special emphasis on the characteristics of the IBM/PC microcomputers.

1.2.1 Main Memory

The main memory, sometimes called the fast memory or principal store or just store, of a digital computer is simply a place to save or store encoded information that is to be manipulated (data) or that has already been produced (intermediate or final results) by parts of the computer responsible for computation (e.g., for adding two numbers together or for parsing a string of characters).

The main memory consists, at the most basic level, of electronic (no moving parts) elements that can be maintained in one of two possible physical states. For example, the element might consist of a tiny capacitor in a solid-state silicon electronic memory chip that can either be charged (one state) or uncharged (the other state). The charge state of a given capacitor can then be used to represent the

current value of any two-valued data item (yes/no), (+/-), etc.

Usually, the two state values of the storage element represent the binary digits (called bits) 0 and 1 (hence the term digital computer). Since almost any kind of information such as Greek letters, punctuation marks, and decimal numbers can be encoded as a unique sequence of binary digits, the memory can be viewed functionally as a place where representations of any kind of symbolic information can be stored and then retrieved later. Most digital computers use the binary (base two) number system for representing numbers and perform numerical calculations using binary arithmetic. Because of the two-state physical elements and the use of binary arithmetic for calculations, digital computers are sometimes also called binary computers.

The states of the individual main memory bits have a tendency change (e.g., charges associated with a capacitor have a tendency to dissipate), so the memory must be refreshed at fairly high frequency (many times per second) to insure that stored information is not lost. If power to the computer is interrupted, the refresh operations stop, and the stored information is lost. Hence, such memories are called volatile.

The main memories of even the smallest computers consist of thousands of such basic storage elements. To simplify the problem of locating a particular sequence of bits in the memory, most computer manufacturers, including IBM, group eight bits together in a collection called a byte. Each byte is assigned a numerical address to distinguish it from other bytes in the memory. The addresses are sequenced in the order 0,1,2,...

Current electronic memory chips usually contain approximately 16, 64 or 256 thousand bits, called 16 K, 64 K, and 256 K bit (kilobit) memory chips, respectively. Memory chips containing more than one million bits (1 megabit) are also now on the market, and researchers are currently working on techniques for manufacturing chips of even greater storage capacity.

Note: 1 K bits or bytes are not exactly 1000, but rather 1024, bits or bytes; the number of memory elements is a power of two because of the binary nature of the computer internally. Similarly, 1 M bits or bytes are not exactly 1,000,000, but rather 1,048,576 bits or bytes.

The individual memory chips can be used as building blocks to create quite large main memories. For example, IBM/PC computers are normally equipped with 128, 192, 256,

384, 448, 512, or 640 K bytes of memory. Some of the PCs and XTs in the College of Engineering have the "standard" 256 K byte memories (actually 256 x 1024 = 262144 bytes), adequate for most word processing tasks and solution of many scientific and engineering problems.

The eight bits in a single byte can be used to represent a number in the binary number system or the digital code for a character, or any of several other possibilities that won't be discussed here. Since there are 256 distinct combinations of eight ones and zeros that can be stored in one byte, it is possible to digitally encode in the byte format up to 256 distinctly different characters such as the upper and lower case Roman letters and punctuation marks. This is in fact done in the IBM/PCs, which have exactly 256 characters in their character sets. Henceforth, you can consider one byte to be equivalent to one encoded character.

Typically, when a number such as 3.14159 is involved in a calculation, the binary representation of the number is stored in several (usually four) bytes, which are subsequently treated as a single entity called a word. In IBM parlance, groups of two, four, and eight bytes are called a half word, full word (or just word), and double word, respectively.

Each byte of main memory can be used to store just one value (bit pattern) at a time. The value stored in a byte can be read back or retrieved without destroying it (non-destructive read out). However, if a new value is stored in a byte of memory, the previous value is lost (destructive read in), since the bit pattern associated with that byte on the memory chip is altered. This is analogous to the operation of a tape recorder for voice or music; recorded (stored) information may be played back without destroying it, but when a new signal is recorded over previous information, the earlier recording is destroyed in the process.

The memory access time is the time required to retrieve the content of one (sometimes more than one, depending on

the computer) byte of memory and is typically in the range of 5 to 500 nanoseconds (billionths of a second) for current memories. The main memory is called a Random Access Memory or RAM because the access time for all storage elements is the same, regardless of the storage address involved.

Another kind of memory, called ROM (for Read Only Memory) is often used in personal computers, as well. As the name implies, this memory can be read from but not written to. It is non-volatile, i.e., the stored contents are **not** lost when the computer is shut off. Frequently, special system programs, such as those to start up the machine automatically when it is turned on, are stored in ROM. The IBM/PCs have several ROM chips whose storage elements are incorporated into the overall addressing structure for the main memory. For example, there is a ROM in the PC (but not in the XT or AT) that contains a program for interpreting statements written in the BASIC programming language.

1.2.2 Secondary (Disk) Storage

It is important for computer users to have the ability to store substantial amounts of information (programs, documents, data) on a more or less permanent basis, so that it can be retrieved and modified over extended periods of time. There is clearly the need for the computer equivalent of the file cabinet.

The main memory of the computer is inappropriate for storage of such information, since (1) the memory is volatile, so information stored there is subject to loss (from a power failure, for example), (2) the total main memory capacity is fairly small, (3) electronic memories are relatively expensive, and (4) the computer will be used to solve many different problems, each requiring use of the same main memory.

One solution to this problem for small computers is the floppy diskette. The diskette used on almost IBM/PCs is illustrated in Figure 1.2 and consists of a 5.25 inch diameter circular mylar sheet covered with a magnetizable surface coating similar to that on a magnetic audio recording tape. The sheet is encased in a stiff paper sleeve.

The surface of the diskette can be used to store magnetic representations of bits and bytes, much as the surface of the magnetic tape in an audio cassette is used to store magnetic representations of sounds. When the diskette is inserted into a floppy disk drive, a hub clamps around

the center hole of the diskette (similar in function to the spindle/platter combination on an audio record turntable) and the diskette rotates inside the sleeve, typically at a speed of 300 rpm.

A recording head on a moveable arm can write information in magnetic form in circular tracks as the diskette surface spins under it. In addition, a playback head on the arm can also read previously stored information from the surface as the diskette spins. The read/write head accesses the diskette surface through an open oval slot in the sleeve shown in Figure 1.2.

The information on the diskette can be write protected (the disk drive will read but not write information on the diskette surface) by covering the write-protect notch (see Figure 1.2) with tape. This notch is similar to the "record protect" tab on a magnetic tape cassette, except that in audio cassettes the tab must be removed to protect against recording.

Personal computers made by different manufacturers normally have a unique diskette format, the layout of information on the surface of the diskette. Thus a diskette formatted for an Apple-2 computer cannot be used by an IBM personal computer. However, all IBM-compatible computers use the same format as the IBM/PCs manufactured by IBM itself; hence, diskettes formatted in the disk drive of one such machine can be used on all of the others.

In PC/DOS 3.1 (described in Chapter 2), the floppy disk drives used on most IBM/PCs format the magnetic information in 40 concentric circular tracks on each surface. Each track is divided into nine sectors of 512 bytes each, so that one surface of the diskette has a total capacity of 40 x 9 x 512 = 184320 bytes (180 K bytes). This is the total capacity of a single-sided diskette, meaning that only the top surface of the diskette is used for storing information.

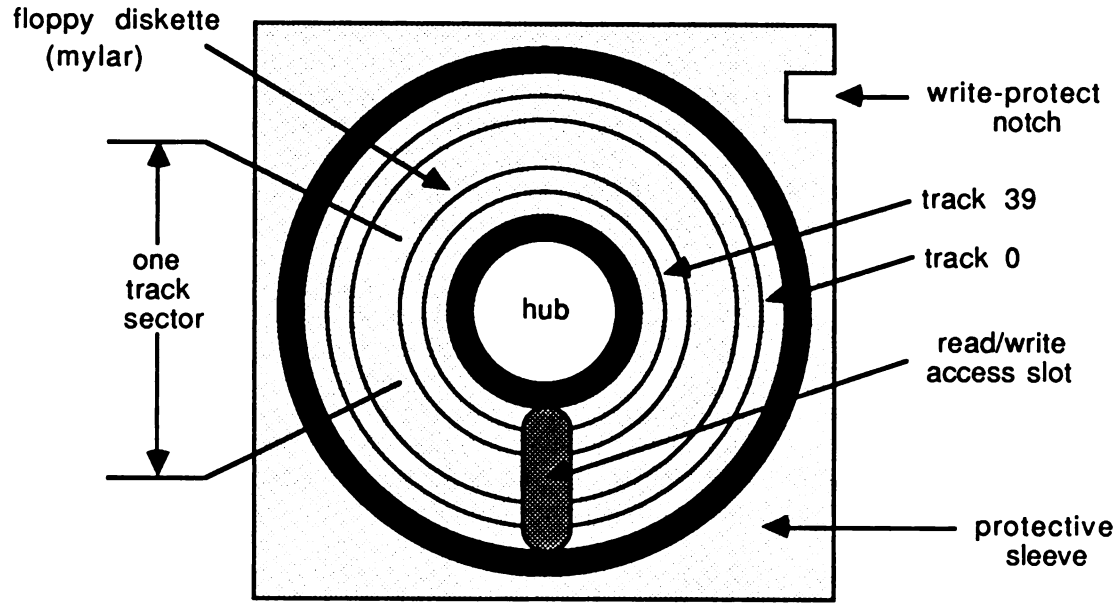

Figure 1.2

Floppy Diskette Storage Format
for the IBM/PC Computers

Most PCs and PC-XTs have double-sided floppy disk drives, which contain two read/write heads and use both surfaces of the diskette for storing information. These double-sided diskettes have a total storage capacity of 360 K bytes (actually 360 x 1024 = 368640 bytes) in DOS 3.1 format. Note that single-sided diskettes can be used on double-sided drives, but double-sided diskettes cannot be used on single-sided drives.

The PC-ATs normally have two double-sided floppy diskette drives. One is called a double-density, double-sided drive and is the same as described in the previous paragraph for the PC and PC-XT. The other is a quadruple-density, double-sided (sometimes called a high density) drive that reads and writes 5.25 inch diameter floppy diskettes with 80 tracks per surface, 15 sectors per track, and 512 bytes per sector leading to a total storage capacity of 1200 K bytes = 1.2 M bytes = 1228800 bytes. The latter drives operate at 360 rpm and can also read from and write to the lower density 360 K byte diskettes.

One IBM/PC model (the "convertible" lap-top portable) and all PS/2 models use a different standard floppy diskette that is 3.5 inches in diameter, encased in a hard plastic shell. On the convertible model, this diskette has double the storage capacity (720 K bytes) of the standard double-density 5.25 inch diskettes described above, on a single surface. This diskette appears to be the same as the diskette used on the Apple Macintosh computer, but is formatted differently, so the same diskette cannot be used on the two different machines.

The PS/2 Model 30 computer uses the same 3.5 inch diskette as the PC lap-top model. However, the other PS/2 models use a double-sided floppy format with a total storage capacity of 1.44 Mbytes. In time (probably a few years), the smaller diskette will become the predominant floppy disk standard. Since we are concerned with the IBM/PC computer family in this text, we will assume use of the 5.25 inch diskette throughout this text.

The PC usually comes equipped with two floppy diskette drives. The PC-XT computer is equipped with one floppy diskette drive and one hard disk drive, sometimes called a Winchester disk (named after an IBM research and development project). The hard disk drive contains two rapidly spinning fixed (they can't be removed) rigid metal disks mounted inside a special enclosure with an air filtering system to keep out dust. The disks, approximately five inches in diameter, rotate at about 3600 rpm.

Moveable heads read and write individual magnets (representing binary bits) in circular tracks on the surfaces, in a manner similar to the way information is read from and written to a floppy diskette. Because of the precise alignment of the heads and the dust-free atmosphere, 305 tracks can be written on each surface; each track contains 17 sectors of 512 bytes each leading to a total of 10370 K bytes of total storage (10,618,880 bytes). These drives are usually called 10 megabyte hard disks (a larger capacity 20 megabyte drive is also available as an option). Because information is more densely packed and the disk spins at much higher speed, information can be read from and written to the hard disk much more rapidly (at least ten times as fast) than is possible with a floppy diskette.

The hard disk drive supplied with the PC-AT has either 20 or 30 M bytes of storage, although disks of different capacities can be purchased from a variety of manufacturers.

The major advantages of the hard disk are the large storage capacity, short access time (the time required to position the read/write heads and to rotate the disk to access the first byte of interest, typically in the 20 - 80 ms range), and high data transfer rate (reading and writing

speed). The data transfer rate is controlled in part by a technical feature called the interleaf factor, roughly the number of complete rotations of the disk required to retrieve all information from on a single track. Ideally, the factor should be 1, but most hard disks are designed for lower transfer rates (e.g., the standard AT hard disk has an interleaf factor of 3).

The principal disadvantages of the IBM hard disks are that the disk storage medium cannot be removed and replaced with a different one (drives with replaceable disk packs are available from other manufacturers, however), and that a drive malfunction can result in loss of or damage to substantial amounts of stored data. The latter problem can be ameliorated somewhat by keeping "backup" copies of important information on floppy diskettes or magnetic tape casettes.

Although not yet in common use, optical disk drives, similar to audio compact disk players, will probably play an important role as mass storage devices in the future. An optical diskette is similar to the audio compact diskette (CD); about 600 Mbytes of information can be stored on one side of a single CD. Currently the CD is a read-only storage medium (the disks are called CD ROMs for "Read Only Memory"), suitable for storing large amounts of fixed retrievable information (e.g., encyclopedias, catalogs, dictionaries, engineering data bases). In some formats, video information can also be stored on the CD.

Like the audio CD, information on an optical CD is recorded in the form of small pits in a reflective coating on a substrate, and read by a laser beam aimed at the surface. The absence of a reflected beam indicates the presence of a pit, which can represent a binary bit (e.g., a 1).

Other optical disks with WORM (write once, read many times) capability are now available from IBM for use with their PS/2 computers. The IBM formatted drives allow the writing of up to 200 Mbytes of information on special 5.25 inch diameter removable diskettes encased in a hard plastic shell. The principal virtues of the optical diskette is that the user can store his or her own information (information is not prestored at the factory as is the case for CD ROMs), that the information can be added incrementally at different times, that the diskettes are removable, and of course, that the storage capacity is enormous (equivalent to over 500 standard 360Kbyte floppies). The principal drawback is that information cannot be erased to create space for new, different (e.g., corrected) information.

The development of *erasable* optical disks that can be both written to and read from an unlimited number of times is currently a subject of intense research and development activity by several major companies. Such optical CDs are likely to assume a very significant role as secondary storage media some time in the future. In the meantime, the Winchester disk will be the predominant mass storage device for personal computers.

1.2.3 Input/Output Units

The input/output units are, as the name implies, devices that allow the user to interact with the computer and *vice versa*. Large computers often have several such devices (card readers and punches, digital plotters, magnetic tape units, printers of various qualities, speeds, and capabilities, video terminals and keyboards, video screens with light pen or touchscreen inputs, etc.). The disk drives described in the previous section can function as input and output devices as well as storage units. In fact, almost any signal generating device (e.g., an electrocardiograph machine) can be used as an input device, and any equipment that accepts electronic signals (e.g., a motor-actuated valve in a chemical plant) can be made to function as an output device. In some computing installations, small computers are used as the principal input and output devices for larger ones.

For desk-top computers, the keyboard is the most common input device. In some microcomputers (e.g., the Apple Macintosh, some IBM PCs, and very likely many PS/2s) the keyboard is supplemented with a hand-held "mouse" that can be moved on a table top and be used for drawing pictures on the video display or for pointing at desired selections from displayed "menus" of alternative actions. Video monitors (TV-like screens), and small printers and plotters are the most frequently encountered output devices.

For the IBM/PCs, the principal input unit is the keyboard. The same keyboards were originally used on both the PC and PC-XT; the PC-AT keyboard is similar but with some minor key rearrangements. The PC-RT keyboard involves more drastic reconfiguration, and has become the "standard" keyboard for all PCs and the new PS/2 computers as well. All of the currently available keyboards are divided into four key sets known as the typewriter keys, the numeric keypad keys, the function keys, and the control keys. The original keyboard for the PC and PC-XT, used on most of the FEC and CAEN machines is pictured in Figure 2.1. The functions of the individual keys are discussed in detail in

Section 2.4, so the keyboard will not be described further here.

The major output devices for the IBM personal computers are the video monitor and the printer. In addition to their principal function as secondary storage devices, the disk drives are also important and frequently used as both input and output devices.

The monitor screen is the most-used output device for the IBM/PCs. Three different kinds of screen support hardware (circuit boards) are available from IBM for the IBM PC personal computers family (others are available from other manufacturers):

1. A monochrome monitor (circuit) adapter that produces very high resolution text displays on a special monochrome alphanumeric monitor; this board and monitor combination can display only the 256 characters in the IBM PC character set, and then only in specific positions on the screen (25 lines by 80 columns). Such alphanumeric monitors are usually used for word processing and other "business" applications.

2. A color graphics adapter (CGA) that can display medium- or high-resolution pictures (320 or 640 dots or pixels horizontally by 200 pixels vertically), as well as text (25 lines of 40 or 80 columns) on either a color or monocolor monitor (but not on the IBM monochrome monitor described above). A palette of up to four of the 16 available colors can be displayed at the same time in medium resolution only.

   The quality of the character display on the IBM standard color monitor is not nearly as good as on the monochrome alphanumeric monitor described above, and many users find that the 200 pixel vertical resolution produces only marginally acceptable graphics-mode displays.

3. An enhanced graphics adapter (EGA) that can display high resolution graphics (640 pixels horizontally by 350 pixels vertically), as well as text (25 lines of 40 or 80 columns) on an enhanced color monitor. Pallettes of as many as 16 of the available 256 colors can be displayed simultaneously. The quality of the character display is not quite as good as on the monochrome alphanumeric monitor described above, but much better than that on a color monitor using the standard color graphics adapter. High-resolution graphics-mode screen displays are of

significantly better quality than is possible on the standard color monitor.

With the introduction of the Personal System/2 computers, IBM changed its approach to graphical display by incorporating hardware elements called graphics arrays directly into the motherboard of all PS/2 models. The multicolor graphics array (MCGA) installed in the SR/2 Model 30 supports all of the IBM PC CGA and EGA color graphics and text display modes on the color monitors described above, without the need for special adapter boards. Several new graphics display modes are also supported for four new monochrome (black and white) and color monitors. Monochrome resolutions as high as 640 (horizontal) by 480 (vertical) pixels can be produced in 64 shades of gray. Color displays at this resolution are limited to two colors simultaneously from a palette of 256 colors.

The more advanced video graphics array (VGA) is installed on the PS/2 Model 50, 60, and 80 computers. It contains built-in support for all CGA, EGA, and MCGA display modes, and can display 16 colors simultaneously (from 256,000 possible colors) in the 640 by 480 format.

Most of the printers that are used with the IBM PCs are manufactured by the Epson Corp. or by IBM itself. A typical model is the Epson MX-80. It uses a closely-spaced set of fine wires to impact a ribbon and create characters from a rectangular matrix of dots and so is called a dot matrix printer. The printing speed is about 80 characters per second and printing is bidirectional (i.e., printing is done from left to right and then from right to left on alternate lines). The printer can write lines in 80 or 132 column format with either 6 or 8 lines per inch on standard 8.5 by 11 inch paper. With appropriate hardware and software, the printer can generate graphics symbols and reproduce the content of the monitor screen on a pixel by pixel (dot by dot) basis.

Another frequently used printer is the IBM Proprinter. It is faster (up to 150 characters per second) than the Epson MX-80, but is otherwise very similar in the quality of printed output.

In addition to these printers, the IBM/PC can support a wide variety of other dot-matrix, letter-quality (comparable to the print quality of a good electric typewriter) and laser printers. The laser printer operates on xerographic principles, and is similar to office copying machines. When properly maintained, the laser printer can produce printed output of near type-set quality, usually with a resolution of of 300 dots per inch. The FEC and CAEN laboratories are all equipped with Apple LaserWriters, one of the more popular laser printers.

1.2.4 The Arithmetic Unit

Referring now to the top portion of Figure 1.1, we see that a "computer" consisting of only a memory and input and output devices could accept information (data) from the user, store the information in the memory and presumably retrieve (the same) information for display on one or more output devices. The ability to save and retrieve the same information is useful in itself (e.g., information in a library card catalog). However, to solve a problem (or even to do selective retrieval), we would like to read data into certain bytes or words of memory, operate on this information in some planned way to produce results (which could be stored in other words of the memory), and finally to display the data and results on some output device.

The arithmetic unit (sometimes called the Arithmetic and Logic Unit or ALU) is the part of the computer that manipulates (performs arithmetical and logical operations on) information retrieved from the main store. The results of the individual operations are usually returned to the main store.

Information in the memory that is to be manipulated is first read non-destructively and passed to special memory locations in the ALU called registers. The ALU contains all the circuitry required to perform the standard arithmetical operations (addition, subtraction, multiplication, and division) and also many non-arithmetical operations (e.g., the shifting or digit-wise examination of numbers, the comparison of numbers for sign or relative magnitude, and isolation of bits associated with a particular character in a string of characters) on information brought from the main store into the registers. The results of manipulations performed in the ALU can be left in registers there if they are to be used as operands for additional operations, or written into selected locations (bytes) of the main store.

Each digital computer has a fixed number of distinctly different operations, called machine instructions, that the

ALU unit is capable of executing. The instruction set or repertoire that can be handled by the ALU is usually different from that of all other models of computers, although those for computers of a particular manufacturer are often similar.

With the addition of the ALU, the digital computer now begins to assume a meaningful form (see Figure 1.1). The machine can read data from its environment and enter them into the memory. The contents of various memory words can then be manipulated in the ALU using the individual instructions that the unit is designed to perform. The results of these operations can be stored in the memory along with the original data, and subsequently retrieved for display on the output equipment. The sequence of events is:

1. Read data into the memory via the input equipment.

2. Operate (in the arithmetic unit) on the data stored in the memory.

3. Store in the memory the results of the operations carried out by the arithmetic unit.

4. Retrieve the data and results from the memory for display on the output equipment.

1.2.5 The Control Unit

In order to process data in a sensible way and to produce useful results, a digital computer contains a control unit that supervises the sequence of activities taking place in all parts of the machine. This control equipment must decide:

1. When, and with which input device, to bring information into the memory.

2. Where to place the information in the memory.

3. What sequence of operations on information in the memory is to be performed in the arithmetic unit.

4. Where intermediate or final results of operations in the arithmetic unit are to be saved in the memory.

5. When, and on which output device, results are to be displayed.

The principal purpose of the control unit is to interpret individual machine instructions (those that are to be implemented by the ALU and others involving input/output and management of the store) supplied by the user. The control unit then sends appropriate signals to other parts of the hardware (e.g., the store, ALU, and input/output devices) so that they can perform the operations involved in each instruction.

With the addition of the control unit, (see Figure 1.1), we now have a machine that is capable of solving suitably stated and defined problems, given a list of individual machine instructions to be performed.

How does the machine user indicate what the machine is to do to solve his or her problem? First, the problem must be carefully defined and a step-by-step procedure or algorithm outlined for its solution. The development of the algorithm is the most difficult and important part of the problem-solving process, and is the responsibility of the computer user. The algorithm must then be transcribed into a list of individual machine instructions, so that the computer can carry out the sequence of operations required to solve the problem. This list, called a program, is then supplied to the control unit for processing, one instruction at a time, in sequence.

PDFp13 p1.05 <=

Only instructions from the machine's instruction repertoire can appear in the program supplied to the control unit; otherwise, the control unit would be unable to interpret the orders and send appropriate signals to other parts of the computer to carry out the desired operations.

When you use a pocket calculator, you assume most of the functions of the control unit in deciding which number is to be used next or which operation is to be performed next. Typically, a new number might be entered with the numeric keys every few seconds; operation buttons, for example for addition or division, would be pressed with about the same frequency.

However, when a program is being interpreted by the control unit of a digital computer, the all-electronic control unit is capable of handling individual machine instructions at an extremely high rate (tens of millions of individual instructions per second can be processed by the control unit of a large mainframe machine such as the IBM 3090; tens to hundreds of thousands can be executed per second on microcomputers in the IBM/PC family).

Somehow, the individual instructions of the program must be supplied to the control unit at a comparable rate if the computer is to function to its full potential. Because direct communication between the machine and its environment

involves the use of of slow electro-mechanical equipment, any approach that requires such contact continuously, (e.g., by supplying the instructions one at a time from a keyboard), would be impossibly slow.

A solution to this problem was first suggested by Burks (now Professor of Computer Science at The University of Michigan), Goldstine, and von Neumann in a classic paper, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*, published in a report of the U. S. Army Ordnance Dept. at the Institute of Advanced Study, Princeton, New Jersey, in June, 1946. (The paper is reprinted in Datamation, Vol. 8, Nos. 9 and 10, 1962.)

As already described in previous sections, the memory of a computer can be viewed as a place for storing any kind of digitally encoded information. Clearly, a computer program is a kind of information. Von Neumann et al. therefore proposed that the individual program instructions be appropriately encoded and then stored sequentially in words of the main memory of the computer along with data and results (in different locations in the memory, of course). The idea was adopted by the early computer builders and virtually all of the general-purpose computers now available are organized in this fashion. Such computers are known as stored-program computers.

Stored program computers in which the instructions are processed one at a time in sequence by a single processor operating on individual operands, such as the IBM/PC, are called Von Neumann machines. Other, newer stored-program computers variously called vector, parallel, and network machines having different architectures (usually incorporating more than one processor) are now coming onto the market. They are often designed to perform arithmetic operations on real numbers at very high rates, currently several hundred MFlops (Millions of floating point operations per second), and are sometimes called "supercomputers".

Since only binary digits may be saved in the main memory of a computer, the instructions in the program must be encoded in a digital form before being entered into the store. The digital code used is called the machine's language (different for each model of computer), and the encoded program is known as a machine-language program. The ultimate form of each machine-language instruction must thus be a sequence of binary digits (for example, 32 bits or four bytes) corresponding to the normal mode for storing information in the main memory of the computer.

Typically, some of the binary digits in the instruction would indicate what is to be done (addition, division, input, etc.); the code for what is to be done is usually

called the operation code or simply the opcode. Other digits in the instruction would correspond to the binary representation of the memory address (or addresses) of the operand(s) for the operation (for example, the location of a number that is to be one operand of an addition operation).

In order to write programs in the machine's language, one must be reasonably familiar with essentially all of the instructions (and their binary codes) in the instruction set of the machine. Memory allocation for instructions, data, and results is the responsibility of the programmer. Since a complicated problem might require many thousands of such individual machine instructions in the program for implementing the solution algorithm, the writing of an error-free program in the machine's language is very difficult and extremely tedious.

A single binary-digit error in an instruction will cause the control unit to execute the wrong (or no) operation (because the opcode is incorrect) or to perform an operation on an incorrect operand from the memory (because an address is incorrect). Obviously, easier ways of communicating with the computer are required to achieve reasonable programming efficiency. Some of the approaches now used to simplify the preparation of computer programs are discussed in Section 1.3.

1.2.6 The Microprocessor

As described in Sections 1.2.4 and 1.2.5, the control unit is responsible for interpreting the individual machine language instructions in the program stored in the computer's main memory, and then for ordering other parts of the computer to carry out the indicated tasks. The ALU is the part of the computer that manipulates (performs arithmetical and logical operations on) information retrieved from the main store. The results of the individual operations are usually returned to the store, although intermediate results may be left in a register temporarily for further processing.

The arithmetic and control units taken together are usually called the central processing unit (CPU) or simply the central processor or even just the processor. In personal computers (microcomputers) such as the IBM/PCs, the CPU is manufactured on a single electronic chip called a microprocessor.

The particular microprocessor used in the PC and PC-XT is the Intel 8088, manufactured by Intel Corp. The 8088 is called a 16 bit microprocessor because it can carry out its

instructions on 16 bit (2 byte) operands, although the data path for moving information into and out of the processor is just 8 bits (i.e., information is transmitted to and from the processor in single byte increments). Timing of microprocessor activity is controlled by an internal clock that cycles at a rate of 4.77 MegaHertz, i.e., 4,770,000 times per second. Individual instructions require different numbers of clock cycles for their completion, so the instruction processing rate for a program depends on the nature of the program, i.e., on the particular mixture of instructions that appear in the program.

Typically, the Intel 8088 processes of the order of 10000 instructions per second; for comparison, each of the four central processing units in the University's mainframe IBM 3090-400 computer can process over 10 million instructions per second (it is called a 10 mips machine). Arithmetic operations (such as addition) on floating point numbers (numbers with fractional parts) are very time consuming on the 8088, requiring of the order of several hundred microseconds each.

Computation on the PC and PC-XT can be speeded up considerably for some programs by adding an Intel 8087 math coprocessor that can perform arithmetic operations, in particular those involving floating point numbers (numbers with fractional parts), much faster than the Intel 8088. Floating point addition operations can be executed by the coprocessor at a rate of about 60,000 per second (60,000 flops). [Note: floating-point numbers are called REAL numbers in FORTRAN, as discussed in our companion text, FORTRAN-77 (with MTS and the IBM PC)]. The Intel 8087 coprocessor fits into special sockets prepared for it on the IBM PC chassis adjacent to the sockets for the Intel 8088.

•

The PC-AT and PC-XT286 is equipped with an Intel 80286 microprocessor, which is similar to the Intel 8088 microprocessor (its machine language is compatible with that of the 8088), but uses 16 bit (two byte) data paths and, in general, executes programs about three times as fast as the 8088. A companion math coprocessor, called the Intel 80287, is also available

•

The PS/2 Model 30 computer uses an Intel 8086 microprocessor, which is faster than the 8088 but slower than the 80286. The Intel 80286 microprocessor that is used in the PC-AT is also used in the PS/2 Models 50 and 60. The Model 50 and 60 processors are run at higher clockspeeds, however, and the machines are designed somewhat differently;

their computing throughput is significantly greater than that for the PC-AT.

The PS/2 Model 80 computer will use the next generation Intel 80386, which is a full 32 bit microprocessor, with perhaps fivefold performance improvement over that of the 80286. An optional Intel 80387 math coprocessor is also available.

1.3 Symbolic Computer Languages

The control unit of a digital computer (see Section 1.2.5) can only process programs that are stored in the main memory and that are written in it's own (i.e., the machine's) digitally encoded language. The machine's language is very foreign to humans accustomed to communicating in natural languages; better languages are needed for efficient preparation of computer programs.

The current solution to this communication problem involves the use of symbolic languages for describing algorithms. Here, symbols are used to represent memory addresses of operands (e.g., **X** rather than address **10100110**) and recognizable words or common arithmetic or algebraic symbols are used for representing operations (e.g., **READ** rather than opcode **10111011**, **+** rather than opcode **11001101**).

In the most basic of these symbolic languages, called assembly languages, each machine language instruction in a program is replaced by one symbolic language instruction. For example, addition of two numbers from symbolic addresses in the main memory named X and Y and storage of the result in symbolic memory address Z might require three individual assembly language instructions such as:

```
LOAD  1  X   (retrieve the first operand from address X
             and load it into register 1 in the
             arithmetic unit)
ADD   1  Y   (retrieve the second operand from address Y
             and add it to the current content of
             register 1)
STORE 1  Z   (store the result, left in register 1, as
             the new content of symbolic address Z in
             the main memory)
```

Note that these instructions are simply illustrative and do not correspond to those for any particular assembly language.

Of course, symbolic instructions such as those listed above could not be processed directly by the control unit of a computer, which only understands instructions encoded in its own digital machine language. Before such instructions can be interpreted by the control unit, they must be translated from symbolic to machine form. This could theoretically be done by hand, much as human translators translate articles written in one natural language (e.g., Russian) to another (e.g., English).

However, it is possible to write a translating program, called an assembler, that will cause the computer to translate from the symbolic form of a program (called the source program) to the machine language program equivalent (called the object program). The assembler must itself be available in machine language form (usually on disk storage), since otherwise it could not be loaded into the main memory and then executed by the control unit. Once the object program equivalent of the symbolic source program has been produced, it is usually stored on diskette or hard disk. In turn, the object program can be loaded into the computer's memory and processed directly by the control unit. The translation procedure is essentially the same as that for more widely used symbolic procedure-oriented languages, which will be discussed next.

The development of assembly languages resulted in a very great saving of time and effort by computer programmers. However, such languages still require the programmer to have detailed knowledge of the instruction repertoire and memory organization for the computer being used (different for each model or family of computers). This means that such languages are very machine dependent and programs written in them are not normally transportable from one model of computer to another.

Most users would prefer to write a computer program in a symbolic form that is more natural for humans (less machine oriented) and that allows use of fairly standard algebraic notation, English words, etc. A typical program to read two numbers (called X and Y) as data, add them together to get a new number (called Z), write out the data and result values on a printer, and then repeat the process for additional pairs of data values might appear in a form such as:

```
START   READ  X, Y
        Z = X + Y
        PRINT X, Y, Z
        GO TO START
```

Detailed knowledge of hardware and/or of the machine's language would be unnecessary for algorithms described in such a language. One line or statement of a source program

might require a large number of machine language instructions in the object program to perform the equivalent operations. In addition, the language might be used to describe programs for more than one computer, making it possible to interchange programs with other computer users.

Languages that allow machine-independent descriptions of algorithms are usually called procedure-oriented. Programs written in such symbolic languages cannot of course be executed directly by the computer; they must first be translated into equivalent machine-language programs. The program that implements the translation of a source program written in a procedure-oriented language into an equivalent machine language program is called a compiler; a compiler is used in a manner similar to that for assemblers mentioned earlier in this section.

As shown in Figure 1.3, a source program written in a procedure-oriented language is first translated (compiled) into an object (machine-language) program. In order to accomplish this, the compiler is first loaded into the computer's memory from one of the disk storage devices (the loading process is accomplished by another program called the loader), and then executed by the processor. The compiler must already be in machine-language form, of course. When the compiler (translating) program is in control of the processor, the source program is read as data (usually from disk storage); the output from the compiler is the object program, which is typically written onto disk storage.

The resulting object program is usually in a relocatable form, meaning that it can be loaded as a contiguous block of instructions into any block of available RAM memory. However, before the object program can be executed by the processor, it is usually necessary to "link" the program with other object programs from a program library (for example object programs that compute trigonometric functions or square roots that are needed by the object program). A system program called a linker is read into the machine's main store by the loader and combines all of the object programs (see Section 4.5 for more details) required into a single executable block of machine language code called the object module; usually the object module is written onto disk storage.

Once the object module is ready, it is loaded into the memory (by the loader program) and subsequently executed by the computer. Thus, the processing of a program written in a procedure-oriented language occurs in three major steps (ignoring, for the moment, the process of loading the compiler program, the linker program, and the object module into the computer's main memory):

1. Translation or compilation
2. Linking
3. Execution

FORTRAN-77 (or simply, FORTRAN) is one of the most popular procedure-oriented languages and is used extensively on the IBM personal computers . Since each different model of computer has its own machine language, there must be a compiler for each procedure-oriented language for every different kind of computer. Fortunately, FORTRAN compilers are available for most machines on the commercial market, so that FORTRAN programs can be processed on most digital computers now in use.

We will not describe the FORTRAN language in this text [see our companion text, FORTRAN-77 with the IBM PC]. However, procedures for using Microsoft's MS FORTRAN compiler on the IBM personal computers will be described in Chapter 4 .

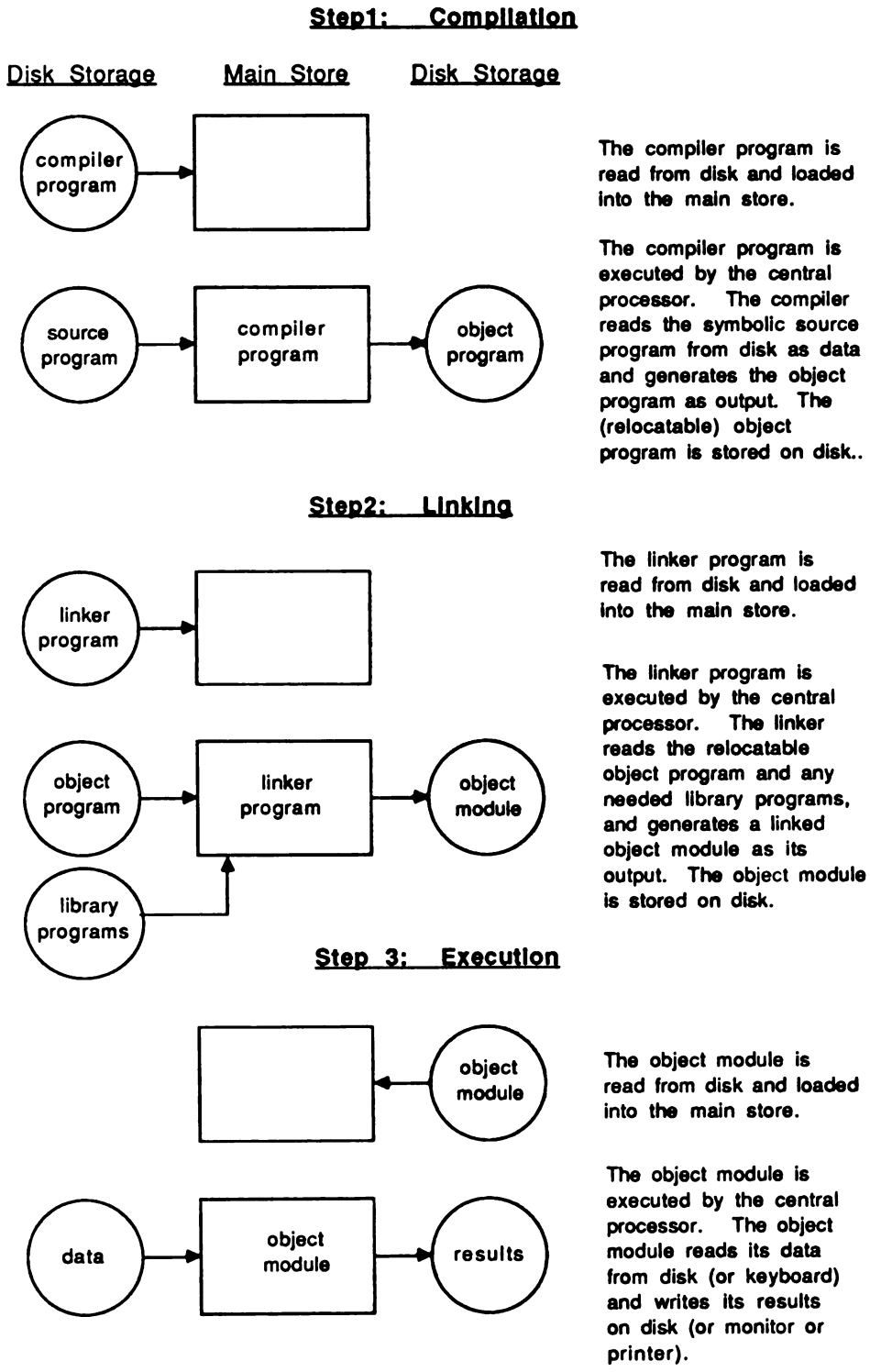

Figure 1.3

Compilation of a Source Program
and Execution of the Object Program

1.8 Operating Systems - PC/DOS

An operating system is a set of computer programs that manages the resources of a computing system and makes these resources available to the user. As the user, you need only enter relatively straightforward operating system commands to carry out computing tasks that otherwise would require quite complicated (and onerous) programming effort.

Sometimes there are several different operating systems available for the same computer. For example, for the IBM/PC there are, among others: CPM/86 from Digital Research Corp., the UCSD (University of California at San Diego) P System, and at least two UNIX-like (UNIX is an operating system for large computers developed originally at Bell Laboratories) systems available from Microsoft and AT&T. However, PC/DOS, written by Microsoft, Inc. and marketed by IBM (an essentially identical operating system for IBM/PC compatibles or clones, called MS/DOS, is marketed independently by Microsoft) is by far the most popular of the operating systems used on microcomputers based on the Intel 8088, 8086, and 80286 microprocessors. PC/DOS (or MS/DOS) accounts for more than 95 percent of the total

Version 3.1 of PC/DOS is described in detail in Chapter 2. A summary of PC/DOS 3.1 features and commands appears in Appendix A.

CHAPTER 2

THE IBM/PC DISK OPERATING SYSTEM - PC/DOS

2.1 Introduction

An operating system is a set of programs that is responsible for managing the resources of the computing system and for making these resources accessible to you for running programs, using the system input and output (I/O) devices, etc. You interact with the operating system programs by issuing relatively simple operating system commands.

A Disk Operating System (DOS) is an operating system that includes features that allow you to create and manage (read from, write to, modify, etc.) collections of related information called files, stored on secondary disk storage.

There are several different disk operating systems available for the IBM/PC, the most popular of which is an IBM product called PC/DOS. The system programs were written by Microsoft, a software development company, under contract to IBM. Microsoft markets a similar operating system named MS/DOS for control of several other "IBM-compatible" microcomputers ("clones") that use the Intel 8088, 8086, and 80286 microprocessors.

Eight versions of PC/DOS, labeled 1.0, 1.1, 2.0, 2.1, 3.0, 3.1, 3.2 and 3.3 have been released to date. PC/DOS 1.0 was made available when the IBM PC first came onto the market in 1981. Version 1.0 was superseded by the improved PC/DOS 1.1 in 1982. PC/DOS 2.0 was released in early in 1983 when the IBM PC-XT was first introduced. PC/DOS 2.1 was released in the Fall of 1983 at the time IBM debuted the PCJr, a small IBM/PC intended for home use (no longer in production). PC/DOS 3.0 was released in August 1984 with the introduction of the PC-AT. PC/DOS 3.1 was introduced in the spring of 1985, along with the IBM/Sytek PCNet Local Area Network. PC/DOS 3.2 was introduced in the Spring of 1986, with the announcement of the IBM PC Compatible portable, and PC/DOS 3.3 accompanied the introduction of the IBM PS/2 computers in the Spring of 1987.

From the user's standpoint, there is little outward difference (in terms of the system commands and features) between versions 1.0 and 1.1 and among versions 2.0, 2.1, 3.0, 3.1, 3.2, and 3.3. Version 3.1 is essentially version 3.0, but with added features for operating in a multi-user

IBM LAN environment. Version 3.2 supports the use of disk drives for either 5.5 inch or 3.5 inch floppy diskettes, but is little changed from version 3.1.

Currently, the most-used versions of the operating system are PC/DOS 2.1 for the IBM PC and PC-XT, and DOS 3.1 for the PC-AT. All of the features of versions 1.0 and 1.1 are present in version 2.0 and 2.1, but not *vice versa*. Versions 1.0 and 1.1 cannot be used to operate the PC-XT, but version 2.0 and greater can be used to operate both the PC and PC-XT. Versions 2.0 and 2.1 cannot be used to operate the PC-AT, but version 3.0 and greater can be used to operate either a PC or PC-XT.

. Any subsequent use of "PC/DOS" or simply "DOS" will refer to version 3.1, unless the version number is specifically attached.

Since some of the features of DOS are oriented toward the use of the hard disk on the PC-XT or PC-AT (not available on the PC) and others are of relatively little use to beginners, we have organized the chapter so that those parts of DOS that are most used and most straight-forward are described first (by and large, these are commands that are common to all versions of DOS); emphasis in the early parts of the chapter will be on use of DOS and the use of a stand-alone IBM PC
p2.25<= (not-connected to a network is also described). Later in the chapter, the use of the PC-XTs and ATs is covered, along with some of the more advanced features of DOS 2.0 and 3.1.

The format of this chapter is a combination of reference manual (a description of the features of DOS) and primer (how to use DOS). Material is presented in the order you are likely to need it as a beginner. In the interest of completeness, topics are covered in fair detail. You need not understand all of DOS to make effective use of the IBM/PC, so don't attempt to commit everything to memory right from the start. Some basic DOS concepts, e.g., files and devices are covered first. Then the individual commands are introduced with examples. You should be able to learn DOS at the computer by following the material from section to section. The best way to learn DOS is to use it.

2.2 DOS Files

The principal role of DOS is to manage information (programs, data, documents, records of virtually any kind) on disk storage (see Section 1.2.2). This information is organized into collections of related information called files. Hence, it is essential that an IBM/PC user have a clear understanding of file types, organization, naming conventions, etc. to make effective use of the operating system. Files can be broadly classified into three types, based on the kinds of information present in them:

text files — Files containing lines of characters in a special encoded form called ASCII (American Standard Code for Information Interchange). Text files can contain any kind of character information such as a source program (e.g., FORTRAN statements), data for a program, a medical record, a chapter of a novel, a letter to a friend, etc.

program files — Files containing programs in machine language (binary) form. Program files often contain object programs that have been produced by a compiler (e.g., a translated FORTRAN source program), and can be loaded into the memory of the IBM/PC and executed directly without further translation.

when they are object modules, then they

p1.25 <=

batch files — These are special text files whose lines contain DOS commands. DOS can process commands directly from a batch file, and treat them as if they had been entered from the keyboard. If a special batch file called an autoexec batch file (it must be named AUTOEXEC.BAT) is present on the master diskette (described later), then commands in that file will be processed immediately by DOS as soon as the DOS system program files are loaded, before you will be allowed to enter any DOS commands at the keyboard.

2.2.1 File Organization

File contents are organized in sequentially ordered lines. Each file line may contain up to 255 bytes. Since each character in a text file requires one byte of storage, text file lines can contain up to 255 characters; however,

text file lines are usually shorter (say 80 or 132 columns at most) so that they can be displayed on an 80 column monitor screen or printed on an 80 or 132 column printer.

text file or special text file

Each line in a^~~file~~ is assigned a sequential line number; the first line is given line number 1, the second line number 2, etc. DOS file line numbers are not permanently associated with the lines

. Instead, lines are numbered implicitly in the order 1, 2, 3, ... This means that if a new line of text is inserted between two already existing file lines or an old line is deleted, there will be an implied automatic renumbering of some lines in the file.

DOS files are stored as blocks of sequential bytes in the track sectors (see Figure 1.2) of disk storage media (floppy diskette or hard disk). DOS sets aside a few sectors on the diskette or hard disk for storage of a file directory that contains the names of the files on the diskette or hard disk and a file allocation table that contains information about the storage sectors occupied by each file. DOS maintains and updates this information as files are saved, modified, erased, etc.

Up to 64 files may be stored in a directory on a single-sided floppy diskette, up to 112 on a double-sided double-density floppy diskette, up to 224 on a double-sided quad-density diskette and up to 512 on the XT and AT hard disks, provided there is sufficient storage space for them. These limits may be increased by using subdirectories as described later in Section 2.21. For now, however, we will assume that there is a single directory for each diskette or hard-disk drive.

2.2.2 File Names

Each DOS file on a diskette or hard disk drive must be given a unique file name. The file name consists of one to eight characters. The name can be followed by a file name extension consisting of a period and then one to three characters. Several of the special characters [in particular, . " / \ [ ] : | < > + = ; and ,] are not allowed to appear in either the file name or name extension; we suggest that you use only the 26 Roman letters and 10 decimal digits when naming files. Upper and lower case letters are treated identically (e.g., the files **ALPHA.FOR** and **aLpha.FoR** are the same).

Technically, the file name and file name extension are distinct, but most users consider the "name" of the file to

include both the name and name extension. Usually, DOS requires that both file name and extension appear when making reference to a file in a DOS command. There are three exceptions to this rule, involving files with the file name extensions **.COM**, **.EXE**, and **.BAT**. In these three special cases, the file name extension need not appear (though it may be included, if you wish) when their corresponding program or batch files are to be processed (executed) by DOS. There are a few prohibited names that cannot be used as file names because they are reserved DOS device names, as described in Section 2.3. These are: **AUX**, **CON**, **COM1**, **COM2**, **LPT1**, **LPT2**, **LPT3**, **PRN**, and **NUL**.

Certain extensions are commonly used for DOS files (but ******* you may choose any name and extension that you like for your ******* own files):

.BAK	A backup file, a copy made in case the original file is accidentally or deliberately changed.
.BAS	A BASIC source program file.
.BAT	A batch command file (can be processed without specifying the file name extension).
.BIN	A file containing binary code (usually an object program).
.COM	A DOS command file (can be executed without specifying the file name extension).
.DAT	A data file.
.DOC	A document file (e.g., containing a memorandum).
.EXE	An executable file (can be executed without specifying the file name extension).
.FOR	A FORTRAN source program file.
.LIB	A library program file.
.OBJ	An object (machine language) program file.
.PIC	A screen image or graphics (picture) file.
.TXT	A text file

Examples: MYFILE
PROB1.TXT
PROGRAM.103
PROG3.FOR
BOEING.747
CHAPTER2.T

2.2.3 Generic File Names

Usually, you will want to refer to a unique file name in your DOS commands. Occasionally, (particularly when you want a list of only part of the directory of file names and when you wish to copy several files from one diskette to another) it is useful to refer to a group of files using a single name, called a generic or global file name.

Two "wildcard" characters, ? and *, can be used to represent any single character or group of characters, respectively. For example, the generic file name **PROB?.FOR** would include the file names **PROB1.FOR** and **PROBP.FOR**, but not file name **PROBLEM.FOR**, while **PROB*.FOR** would include all three.

Examples: PROG?.103

*.FOR	All files with the file name extension .FOR
FOR*.PRO	All files with extension .PRO whose names start with FOR
F??.*	All files with three character names beginning with the letter F and having any extension.
.	All files on the diskette, except for the DOS hidden system files (described later in Section 2.5.2).

Note: A generic file name that begins with an asterisk is assumed to include all possible file names; for example, the generic name *PROG.FOR is equivalent to *.FOR. Similarly, generic file name F*PROG.FOR is equivalent to F*.FOR.

2.2.4 File Name Directory

DOS assigns and maintains a small section of the diskette or hard disk for storing and maintaining a directory of all files present, including the file name, the file name extension, the number of bytes of storage occupied, the date of last change, and the time of last change. DOS also updates an associated allocation table

containing information about the tracks and sectors occupied by each file, the total space occupied by all files, and the space left for storage of additional files.

2.3 DOS Devices

DOS assigns a device name to each device attached to the IBM/PC. The most important devices are disk drives, the keyboard, the monitor, printers, and communication ports.

2.3.1 Disk Drives (Floppy, Hard, RAM)

DOS allows the user of the IBM personal computers to store, retrieve, erase, and otherwise manipulate the contents of files stored on floppy diskettes or hard disk drives. Each drive has a name called the drive designator that consists of a letter followed by a colon and has one of the following forms:

A:, B:, C:, D:, ...

The standard IBM PC, PC-XT, and PC-AT computers are usually equipped with two or three drives having drive designators as follows:

PC	Two double-density floppy drives	**A:** and **B:**
PC-XT	One double-density floppy drive	**A** or **B:**
	One hard disk drive	**C:**
PC-AT	One quad-density floppy drive	**A:**
	One double-density floppy drive	**B:**
	One hard disk drive	**C:**

However, additional floppy and hard disk drives can be attached to any of these machines, and DOS allows you to manipulate files on any of them simply by using the proper drive designator. In fact, DOS allows a part of the main store of the computer to be assigned as a RAM disk. Files stored in a RAM disk are treated exactly as files on a diskette in a floppy disk drive or on a hard disk would be. DOS assigns parts of the allocated RAM memory for storage of a file directory, file allocation table, individual files, etc., just as it would if the storage medium were on a floppy or hard disk drive.

The principal advantage of a RAM disk is that the disk access time (see Section 1.2.2) is extremely small (of the order of microseconds), since the files of interest are already available in the electronic (no moving parts) fast memory of the computer. This can result in a significant time saving for accessing frequently used files.

The principal disadvantages of the RAM disk are three in number: (1) main memory space assigned to the RAM disk cannot be used for loading and executing programs, so the effective size of the main memory from the standpoint of doing computation is reduced, (2) the RAM disk must usually be fairly small in size because of (1) so that only files of small or medium size can be stored, and (3) since the RAM memory is volatile and is present in the main memory, it must be viewed as a very "temporary" storage medium; all RAM disk files will be lost in the event of a system failure or power outage.

The latter problem is not too serious, since the files in the RAM disk are usually treated as scratch files (for temporary use only, like scratch paper), and RAM disk files that are of permanent value are invariably backed up with permanent copies on a floppy diskette or hard disk.

2.3.2 The Keyboard

The keyboard is assigned DOS device names

CON: or **CON**

for CONsole.

2.3.3 The Monitor

The monitor screen is assigned DOS device names

CON: or **CON**

for **CON**sole. While it may seem confusing to use the same device name for both the keyboard and the screen, the context of the command involved always makes it clear to DOS when **CON:** is being used for input (keyboard) or for output (screen).

Normally, when text is displayed on the screen it is formatted in up to 25 lines of 80 characters each; however, the screen display can be changed to 40 columns across using the MODE command described in Section 2.14.

2.3.4 Printers

The IBM personal computers can support as many as three different attached printers (e.g., one might have a dot-matrix printer for printing rough drafts, a letter-quality printer for business correspondence, and a laser printer for high volume output). The DOS names for the printers are:

First printer : **LPT1:** or **LPT1** or **PRN:** or **PRN**
Second printer: **LPT2:** or **LPT2**
Third printer : **LPT3:** or **LPT3**

In the examples used in this text, we will use **LPT1:** (Line PrinTer 1) as the DOS name for the printer.

The standard IBM or Epson matrix printer can be formatted to print using either 80 or 132 columns per line and either 6 or 8 lines per vertical inch.

You can change the printer format using the DOS command **MODE** as described in Section 2.14.

2.3.5 Communication Ports

The IBM computers can support up to two asynchronous communication ports (see Section 1.4) whose DOS names are:

First port : **COM1:** or **COM1** or **AUX:** or **AUX**

Second port : **COM2:** or **COM2**

2.3.6 Dummy Device

On occasion it is useful to have a dummy device that is nonexistent to which DOS assigns the name

NUL: or **NUL**

When treated as an input device, **NUL** is considered to be empty, and will generate an immediate end-of-file (the signal that no additional information is present to be read). As an output device, one can think of **NUL** as a bottomless "wastebasket" into which information can be dumped without ever appearing anywhere (on the screen, a printer, or disk file, for example).

2.4 The Keyboard Layout

A schematic of the IBM PC and PC-XT keyboard is shown in Figure 2.1 (the PC-AT keyboard is similar but slightly rearranged). There are four fairly distinct sets of keys in different areas of the keyboard.

1. A typewriter-like section of ivory-colored keys at the center called the typewriter keys.
2. A set of ivory-colored keys on the right side, called the numeric keypad keys.
3. A set of ten light gray keys on the left called the function keys.
4. A set of light gray keys immediately to the left and to the right of the typewriter keys called the control keys.

All keys with the exception of the **<Ctrl>**, **<Alt>**, **<Num Lock>**, **<Scroll Lock>**, **<Break>**, **<PrtSc>**, **<Caps Lock>** and the two **<Shift>** keys (with the open up arrows) are typamatic when struck individually (but not necessarily when used in combination with other keys), meaning that if they are held down for more than a half second or so, they automatically repeat. Combination keystrokes (when more than one key is held down at the same time as described in Section 2.4.4) involving the **<Ctrl>**, **<Alt>** and **<Shift>** keys are also typamatic.

Different IBM/PC programs interpret some of these keys differently than does DOS. Therefore, you must know which program is running on the IBM/PC to know which keys to use to perform particular functions. The keyboard interpretation for **DOS** is highlighted in

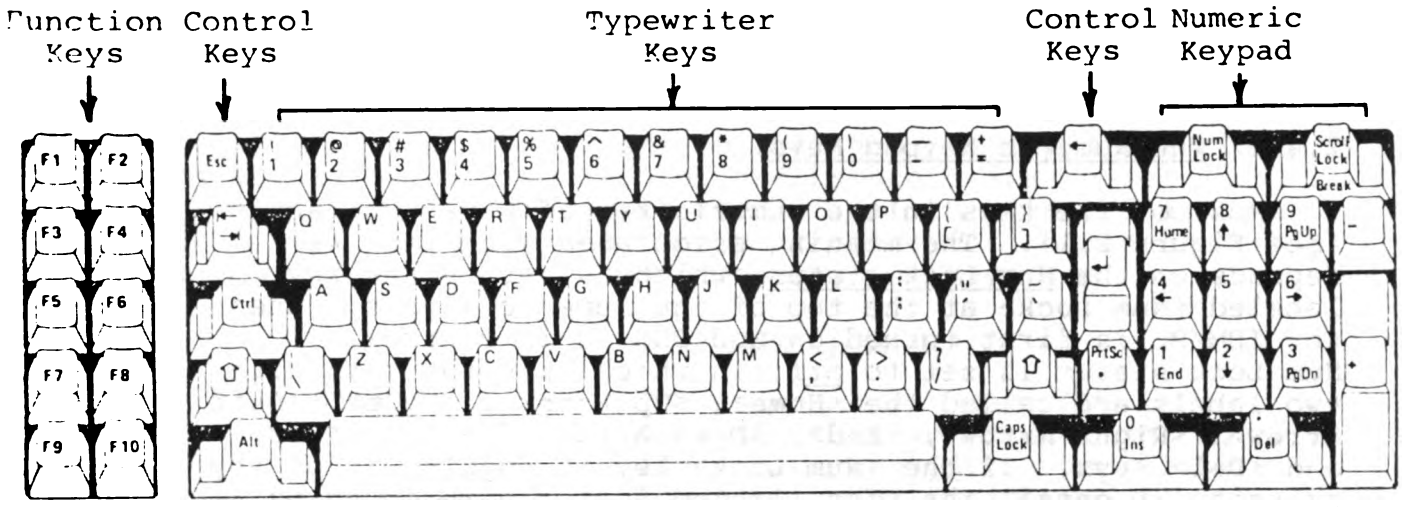

Figure 2.1

The IBM PC and PC-XT Keyboard

the list and keyboard chart in the Quick Reference Guide included as Appendix A.

For now, we will discuss the keyboard in general, and the key interpretation for DOS, in particular.

2.4.1 The Typewriter Keys

The typewriter keys in the center of the keyboard are laid out in the usual typewriter format (QWERTY..., etc.). A few of the keys for punctuation marks and special characters are in somewhat different locations than on the standard typewriter. In particular, the two **<shift>** keys (with the open up arrows) are in rather unusual positions, and take some time to get used to.

There is a key labeled **<Caps Lock>** immediately to the right of the space bar that functions similarly, but not identically, to the shift-lock key on a standard typewriter.

This key serves to toggle Caps Lock status on and off. When the IBM/PC is first turned on and DOS is in control (described in Sections 2.6 and 2.24), Caps Lock status is set to off and the <**shift**> keys function like the shift keys on a standard typewriter. When Caps Lock status is toggled to on, the letter keys produce upper case letters; if the <**shift**> key is then used, the letter keys are shifted to lower case. The Caps Lock status has no affect on non-alphabetic keys.

2.4.2 The Numeric Keypad Keys

Most of the keys in the numeric keypad have two labels (see Figure 2.1). The meaning associated with each key depends on the Num Lock status, which is set by the key labeled <**Num Lock**> at the top of the numeric keypad. When the IBM/PC is first turned on and DOS is in control, Num Lock status is set to off, in which case the keys with two labels are called the <**Home**>, <**Up Arrow**>, <**PgUp**>, <**Left Arrow**>, <**Right Arrow**>, <**End**>, <**Down Arrow**>, <**PgDn**>, <**Ins**> and <**Del**> keys. If the <**Num Lock**> key is toggled to on (by pressing it once), the keys (listed in the same order as above) represent the digits **7, 8, 9, 4, 6, 1, 2, 3, 0** and the **decimal point.**

Since the numeric keys and the decimal point (the same as the period) are duplicated in the typewriter section of the keyboard, there is no need for the numerical interpretation of the numeric keypad keys. The Num Lock status should almost always be left in the off position; an exception might be when you are entering a very large amount of numerical information. Should you find that these keys are generating digits when you don't want them to, simply press the <**Num Lock**> key once to toggle Num Lock status back to off.

When Num Lock status is off, these keys are widely used by various programs for positioning the cursor, a lighted rectangle displayed on the screen to indicate where the next typed character will appear.

[Note: When Num Lock status is off, the digits on the numeric keypad keys can still be generated by holding down one of the <**Shift**> keys when typing the numeric keypad keys.]

2.4.3 The Function Keys

The set of ten keys on the left side of the keyboard labeled **F1** through **F10** are called the function keys. The significance of these keys varies from program to program.

2.4.4 The Control Keys

The keys to the immediate left and the immediate right of the typewriter section of the keyboard are called the control keys. Starting to the left of the typewriter keys from top to bottom, and then proceeding to the right of the typewriter keys from top to bottom, the keys are:

Key	Name and Function
<Esc>	The Escape key. When DOS is in command, this key causes the current line to be erased. A backslash appears on the screen to indicate that a deletion has occurred, and the cursor moves to the next line. Other programs will interpret this key differently
<Tab>	This key is the one with two arrows immediately below the **<Esc>** key. When used without the **<Shift>** key, the function of this key is similar to that of the **Tab** key on a typewriter, provided that the program in control at the time recognizes tab stops. The shifted key (when the **<Shift>** and **<Tab>** keys are held down simultaneously, corresponding to the left arrow) normally does nothing.
<Ctrl>	The Control key is always used in combination with one or more other keys, in which case the key combination is written (in this text) as **<Ctrl-xxx>**, where **xxx** represents another key. For example, when the **<Ctrl>** key is used with the **<F1>** function key, the key combination is called control-function 1 and will be written as **<Ctrl-F1>**. The interpretation of a

key combination depends on the program in execution at the time.

<Alt> The **Alternate control** key. This key is similar to **<Ctrl>** and is also used only in combination with one or more other keys **<xxx>**, in which case the key combination is written as **<Alt-xxx>**.

<Backspace> This key (with the backward arrow at the top of the keyboard on the right) functions like the backspace key on a typewriter, in that the screen cursor moves left one space. (more than one, if the key is held down). However, the character that was originally immediately to the left of the cursor is erased; all characters originally at and to the right of the cursor position are shifted left.

<Enter> This key (the large one with the back arrow immediately below **<Backspace>**) is called **Enter**, **Return**, or **<cr>** (for carriage return). These terms are used interchangeably in the PC literature and throughout this text. The function of this key is similar to that of the RETURN key on a typewriter. Most of the lines that you enter at the keyboard will be terminated with this key.

<PrtSc> When unshifted, this key generates an asterisk *. When shifted, the key combination **<Shift-PrtSc>**, causes DOS to **Print** a copy of the current **Screen** image on the printer. The printer must be attached and turned on, and the ON LINE indicator on the printer console must be lit.

When used with **<Ctrl>**, that is, **<Ctrl-PrtSc>**, the key combination serves as an on/off toggle for sending any subsequent lines that are displayed on the screen to the printer (called "echo" printing) as well. When DOS is originally loaded, this printing switch is set to off. Once printing is turned on

with **<Ctrl-PrtSc>**, all lines subsequently displayed on the screen will be printed until printing is turned off with the same key combination. As with **<Shift-PrtSc>**, the printer must be turned on and the ON LINE indicator on the printer must be lit.

<Ins>

The Insert key is not strictly speaking a control key, since it is part of the numeric keypad. Nevertheless, when Num Lock status is off (controlled by the **<Num Lock>** key), the **<Ins>** key serves as a toggle for control of Insertion mode.

When insertion mode is on, characters typed on the keyboard are normally inserted at the cursor position in the current line. Any characters already present on the line at the cursor position and to the right will be shifted one column to the right. When insertion mode is off, the typed character replaces any character already existing at the cursor position.

Note: this interpretation of insertion mode behavior applies to most programs

, but not when DOS itself is expecting input (i.e., awaiting a command)! In that case it has a different (and rather obscure) use which will not be discussed here.

<Num Lock>

Controls the Num Lock status of the numeric keypad, as described in Section 2.4.2.

When used with **<Ctrl>**, that is, **<Ctrl-NumLock>**, any writing to the screen will be suspended temporarily (this enables you to examine information being displayed too fast to read, without having it scroll off the top of the screen). When you want to resume display of additional information, simply press any key.

<Del>	When Num Lock status is toggled to the off position using the **<Num Lock>** key, the Delete key causes the character at the cursor position to be erased. Characters already on the line to the right of the cursor position move left one column.
<Scroll Lock> or **<Break>**	This key is used in combination with the **<Ctrl>** key to interrupt whatever the computer is doing at the time. Control normally returns to DOS, which then waits for its next command to be entered. It should be noted that this function can be disabled by some programs , so the key combination **<Ctrl-Break>** will not always stop program execution.

When control key combinations involving **<Ctrl>** and **<Alt>** are used, the control key should be pressed and held down slightly before the accompanying key is pressed. Both keys should then be released. If several key combinations are to be entered in sequence, the **<Ctrl>** or **<Alt>** key can be held down continuously while the other keys are typed. Note that this is completely analogous to the way the **<Shift>** key is used on a standard typewriter.

One final key combination

<Ctrl-Alt-Del>

should also be mentioned. This three-key combination causes all computing activity to cease, including execution of the DOS system programs. The computer is warm booted, which means that the system is reset; the computer behaves as if it had just been turned on (even though it is already turned on!). This restart procedure is discussed later in Section 2.6.2.

2.4.5 Summary of Keys for Special DOS Functions

The following is a list of some of the more important keys for implementing special DOS service functions:

Function	Key(s)
Interrupt current activity	**<Ctrl-Break>**
Print the current screen display	**<Shift-PrtSc>**
Echo print future screen lines	**<Ctrl-PrtSc>**
Stop echo printing of screen lines	**<Ctrl-PrtSc>**
Stop screen display temporarily	**<Ctrl-NumLock>**
Resume screen display after stop	any key
Reset the system	**<Ctrl-Alt-Del>**

2.4.6 Summary of Keys for Editing DOS Input Lines

When you type in an input line for DOS and terminate it with the **<Enter>** (**<Return>**) key, DOS saves a copy of the line as the new contents of a hidden one line input buffer in the main memory. You can retrieve characters from this buffer to prepare the next input line for DOS (the line is displayed on the screen) by using the following special DOS Editing Keys:

Function	Key(s)
Display next character in buffer	**<F1>**
Display all characters left in buffer	**<F3>**
Display all characters in buffer up to 'x'	**<F2-x>**
Delete all characters up to 'x'	**<F4-x>**
Insert character(s) into input line	**<Ins>**
Delete characters from input buffer	**<Del>**
Replace buffer contents with input line	**<F5>**
Erase all characters in input line	**<Esc>**

The **<Esc>** key causes the input line to be erased, but does not affect the content of the input buffer.

To illustrate, suppose that you have entered the following characters at the keyboard:

PRINT FILE1.FOR FILE1.DAT DOCUMENT.5<cr>

Here, <cr> means "carriage return", i.e., the **<Enter>** or **<Return>** key. DOS would process the command and also copy all of the characters in the input line (except for the carriage return) into the input buffer.

Then, if you entered the following characters in sequence

<F2-1> 2 <F1> BAS*b* **<F4-M> <F1> <Ins> OD <F3>**

(here, *b* is a blank), the following new input line would appear on the screen:

PRINT FILE2.BAS MODENT.5

2.5 Diskettes

When you want to use a standard IBM PC , you should have available two double-sided floppy diskettes. The first, called the master diskette, contains the disk operating system (DOS) and other important system or course-related programs .

(If you are using a standard IBM PC , the master diskette is the DOS system diskette that usually is purchased with the computer). The second diskette will eventually contain your own programs, data, and results. We will call this one your personal diskette (you may have more than one) throughout this text.

When you use a PC-XT or PC-AT , the hard disk normally functions as the master disk drive, and you will need only your personal diskette for saving your own files.

2.5.1 Disk Drive Assignments

The typical IBM PC comes equipped with two floppy disk drives, named **A:** and **B:** for the left and right drives, respectively. The typical IBM PC-XT has one double-density floppy disk drive on the left named either **A:** or **B:**, depending on the context, and a hard disk drive on the right named **C:**. The standard PC-AT comes with a quad-density floppy disk drive **A:** on the top half of the right side of the computer case, a double-density floppy disk drive **B:** situated immediately below drive **A:**, and a hard disk drive **C:** on the left. However, as described in Section 2.3.1, DOS can support and manage several different floppy, hard, or RAM disk drives associated with a single IBM personal computer. Hence, a large number of different configurations and disk assignments is possible.

In order to identify a file on a diskette in one of the disk drives, the file name is preceded by the appropriate drive designator. For example, a file named **MYFILE.FOR** on Drive **B:**, is written as

B:MYFILE.FOR

in DOS commands.

The drive designator is not required (but may be used if desired) for files present on the default drive (floppy Drive **A:** when DOS is first loaded into the IBM PC memory), described in Section 2.5.4; the default drive can be changed at any time by the user as described in Section 2.7.3.

On the PC-XT or PC-AT the hard disk Drive **C:** is usually the default drive, as described in Section 2.24.

2.5.2 The Master Diskette for the IBM PCs

Most of the files on the diskette are program files including the following:

1. **AUTOEXEC.BAT**, containing a batch file with DOS command lines that are processed by DOS immediately after the DOS program files are loaded (before any DOS commands can be entered by the

user). This batch file contains text (the DOS commands) so it is not a program file.

p2.27<=

2. **ASTCLOCK.COM**, containing a program file responsible for reading the date and time from an external battery-powered calendar/clock, and for resetting the computer's internal clock.

8. Two hidden files named **IBMBIO.COM** and **IBMDOS.COM**, containing the most important of the DOS system programs. These files appear to be unnamed, since they do not show up in the file directory for the master diskette, and are not accessible to the user.

9. **COMMAND.COM**, containing the DOS program responsible for interpreting each DOS command as you key it in, and then for calling on the appropriate part of DOS to actually process or carry out the command.

10. Under normal circumstances (for a stand-alone IBM PC), several other DOS system files with name extension **.COM** (for executing DOS transient commands, described in Section 2.7.2) would also appear on the master diskette.

The master diskette is always inserted into Drive **A:** (the left disk drive) of the IBM PC. We will sometimes refer to Drive **A:** as the master drive since the master diskette resides there. The master diskette is write protected to insure that the system program files cannot be modified or erased accidentally.

2.5.3 Your Personal Diskette

You should buy one or two blank double-sided soft-sectored diskettes

Diskettes are available at most local book stores and computer stores. You will use these personal diskettes to store your own text and program files.

Be sure to write on the diskette label with a felt pen .
Don't fold, spindle, or mutilate your diskettes! In addition, keep diskettes free from dust, extreme temperature (particularly heat) and magnetic fields such as those associated with television sets, stereo speakers, and telephones (magnets drive the bell). Once a diskette is damaged, it is usually impossible to recover files from it. Thus it is a good idea to copy valuable files to a second backup diskette as a precaution against loss of or damage to your primary personal diskette.

2.5.4 The Default Drive - IBM PC

When DOS is first loaded into the IBM PC's memory as described in the next section, it immediately assigns Drive **A:** as the default system drive, meaning that:

1. Files referenced in DOS commands without a drive designator (**A:** or **B:**) are assumed to be present on the diskette in Drive **A:**.

2. References to files on the diskette in Drive **B:**, (e.g., your personal diskette), must be preceded by the file designator **B:**, as in **B:PROB1.FOR**.

DOS allows you to change the default drive as described later in Section 2.7.3.

For a stand-alone IBM PC, all DOS system program files, including those for the DOS transient commands, described later in Section 2.7.2, are also assumed to be present on the diskette in Drive **A:**.

2.6 Getting Started on the IBM PC

Now that you have some acquaintance with DOS, DOS files and devices, and with the layout of the keyboard and the functions of the various keys and key combinations, you are ready for your first use of DOS on the IBM PC. Real familiarity with the various features of the IBM PC and DOS only comes with practice. So don't be too concerned if you haven't followed all of the material to this point.

Remember that it is impossible for you to damage the computer hardware by entering something incorrectly at the keyboard. Of course, the machine is vulnerable to physical abuse (spilled coffee, a hot or dirty environment, etc.). In particular, the disk drives must be treated gently. **Never force a diskette into a disk drive** (there may already be a diskette in the drive!) **or open the disk drive hatches** (to remove or insert a diskette) **while the red read/write light on the front panel of the drive is lit.** Doing so may damage the disk drive mechanism, the read/write heads, or the diskette surface.

If you are operating a stand alone IBM PC , then you should have two diskettes in hand, (1) the DOS system master diskette and (2) your personal diskette for storing programs, data, etc.

Before attempting to use the IBM PC, turn all of the power switches (main cabinet, printer, monitor) to the off position. Check to insure that there is paper in the printer.

It is very important that the power to the printer be turned off when inserting paper. If it isn't, you may strip the gears in the paper feed mechanism.

2.6.1 Inserting the Master and Personal Diskettes

Now, open the covers to the two disk drives by flipping them up. **Drive A:** (the master drive) is the one on the **left**. **Drive B:** is the one on the **right**. Check to see if there is already a diskette in one or both drives. A diskette in Drive **A:** is probably a master diskette. A diskette in Drive **B:** probably belongs to another student. In either case, give the diskette(s) to the laboratory monitor.

Next, take the master diskette with the label up and pointing toward you (so that the oval slot that exposes part of the diskette surface points away from you) and insert it gently into master Drive **A:**. Insert your personal diskette with the same orientation into Drive **B:**. Carefully close the covers of both drives.

2.6.2 Turning on the IBM PC and Booting DOS

Now, turn on power to the main cabinet of the IBM PC, the monitor, and the printer. The power switches for the main cabinet and for the printer are on the right-hand side of the unit. [The switch for a Zenith monitor is on the lower right corner of the front panel. The on-off switch for IBM monitors is the topmost of the three controls to the right of the screen.] The startup procedure that now follows automatically is called cold booting the operating system DOS.

Each time that you cold boot the system, the IBM PC spends a minute or so checking its memory and most of its circuits to insure that everything is working properly. While the checking is going on, a flashing cursor appears in the upper left corner of the screen (called the screen Home

position). The more main memory the machine has, the longer this checkout procedure takes.

Once the checkout has been completed successfully, you will see the red light on the front panel of master Drive **A:** come on and hear a beep from the IBM PC's speaker. At this point, the two hidden DOS program files **IBMBIO.COM** and **IBMDOS.COM** and the program file **COMMAND.COM** (see Section 2.5.2) are being read from the master diskette into the IBM PC's main memory. Once the programs are read in, they are given control of the computer.

The identical DOS booting procedure can be initiated at any time when the computer is already turned on, by entering

<Ctrl-Alt-Del>

In this case, the automatic memory and circuit checking of the cold boot is not performed, and the process is called warm booting. Note: Any information present in the main memory will be lost when the warm boot is initiated.

2.6.3 Entering the Time and Date on a Stand-Alone IBM PC

******* This section applies only to operation of stand-alone (not connected to a network) IBM PCs.

The IBM PC has a very accurate internal 24 hour clock which must normally be set externally when you first power up the computer. The first action taken by DOS after the DOS system program files are loaded into the IBM PC's main memory is to set the internal clock.

For IBM PCs equipped with standard IBM memory boards, the next thing you will see on the screen is a request to enter the date and time. The date should be entered in the form

mm/dd/yy
or **mm-dd-yy**

where **mm** is a one or two digit month, **dd** a one or two digit day and **yy** is a two or four digit year. The time is entered in the form

hh:mm:ss:xx

where **hh** is a one or two digit hour (24 hour clock), **mm** is a one or two digit minute, **ss** is a one or two digit second,

and **xx** is a one or two digit hundredths of a second. **mm**, **ss**, and **xx** can be left off if you wish.

If you want to bypass the entry of date and time completely, simply press the **RETURN** key in response to the requests for date and time, in which case DOS sets the time to midnight on January 1, 1980. We recommend that you not do this, since DOS uses the date and time as part of the file directory information, so that later on you will be able to tell exactly when your files were created or last modified.

In the following illustration, what DOS displays is shown in normal type; what **you** type is shown in **bold face**. Every line you type should be terminated with a **RETURN** or **ENTER**, shown here as <cr>.

```
The current date is Tue  1-01-1980
Enter new date:  9/14/87<cr>
Current time is 0:00:00:00      (the time may differ)
Enter new time: 14:30<cr>
```

Here the date entered is September 14, 1987 and the time is 2:30 PM.

This will be followed by a message similar to the following:

```
The IBM Personal Computer DOS
Version 3.1 (C)Copyright IBM Corp
1981,1982,1983,1984,1985
```

Finally, the characters

```
A:\>
```

will appear in columns 1 and 2 of the next screen line. These are called the DOS prompt characters, or simply the DOS prompt and indicate:

1. DOS is expecting you to enter a DOS command.

2. Disk Drive **A:** is the current DOS default drive and (for a stand-alone IBM PC), contains the diskette where the DOS operating system program files, particularly those for transient DOS commands listed in Section 2.7.2, can be found.

3. Anytime that a file name is used without a drive designator, DOS will first assume that the file is on the diskette in Drive **A:**, e.g.:

   VEDIT is interpreted as A:VEDIT

4. Any reference to a file on Drive **B:** must be preceded by the **B:** drive designator, e.g.,

```
B:MYFILE
B:PROB1.TXT
```

Some memory boards available from independent manufacturers contain a battery powered calendar/clock, that can be accessed directly by DOS. [For example, the memory boards in the IBM PC computers manufactured by the AST Corp., are equipped with such clocks.]

p2.21<=

When DOS is first booted, DOS immediately checks the master diskette to see if it contains a batch file named **AUTOEXEC.BAT**. If the file is present, DOS executes all DOS commands that it finds in the file before doing anything else. Hence, if this batch file contains a DOS command to execute a clock-setting program file (for the AST clock the program file is named **ASTCLOCK.COM**, and is also stored on the master diskette) DOS will read the battery-powered clock and set the computer's internal clock to the correct date and time automatically. In this case, you will not be asked to set the time and date manually as described above. You may see lines similar to the following displayed on the screen (the last line being the DOS prompt indicating that Drive **A:** is the default drive):

p2.21<=

```
A:\>autoexec

A:\>astclock
Current date is 09/14/87
Current time is 14:30:10:44

A:\>
```

2.6.5 Setting the Printer to Top-of-Form

The Epson or IBM dot matrix printer contains a microprocessor that, among other tasks, keeps track of the

paper spacing as each line is printed. In order to use this feature effectively, it is essential that the line counting be initiated starting at the top of a page (called top-of-form). To do this, perform the following steps:

1. Turn the printer on, if it is not already turned on. The POWER, READY, and ON LINE indicators on the printer console should be lit.

2. Press the ON LINE button once. The READY and ON LINE lights should turn off.

3. Lift up the protective cover on the top of the printer.

4. Press the LF (line feed) button repeatedly until the page perforations appear at the paper guide (the metal crossbar with the column numbers embossed on it).

5. Lower the protective cover.

6. Turn the printer switch off.

7. Turn the printer switch on.

When the printer is turned on in step 7 it assumes that the paper is positioned at the first line of a page, and maintains a line count thereafter.

Later, if you print some output that ends in the middle of a page and you wish to start subsequent printing at the top of a new page, use the following procedure to position the paper to top-of-form for the next page:

1. Press the ON LINE button once. The READY and ON LINE lights should turn off.

2. Press the FF (form feed) button once. The paper will advance until the top of the next page is reached.

3. Press the ON LINE button once. The READY and ON LINE lights should come on.

If you wish to eject an additional page so that already-printed output can be removed from the printer, simply repeat step 2 of the process before going on to step 3.

Note: Never use the platen knob to position the paper in the printer when the printer is turned on.

2.7 DOS Commands - Changing the Default Drive

Whenever the DOS prompt (e.g., **A:\>**, **B:\>**, **C:\>**) appears, DOS is waiting for you to enter a DOS command. DOS recognizes only a small number of commands, the most important of which are described in the remaining sections of this chapter. DOS commands can be classified into two major categories, called resident and transient.

2.7.1 Resident Commands

The main PC/DOS programs are loaded into main memory from the master diskette program files **IBMBIO.COM**, **IBMDOS.COM**, and **COMMAND.COM** during the booting process, and remain in the IBM PC memory at all times when DOS is in control. (If a large application program is run, then it may be necessary to allocate temporarily the main memory space occupied by some system programs to the large program and then to reload the system programs later on when the application program has finished execution - DOS handles such situations automatically).

Some DOS commands are completely processed by these memory-resident parts of DOS, and hence are called resident commands (or sometimes internal commands). Such commands can be executed immediately, since no additional programs need be read from disk to process them.

The following resident DOS commands are the ones you are most likely to use as a beginner, and are described in Sections 2.7.3 through 2.20:

```
CLS
COPY
DATE
DIR
ERASE  (or DEL)
RENAME
TIME
TYPE
VER
VERIFY
VOL
```

Other more advanced resident commands (CHDIR, MKDIR, RMDIR and PATH) are covered later in Sections 2.21 and 2.22.

2.7.2 Transient Commands

Certain of the more complicated and/or less frequently used DOS commands cannot be processed by the resident parts of DOS alone. In these cases, on a standalone IBM PC, DOS first searches the diskette in the current DOS default drive (normally Drive **A:**), for a program file with the same name as the command and with the filename extension **.COM**. If DOS finds such a program file, it reads the program into the main memory and executes it.

These DOS commands are called transient commands (or sometimes external commands) since the programs that do the processing are not resident in the memory at all times. The transient DOS commands described in Sections 2.9 through 2.20 of this chapter are:

ATTRIB
CHKDSK
COMP
DISKCOMP
DISKCOPY
FORMAT
GRAPHICS
LABEL

MODE
PRINT
SYS

Other more advanced transient commands, TREE, MORE, SORT, and FIND, are covered in Sections 2.21.1 and 2.23.

When you enter one of these commands, you will probably notice a significant time delay before it is fully processed. The delay is caused primarily by the time required to read the pertinent program file from the disk drive and load it into the fast memory of the IBM PC.

2.7.3 Changing the Default Drive

When the default drive prompt **A:\>** appears at the left side of the screen, DOS expects you to enter a DOS command.

Note: In the remainder of this chapter, most of the examples will assume that you are using a stand-alone IBM PC
for which the default drive is normally **A:**. If you are using a PC-XT or PC-AT (see Section 2.24), the default drive after the normal booting process is complete will be **C:** and the DOS prompt will be **C:\>**; most appearances of the prompt **A:\>** in the examples that follow should be replaced by the prompt **C:\>**.

For starters, enter a very simple DOS command, **VER**, that will cause DOS to display its version number, (3.1 in our case). Simply type **VER** followed by <cr> (we use this as the abbreviation for carriage return, Return, or Enter throughout the text; press the key on the right-hand-side of the keyboard with the bent left-pointing arrow immediately below the backspace key) as in:

A:\>VER<cr>

Remember to enter only the part in **boldface.**

You should see the following on the screen at this point:

```
IBM Personal Computer DOS Version 3.10

A:\>
```

DOS processed the command, displayed its version number, and then displayed the **A:\>** prompt to let you know that it is ready to accept another command from you.

Since DOS does not differentiate between lower- and upper-case letters in its commands, the **VER** command could also be entered as **ver** (try it!):

A:\>ver<cr>

You can change the default drive simply by entering the new default drive name following the currently displayed DOS prompt. For example, to change the default drive from **A:** to **B:**, simply enter (try it!):

A:\>B:<cr>

DOS will next display the prompt character

B:\>

rather than **A:\>**; the default drive is now Drive **B:**.

Changing the default to Drive **B:** has the following affects if the **PATH** command (introduced later in Section 2.21.3) has not been used:

1. Files referenced in DOS commands without a drive designator are assumed to be present on the diskette in Drive **B:**.

2. All DOS system program files, including those for the transient commands listed in Section 2.7.2, are assumed to be present on the diskette in Drive **B:**.

3. References to files on the diskette in Drive **A:**, presumably the master diskette, must be preceded by the file designator **A:**, as in:

   B:\>A:VEDIT<cr>

4. If you wish to execute any DOS transient commands when the associated program files are present on the master diskette in Drive **A:**, the command (actually the name of the program file for the command) must be preceded by the file designator **A:**. For example:

   B:\>A:FORMAT<cr>

This need to remember where important program files are located with respect to the current default drive assignment is rather bothersome. In addition, if the default drive is changed from **A:** to **B:**, then Drive (**A:**) (we call it the master drive), containing the master diskette, and the default drive, (**B:**), will be different. Because of the confusion this may cause for beginners, we recommend that you not change the default drive for now.

If you have changed the default drive from **A:** to **B:** as suggested above, change it back to Drive **A:** by entering **A:**, as in:

B:\>A:<cr>

The DOS prompt appearing on the screen should be restored to:

A:\>

We should note that DOS has an alternative solution to this problem (whether or not a drive designator must appear as part of a referenced file name), but that involves use of one of the more advanced commands, **PATH**, described later on in Section 2.21.3. For the moment, we will ignore the availability of that command.

2.8 Displaying a Diskette Directory (DIR Command)

One of the most used DOS commands is the resident command **DIR** (**DIR**ectory). To get a listing of the file directory for the master diskette in default Drive **A:** on the screen, simply enter the command immediately following the DOS prompt (try it!):

A:\>DIR<cr>

The screen display will contain a fairly long list of the names of the files on the master diskette and include those shown in Section 2.5.2). The information for each file is written on a separate line in the order: **file name**, **file name extension** (without the period), **bytes of disk storage** occupied, **date of last change**, and **time of last change**. You should see entries similar to the following:

```
 Volume in drive A has no label
 Directory of A:\

COMMAND  COM     23210   3-07-85   1:43p
VEDIT    COM     40961   6-01-87  11:27a
  .       .       .       .       .
  .       .       .       .       .
```

The entry for VEDIT.COM indicates that the program file occupies 40961 bytes of diskette storage, and was last changed at 11:27 AM on June 1, 1987.

The general form of the **DIR** command is (ignoring the prompt and carriage return):

DIR [d:][filename.ext][/P][/W]

Here and in the remainder of this chapter, the square brackets indicate that the entry is optional, i.e., the entry can either be included or not. **d:** is the disk drive designator (e.g. **A:**, **B:**, **C:**, etc.), **filename** is a file name or a generic file name, and **.ext** is a file name extension. Since all entries are optional, the simplest form of the command is the one you have already entered (**DIR**), in which case the complete directory of the default drive will be listed.

If you want to see the directory information for a particular file on the master drive (say **VEDIT.COM**), the command is (try it):

A:\>DIR VEDIT.COM<cr>

or **A:\>DIR A:VEDIT.COM<cr>**

Note that if the file is on the default drive, then the drive designator need not be attached to the file name.

If you want to see the directory information for all files on the master diskette whose names start with the letter **V** and have the file name extension **.COM**, the command is (try it!):

A:\>DIR V*.COM<cr>

Note that the generic file name **V*.COM** is being used in this case.

If the characters **/W** appear, as in

A:\>DIR /W<cr>

then the directory listing includes the file names only, written in horizontal (Wide) format, five file names per line (try it!).

If the **/P** parameter appears, as in

A:\>DIR /P<cr>

then DOS will stop listing the directory when the screen is full so that you can examine it. The **/P** parameter is useful when the directory is too long to be displayed on a single screen image (25 lines). To continue the listing process, just press any key in the typewriter section of the keyboard.

To get a complete directory listing for the diskette in Drive **B:**, the command is (try it!):

A:\>DIR B:<cr>

If this is the first time you have used a newly purchased diskette, you should see the following information displayed on the screen:

```
Disk error reading drive B
Abort, Retry, Ignore? A
```

As indicated, you should enter **A** or **a** (for Abort), in this case not followed by <cr>.

What caused the error? The problem is that the new Drive **B:** diskette has no directory until it has been formatted using the DOS command **FORMAT**, as described in the next section.

2.9 Formatting Your Personal Diskette (FORMAT, SYS, LABEL and VOL Commands):

The IBM/PC requires that files be stored on diskettes in an orderly fashion called the disk format. In PC/DOS 3.1, the surface of a double-density diskette is divided into 40 annular tracks that are concentric from inside to outside; each track is divided into 9 sectors containing 512 bytes each, for a total of 184320 (180 K) bytes for a single-sided diskette (not much used) and 368640 (360 K) bytes for a double-sided diskette (used on all of the IBM PCs, PC-XTs and PC-ATs).

Double-sided quad-density diskettes can be formatted on Drive A: of the PC-AT; however, such diskettes cannot be used on a PC or XT. Therefore, if you plan to use a diskette on the PCs and on the XTs and ATs , it must be prepared in the double-sided double-density format.

The format is written in magnetic form on the surface of the disk. DOS uses the disk directory and the associated allocation table to keep track of which files are stored on the various sectors of the diskette surface.

Because of the proliferation of personal computers on the market, there are many types of formats used for storing and retrieving data from diskettes. Hence, diskettes from another kind of computer (unless it is "IBM compatible") probably cannot be read by the IBM/PC floppy disk drives.

When you buy a new diskette, it has no format whatsoever. Before it can be used on any computer, the appropriate format for that computer must be imposed upon it. This format can be removed by reformatting the diskette on a disk drive on the same or on another kind of computer. However, when a diskette is reformatted, any information on it is erased (hence will be lost) during the formatting process.

In DOS, the disk formatting command has the form

FORMAT d:[/S][/1][/8][/4][/V][/B]

where, as in the previous section, the square brackets indicate optional entries, and **d:** is the drive designator. Note that **d:** appears without brackets, so that the appropriate drive designator must appear in the command.

The **/S** entry will cause DOS to copy the system files **IBMBIO.COM**, **IBMDOS.COM**, and **COMMAND.COM** from the diskette in the default drive (normally these system files are on Drive **A:** of an IBM PC) to the diskette in Drive **d:** (usually Drive **B:**) after the diskette has been formatted. The entry /1 indicates that the diskette is to be formatted for single-sided disk drives (usually found only on very old IBM PCs). If the drive is itself single-sided, then the /1 entry is superfluous, since the drive can only format single-sided diskettes anyway.

The entry **/8** indicates that the diskette should be formatted with 8 rather than 9 sectors per track; this format allows the diskette to be processed using PC/DOS 1.1 as well as later versions of DOS, but reduces the amount of information that can be stored from 180 K bytes per diskette surface to 160 K bytes. Since DOS 1.1 has been supplanted by later versions, this entry is normally not used.

The entry **/4** indicates that the diskette should be prepared in double-density double-sided format on a quad-density drive; it can be used only when formatting on the high-capacity floppy drive **A:** of a PC-AT, and should be used with caution (such a diskette may not be read or written reliably on a single or standard double-sided drive).

The **/V** entry indicates that you wish to specify a volume label or name for the diskette. After the diskette is formatted, DOS will request that you enter a label of up to 11 characters (we suggest that you use letters and digits only). This label will be incorporated into the directory information for the diskette and displayed whenever the **DIR** command is issued.

The entry **/B** is not used very often. It causes the diskette to be prepared in double-density eight-sector format, with entries for the hidden DOS files **IBMBIO.COM** and **IBMDOS.COM** placed in the directory. However, the hidden files themselves are not written on the diskette (as would happen with the **/S** entry, for example). The purpose of this entry is to allow for the preparation of diskettes for application programs that are designed to use DOS, but that cannot be distributed with the DOS system programs on them (since the DOS software is copyrighted by IBM). If the **/B** option is used in preparing the diskette, then the customer can later on copy his own DOS system files (presumably he

has paid for them!) to the diskette using the DOS transient command **SYS** for which the prototype is:

SYS d:

where **d:** is the drive containing the diskette to which the system files are to be copied. This is the only way one can copy the hidden system files **IBMBIO.COM** and **IBMDOS.COM** to an already formatted diskette.

Normally a diskette is formatted only once, the very first time it is used on a particular model computer.

Note: If you have already formatted your personal diskette, skip the rest of this section and resume reading at Section 2.10.

Assuming that your personal diskette in Drive **B:** has not yet been formatted, enter the DOS command **FORMAT** as follows:

A:\>FORMAT B: /V<cr>

DOS should next respond with the message:

```
Insert new diskette for drive b:
and strike ENTER when ready
```

Since your blank diskette is already in Drive **B:** if you have followed the instructions in this chapter to this point, strike any (typewriter) key. The formatting operation takes a minute or so to complete. The command above will cause your personal diskette in Drive **B:** to be formatted as a double-sided diskette.

When DOS is finished with the formatting operation, it will display the following message requesting a volume name for your diskette:

```
Formatting ...
Volume label (11 characters, ENTER for none)? name<cr>
```

You should enter an appropriate label, for example, your **name.** Once the label has been entered, DOS will display the message

```
Format complete.

Format another (Y/N)?N<cr>
```

Respond with **N** (No) followed by <cr>. Your personal diskette is now formatted and available for use.

Note: Twelve sectors of the diskette surface containing 6144 bytes are reserved by DOS for the file directory, storage allocation table, and volume name, leaving 368640 - 6144 = 362496 for file storage.

To insure that the diskette is indeed formatted, reenter the **DIR**ectory command

A:\>DIR B:<cr>

and note that the earlier error message of Section 2.8 does not appear. If you wish to determine the label of the volume without displaying a directory, you can enter the resident DOS command **VOL** which has the form:

VOL [d:]

Now enter the command:

A:\>VOL B:<cr>

DOS should display a message similar to:

Volume name J O Wilkes created September 14, 1987 15:05

It is possible to change or delete a volume label or to create a label for a previously unlabeled diskette using the transient command **LABEL** for which the command prototype is:

LABEL [d:][volname]

Here, **d:** is the drive designator and **volname** is the desired new volume name. If the optional parameters are left off, then the default drive is assumed, and DOS will prompt you to enter the new name (type <cr> without a name to delete an existing volume name).

The DOS commands **FORMAT** and **LABEL** are transient commands. When you enter the transient **FORMAT** command, you will notice that the **FORMAT.COM** file must be read from the master diskette in Drive **A:** (the read/write indicator on the front of Drive **A:** will light up) before the formatting operations can begin (see Section 2.5.2).

2.10 Determining Diskette Storage Status (CHKDSK Command)

It is often useful to know how much space is occupied by files on a diskette and how much free space is available for storing more files. The **CHKDSK** (**CH**ec**K** **D**i**SK**) command allows you to do this. The general form of the command is:

CHKDSK [d:][filename][/F][/V]

CHKDSK is a transient command, and produces a table on the screen that is similar to the following for a double-sided diskette:

```
362496 bytes total disk space
 38912 bytes in 2 hidden files
 23552 bytes in 1 user file
  4096 in bad sectors
295936 bytes available on disk

458752 bytes total memory
229744 bytes free
```

If the entry **/V** appears in the command, then a list of all files on the diskette will be displayed as well.

Most of the entries are obvious, but a few need explanation. The hidden files referred to are the DOS system files from the master diskette (**IBMBIO.COM** and **IBMDOS.COM** described in Section 2.5.2). Bad sectors are sectors that were discovered to have a flaw (perhaps a surface scratch) during the formatting process. The **FORMAT** command program blocks these sectors from further use by DOS, but still allows the good parts of the surface to be used later for storing files. The last two entries refer to the main memory of the IBM/PC computer rather than to the diskette in Drive **d:**; the main RAM memory has 448 K bytes of RAM in this example.

Now enter the two commands

A:\>CHKDSK<cr>
and **A:\>CHKDSK B:<cr>**

These commands will give you storage status information about the master diskette and your personal diskette, respectively.

If the entry **filename** appears in the command, then DOS will tell you how many non-contiguous sectors are occupied by the file named **filename** on the diskette; this is a measure of the fragmentation of the file (the more fragmented a file is, the longer it will take DOS to read

the full file from diskette). This is an option that beginners need not be concerned with.

Occasionally, errors in the diskette directory or file allocation table are generated (usually by badly behaved application software), leading to what DOS calls lost clusters. Lost clusters are track sectors for which pertinent allocation information has been destroyed or incorrectly altered. If the /F optional entry is included with the **CHKDSK** command, information in the orphaned track sectors is collected and written into new files named FILE0000.CHK, FILE0001.CHK, etc. These files can be examined to see if they contain useful information, but usually are of little value (if so, they should be erased to recover diskette storage space).

2.11 Executing Program and Batch Files

As described in Section 2.2, a DOS program file contains a binary object program (a program in machine language form) that needs no further processing before being loaded into the main memory of the IBM/PC by DOS. The loaded program can then be executed, meaning that DOS temporarily turns over control of the computer (the microprocessor) to it. When program execution is complete or when an error occurs during program execution, control returns to DOS, which is then ready to accept another command (for example, to load and execute another program file).

Batch files are special text files containing DOS commands. DOS processes the commands from sequential lines in the file as if they had been entered at the keyboard in the same order. A batch file is handy when the same command sequence is needed many times, since its use precludes the necessity of entering the commands one at a time, over and over again.

In addition, it is often useful to have DOS carry out some commands automatically when DOS is first booted. DOS will do this if the commands are present on the diskette in Drive **A:** (on the hard disk, Drive C:, for the PC-XT or PC-AT) in an autoexec batch file with the special name **AUTOEXEC.BAT**. There can, of course, be only one autoexec batch file on the master diskette, since DOS requires that every file stored on a diskette have a unique name.

2.11.1 Program Files

To execute the program in a DOS program file named **progfile** with the file name extension **.ext**, the DOS command, using the notation introduced earlier, is:

[d:]progfile[.ext] [pars]

Here, as before, the square brackets indicate an optional entry, so that, for example, the drive designator **d:** need not appear. If **d:** is omitted, DOS assumes that the file is on the default drive indicated by the DOS prompt, e.g., **A:** (or **C:** on the PC-XT or PC-AT). If the file cannot be found on the default drive, DOS will print a message to that effect.

Note: If a directory search path (described later in Section 2.21.3) has been assigned, then directories in the directory search path will also be searched to see if the missing file appears there.

Some, but not all, program file execution commands may have additional optional parameters, indicated as **pars** in the prototype above.

In general, the filename must appear in its full form, including the file name extension, **.ext**. However, the file name extension need not appear if it is one of the following:

.COM DOS interprets the file as one containing an object program that is to be loaded into a specified block of the main memory normally used for DOS transient COMmand program files.

.EXE DOS interprets the file as one containing an object program that can be loaded into any available block of main memory space; it is called a relocatable EXEcutable file.

.BAT DOS interprets the file as a BATch file, and processes all of the lines in the file as DOS commands.

Note that you have already loaded and executed program files named **FORMAT.COM** and **CHKDSK.COM** commands:

```
A:\>FORMAT B: /V<cr>
A:\>CHKDSK B:<cr>
```

These transient command program files are normally present on drive **A:** of an IBM PC.

Note that because the filename extension for the **FORMAT** and **CHKDSK** program files is **.COM**, it need not be specified as part of the file name in the DOS commands.

At this point, you do not yet have any program files (or text files either!) on your personal diskette. Nevertheless, if you did have a program file named **MYPROG** on your personal diskette in Drive **B:**, the appropriate DOS command to load and execute it would be:

A:\>B:MYPROG <cr>

Note that the drive designator **B:** would have to appear in this case, since Drive **B** is not the default drive. The filename extension is one of the three exceptions to the rule: **.COM**, **.EXE**, and **.BAT**.

The original text was heavily modified here

2.11.2 Batch Files

It is not likely that you will make much use of batch files as a beginner. You may wish to skip the remainder of this section, which is included for the sake of completeness.

If you check the directory of the master diskette, you will find an entry for a file named **AUTOEXEC.BAT**, that was mentioned in Section 2.6.4. This file is a text file that DOS will treat as a batch file (because the file name extension is **.BAT**). It contains several lines that

implement the the initial setup described in Section 2.6.4 including:

```
Astclock
PROMPT $P $G
LOGIN
D:INIT
```

When DOS is first booted, it checks the file directory of the diskette in the default drive (initially **A:** for the IBM PC and usually C: for the PC-XT or PC-AT) for a file named **AUTOEXEC.BAT**. If it finds the file, DOS interprets each line as a DOS command. For the autoexec batch file on the master diskette, DOS processes the first of the commands listed above as if you had entered

A:\>Astclock<cr>

at the keyboard. If you recheck the directory (see Section 2.8) for the master diskette, you will find a program file named **ASTCLOCK.COM**. Note that the file name extension **.COM** need not appear in the DOS command in the batch file, and that DOS does not distinguish been upper and lower case letters in file names.

DOS loads and executes the **ASTCLOCK.COM** program as described in the previous subsection, and reads the date and time from the battery-powered AST calendar/clock. The second line listed causes DOS to load and execute the DOS program file named **PROMPT.COM**, that sets the appearance of the DOS prompt on the screen.

After completely processing the batch file commands, DOS displays the **A>** prompt to let you know that it is ready to accept DOS commands from the keyboard.

If you wish, you can create batch files of your own. You simply create the appropriate text file with the file name extension **.BAT** and enter the DOS commands as lines in the file.

Since we haven't described how to create and enter lines into text files yet, you won't be able to create your own

batch files at this point. However, once you have a batch file of your own (named **ORDERS.BAT**, for example) on your personal diskette, you can have DOS process the DOS commands in it by entering (the file name extension **.BAT** is not required in the command):

A:\>B:ORDERS<cr>

2.11.3 Stopping Program Execution

You can usually interrupt the processing of a batch file or DOS command or execution of a loaded program file by entering

<Ctrl-Break>

Press the **<Ctrl>** and **<Break>** (also marked **Scroll Lock**) keys simultaneously. DOS will stop any current activity and then display the DOS prompt, indicating that it is ready to accept another command.

It is possible for programs to disable this interrupt feature, however, so this key combination will not always cause DOS to interrupt processing. If this is the case (particularly in commercial software), there will almost always be an alternative way of stopping the program via a local command, special key, or menu selection.

If all else fails, and you want to stop whatever the computer is doing at the time, you can enter the three key combination

<Ctrl-Alt-Del>

This will result in a warm boot of DOS (see Section 2.6.2), and you can start all over again. If processing is stopped in this way, DOS will not retain any of the information that was in the main store of the computer at the time of the interrupt; all information in the store will be unrecoverable (including any files that are stored in a RAM disk). Files on floppy diskettes or hard disk will usually be unaffected, although there is a risk that files that were "open" for writing (by a compiled FORTRAN program, for example) may be missing some of the last output written to them, since they will not have been properly "closed" by DOS before the unanticipated interrupt.

If you are truly desperate to stop the computer while a program is executing, you can, of course, always turn off

the power switch! This will not cause any damage to the computer, but, again, information may be lost. It is always acceptable to turn off the computer when the DOS prompt is showing, since no (application) program is in execution at that time; any files in a RAM disk (if present) will still be lost, of course, since the RAM memory is volatile.

2.12 Entering Lines into a Text File (COPY Command)

Now that your personal diskette in Drive **B:** is formatted, and you have some acquaintance with the most important features of PC/DOS files, devices, and commands, and the loading and execution of program files, you are ready to begin using most of the DOS commands. You will find that the commands are straightforward, and that you can become proficient in using DOS very quickly. There are only 31 commands to learn, including the **VER**, **DIR**, **FORMAT**, **VOL**, **LABEL**, **SYS**, and **CHKDSK** commands already covered in Sections 2.7.3 through 2.10.

The remaining sections of this chapter describe briefly the rest of the DOS commands in the order **COPY**, **TYPE**, **PRINT**, **TEXT**, **MODE**, **CLS**, **GRAPHICS**, **VERIFY**, **ATTRIB**, **COMP**, **RENAME**, **ERASE**, **DISKCOPY**, **DISKCOMP**, **DATE**, **TIME**, **TREE**, **CHDIR**, **PATH**, **MKDIR**, **RMDIR**, **MORE**, **SORT**, and **FIND**. Most of these commands involve the manipulation of files and devices. As a beginner, you are likely to make regular use of only a few of them, in particular, **COPY**, **TYPE**, **PRINT**, and **ERASE**, so focus your attention on learning to use them first.

The only way to create a new text file named **textfile** and to enter lines of characters into it directly from the keyboard using **only DOS commands** is with the resident command **COPY**, for which the prototype is:

COPY CON: [d:]textfile

Here, DOS interprets the command as **COPY from the keyboard (CONsole) to the text file textfile**. As a result of entering a command such as

A:\>COPY CON: B:MYFILE.TXT<cr>

DOS does the following:

1. It checks the file directory of the diskette in Drive **B:** to see if the file **MYFILE.TXT** already exists.

2. If **MYFILE.TXT** already exists, DOS erases the file contents (removes any lines currently in the file), but leaves the file name in the file directory. If **MYFILE.TXT** does not already exist, DOS creates an entry for the file name in the directory and assigns the file some storage space on the diskette.

3. Sets internal controls so that any lines entered at the keyboard are interpreted as lines destined for the file **MYFILE.TXT**

4. Accepts characters one at a time from the keyboard until you key in a carriage return **<cr>** to indicate the end of a line. Before you enter **<cr>**, you can use the editing keys, such as **<Backspace>**, **<Ins>**, and **<Del>**, to correct any errors in the line. The line itself is displayed on the monitor screen as you type and edit it.

5. When **<cr>** is keyed in, DOS assumes that the line is complete. DOS then copies the line of characters into the next line of the file **MYFILE.TXT** on the diskette in Drive **B:**.

6. The process of accepting lines from the keyboard and copying them in sequence into the target file continues until you enter the keystrokes:

   <F6><cr>

   These characters should be entered separately after the **<cr>** for the last file line has been keyed in.

Now, try entering the following on the keyboard:

```
A:\>COPY CON: B:OWNER.TXT<cr>
The Owner of this diskette is:<cr>
Name:  first last<cr>
Address:  number and street<cr>
          city and state<cr>
          zip code<cr>
Phone:    phone number<cr>
<F6><cr>
```

The **COPY** command and the entry of the appropriate lines (put your **first** and **last** name in the **Name:** line, etc.) causes DOS to create a file named **OWNER.TXT** on your personal diskette in Drive **B:** and to enter your personal information into the first six lines of the file. Note that DOS gives no indication that it is waiting for characters to be entered. Simply start typing the first line for the file after entering the **COPY** command. If you have any problems while entering the information, key in **<cr>** followed by

<F6><cr> and start over again with the **COPY** command when the DOS prompt **A:\>** appears.

After you enter **<F6><cr>**, you will see the characters **^Z** displayed on the console; this comes about because the **<F6>** key has been programmed to produce the same internal code as the DOS "end-of-file" key combination **<Ctrl-Z>**, displayed as **^Z** by DOS.

Now, enter the command

A:\>DIR B:<cr>

The directory should contain the file named **OWNER.TXT** and there should be 150 - 200 bytes of storage occupied by it, one byte for each character entered, plus a few special control characters.

At this point, it is important to note that this way of creating and entering lines into a file is practical only for extremely short files, since there is no direct way of correcting mistakes once the lines have been written into the file using DOS commands only.

2.13 Displaying and Printing a Text File (TYPE, COPY, PRINT Commands)

The prototype DOS command for displaying a text file **textfile** on the monitor screen is:

TYPE [d:]textfile

The name given this DOS resident command (**TYPE**) is perhaps not descriptive of what the command does for you, since there is no "typing" involved (**DISPLAY** would seem to be a more appropriate command name). Regardless of the name, the **TYPE** command simply causes DOS to read the lines from the named text file, and then send them to the screen for display.

For example, to see the lines of the text file **OWNER.TXT** that you created on your personal diskette in Section 2.12, enter:

A:\>TYPE B:OWNER.TXT<cr>

If the file were longer than 25 lines (the capacity of one screen image), not the case here, then the **TYPE** command would write the lines one after another on the screen, scrolling them by too rapidly to read. To stop this scrolling action, simply enter the key combination:

<Ctrl-S>

or **<Ctrl-NumLock>**

You can then read the lines at will. When you want to have DOS display the next lines in the file, strike any key in the typewriter section of the keyboard, and scrolling will resume.

If you would like to have the lines of the text file printed on the attached matrix printer, in addition to being displayed on the screen, press

<Ctrl-PrtSc>

before you enter the **TYPE** command. The printer must be turned on and the ON LINE indicator must be lit when you do this. Now, press **<Ctrl-PrtSc>**, and then reenter the last **TYPE** command again.

After the **TYPE** command has finished displaying the file, enter **<Ctrl-PrtSc>** again to toggle the screen printing switch to off; otherwise, lines written on the screen later on will also be printed.

An alternative way to achieve the display of a text file on the screen is to use the **COPY** command (introduced in Section 2.12) in different prototype form:

COPY [d:]textfile CON:

In this case, the lines in the source file **textfile** on Drive **d:** are copied to the output device **CON:**, i.e., to the monitor screen. For example, a command completely equivalent to the **TYPE** command already given is (try it):

A:\>COPY B:OWNER.TXT CON:<cr>

The key combination **<Ctrl-S>** or **<Ctrl-NumLock>** can be used to stop the copying process temporarily; copying resumes when you type any key in the typewriter section of the keyboard.

A slight variation of this form of the **COPY** command can be used to get a printed copy of the file **textfile:**

COPY [d:]textfile LPT1:

Here, the lines in **textfile** are copied to the line printer, **LPT1:**. You will usually want to formfeed the paper in the printer before making the listing (see Section 2.6.5), so that printing will begin at the top of a new page. The printer must be turned on and the ON LINE indicator on the printer console must be lit. The appropriate command to print the contents of the file **OWNER.TXT** is (try it):

A:\>COPY B:OWNER.TXT LPT1:<cr>

One of the more powerful DOS features is the ability to spool or queue the printing of text files with the **PRINT** command. Files are printed in the order they are entered into the queue, and printing is done in a background mode, so that once you initiate the printing of one or more files, you need not wait until printing is finished before continuing with other computing tasks.

The prototype for this transient command is:

PRINT [d:][textfile][/T][/C][/P]

Here, the one or more files **textfile** on Drive **d:** are entered into the print queue. If **/T** appears, the printing of all already-queued files is terminated; the **/C** entry causes printing of any listed files (up to ten files may be assigned to the queue) to be cancelled. The entry **/P** initiates printing, and is assumed to be present by default if no other / entries appear.

For example, if you wanted to print two files on your personal diskette named **FILE1** and **FILE2**, while continuing to do something else on the computer, the command would be:

A:\>PRINT B:FILE1 B:FILE2<cr>

The first time you use the **PRINT** command, DOS will ask you which output device you would like to "print" on. The printer **LPT1:** (or **PRN:**) is the default device, and will almost always be the one you will want to use, so simply enter **<cr>**; however, any of the standard output devices, such as a disk drive can be specified if you wish.

2.14 Screen and Printer Modes (MODE, CLS, and GRAPHICS Commands)

The default settings for the monitor screen and printer formats for the IBM PCs in the Freshman Engineering Computing Laboratories are:

Monitor Screen	25 lines of 80 columns each
Printer	132 columns per line 6 lines per inch

These settings are the ones you will usually want to use, but DOS allows you to choose optional formats as follows:

Monitor Screen	25 lines of 40 columns each
Printer	80 or 132 column lines 6 or 8 lines per inch

To reset the parameters, use the transient DOS command **MODE** for which the prototype is:

2.56 Screen and Printer Modes (MODE/CLS/GRAPHICS Commands)

MODE [LPT#:][n][,[,m]]

Here, if **LPT#:** appears, the printer format for printer # (1, 2, or 3) is reset. **n** is then the number of columns per line (either **80** or **132**) and **m** is the number of lines per inch (either **6** or **8**). If **m** is omitted, the default value **6** is assigned. If both **n** and **m** are omitted, DOS assigns the values **132** and **6** for the printers in the FEC Laboratories and the values **80** and **6** for the printers in the CAEN Laboratories.

Examples: **A:\>MODE LPT1:132,8<cr>**
A:\>MODE LPT1:80<cr>

If **LPT1:** does not appear, then the screen format is reset. In this case, **n** is the number of columns per line and is either **40** or **80**; **m** should not appear at all.

Example: **A:\>MODE 40<cr>**

If the IBM/PC is equipped with one of the color/graphics adapters, then the attached monitor (either monocolor or color) can produce screen displays in either text mode (25 lines of either 40 or 80 columns) or graphics mode (pictures are constructed from an array of dots or pixels, as described in Section 1.2.3). All IBM/PCs in the College of Engineering laboratories are equipped with color/graphics adapters (even the PCs in the Engineering Freshman Computing Laboratories, which have monocolor green-screen monitors).

If the monitor supports color (the monitors on the XTs and ATs in the CAEN Laboratories, for example) then **n** in the **MODE** command can be any of the following: **BW40**, **BW80**, **CO40**, **CO80**. **BW** stands for black and white (i.e., color is disabled), **CO** stands for color, and **40** and **80** is the number of columns per line in text mode.

Example: **A:\>MODE BW40<cr>**

This command would set the color monitor to black and white display only, with 40 columns to the screen text line.

If the PC is equipped with both a monochrome and color adapter and with two monitors, an IBM monochrome alphanumeric and a color monitor, then **n** in the **MODE** command may be **MONO**, in which case the active display adapter is switched from the color to monochrome monitor.

As described in Section 2.4.4, the key combination **<Shift-PrtSc>** causes DOS to make a copy of the current screen image on the attached printer LPT1:. In general, only text mode screens will be copied. However, if the IBM/PC is equipped with a dot matrix printer, then the pixel pattern in a graphics mode screen can also be copied,

provided that the transient DOS command **GRAPHICS** has been executed previously.

Hence, if you expect to make printed copies of screen images containing graphs, pictures, etc., you should execute the **GRAPHICS** command just once as soon as DOS is booted (it is not necessary to execute the command again during the computing session):

A:\>GRAPHICS<cr>

The DOS command to clear the screen is **CLS**. To clear the screen, enter the resident command (try it!):

A:\>CLS<cr>

2.15 Copying Files (COPY, VERIFY and ATTRIB Commands)

You may want to copy one of your files from your personal diskette to another (backup) diskette or perhaps to make an identical copy of the file on the same diskette (to preserve the original file while you make changes in a copy, for example). The DOS **COPY** command, already introduced in several previous sections, is used for this purpose. The prototype of this resident command is:

COPY [/B][d1:]sorcfile [d2:][trgtfile][/V]

Here, the contents of the source file **sorcfile** on drive **d1:** are copied into the target file **trgtfile** on drive **d2:**. **sorcfile** need not be a text file, i.e., program files can also be copied.

[Note: **Only non-copyrighted program files may be copied legally**. These would include, for example, object program files resulting from compilations of your own FORTRAN programs. It is illegal to copy copyrighted software, even if it is technically possible to do so. Many vendors copy protect their program files, so that the programs cannot be duplicated using the DOS **COPY** command.]

Some comments are in order concerning the prototype form of the **COPY** command shown above:

1. If **trgtfile** is omitted, Drive **d1:** must be different from Drive **d2:**, and **trgtfile** will be given the same file name as **sorcfile.**

2. If drives **d1:** and **d2:** are the same, the file names for **sorcfile** and **trgtfile** must be different.

3. If drives **d1:** and/or **d2:** are omitted, the source and/or target drives are assigned the default drive designator.

4. **sorcfile** may be a generic file name (in which case more than one file may be copied).

5. If you are copying a file from your personal diskette in Drive **B:** to another diskette, you should first open the **A:** drive door and remove the master diskette. Then insert the second diskette (it must have been formatted previously) into Drive **A:**. After the copy operation is finished, remove the diskette from Drive **A:** and reinsert the master diskette. Do not shut off the computer while changing diskettes.

   Examples:
   ```
   A:\>COPY B:MYFILE.TXT B:SAVEFILE<cr>
   A:\>COPY B:*.FOR A:<cr>
   A:\>COPY A:INFO B:<cr>
   A:\>COPY B:*.* A:<cr>
   ```

   The last example will cause DOS to copy all files whose names appear in the file directory of the Drive **B:** diskette into files with the same names on the diskette in Drive **A:**.

6. Files can be combined by use of the concatenation operator (+). For example, to copy the complete contents of files F1, F2 and MAIN.FOR on drive **B:** in the order MAIN.FOR, F1, F2 into a file named PROG.FOR on drive **A:**, the command would be:

   ```
   A:\>COPY B:MAIN.FOR+B:F1+B:F2 A:PROG.FOR<cr>
   ```

   All end-of-file designators are removed except for the one associated with the last file; i.e., the target file contains just one end-of-file mark.

   If the source file is concatenated and no target file appears, then the added files are appended, in order, to the end of the first file in the list. For example, the command:

   ```
   A:\>COPY B:MAIN.FOR+B:F1+B:F2<cr>
   ```

   would cause files F1 and F2 from Drive **B:** to be added in sequence at the end of file PROG.FOR on Drive **B:**. All end-of-file indicators are removed except the one for the last-appended file.

7. If a generic source file name appears with a non-generic target file, then all of the appropriate source files will automatically be concatenated and written into the target file. For example, the command

   A:\>COPY B:*.FOR B:COMBINED.FOR<cr>

   will cause DOS to concatenate all files on drive B: with extension .FOR (in the order they appear in the file directory) and then copy the concatenated file to the single file COMBINED.FOR on Drive B:.

8. Concatenation is normally performed on text files. However, binary (usually program) files can also be concatenated. To insure that the copy operation is performed properly for such files, the /B option should appear on the **COPY** command, as in:

   A:\>COPY/B B:F1.BIN+B:F2.BIN B:F.BIN<cr>

9. The /V option on the copy command causes DOS to verify (by rereading and comparing) that all sectors copied have been copied correctly. Writing errors occur infrequently, but this option gives an added measure of assurance that the copy operation has been performed correctly; it does, however, considerably slow down the completion of a file copy operation, and is not needed under most circumstances.

   An alternative to appending the /V option to every **COPY** command is the DOS resident command **VERIFY**, entered simply as

   VERIFY ON
   or **VERIFY OFF**

   for turning automatic verification on and off, respectively. Once verification is turned on, DOS will process all subsequent COPY commands as if the /V option appeared.

Now, make a copy of the file **OWNER.TXT** on your personal diskette (use the name **NEWOWNER.TXT** for the new file) by entering the command:

A:\>COPY B:OWNER.TXT B:NEWOWNER.TXT<cr>

Check the file directory to insure that the file **NEWOWNER.TXT** is present on your personal diskette:

A:\>DIR B:<cr>

Display lines from the new file on the screen with the command:

A:\>TYPE B:NEWOWNER.TXT<cr>

As indicated in Section 1.2.2, a diskette can be write protected by attaching a tab to the write-protect notch on the floppy diskette sleeve. This protects all existing files on the diskette from modification or erasure, but also means that no new files can be written on the diskette. In many cases, it is useful to insure against accidental modification or erasure of individual files on a diskette which is otherwise not write protected. A new DOS command available only in DOS 3.0 and 3.1 allows you to change the read attribute of individual files with the transient **ATTRIB** command which has the form:

ATTRIB [+|- R][d:]filename

Here (| indicates "one or the other"), the optional entry **+R** makes the file named **filename** read-only, while **-R** restores the read attribute to read/write status. If the entry is left off, then the current read status of the file **filename** is displayed. For example, to make all files with name extension .FOR on drive **B:** read only, the command would be:

A:\>ATTRIB +R B:*.FOR<cr>

2.16 Comparing Files (COMP Command)

Occasionally it is useful to compare two different files for equality, for example, to check a backup file against an original one. The DOS transient command for making this comparison is:

COMP [d1:][filenam1] [d2:][filenam2]

Here, **filenam1** and **filenam2** are the two files to be compared on a byte-by-byte basis. If either (or both) filename(s) is (are) unspecified, DOS will prompt you for for it (them). If a drive designator is not specified, then the default drive is assumed.

After the command is entered, DOS will respond with the message:

```
Insert diskette(s) with files to compare
and strike any key when ready
```

If the two files to be compared are on a diskette(s) that is (are) already present in the indicated drive(s), simply strike a key in the typewriter section of the keyboard. DOS will then begin the file comparison and note (on the monitor screen) any unequal bytes found in the two files (after ten such mismatches, DOS will discontinue the comparison operations), or indicate that the two files are identical with the message:

```
Files compare ok

Compare more files (Y/N)?
```

Respond with **N** (no) followed by **<cr>**.

If one or both of the files are on diskettes not currently in the specified drives, simply open the drive door(s), remove any diskette(s) currently in the drive(s), insert the diskette(s) containing the files to be compared, close the drive door(s), and then strike any key in the typewriter section of the keyboard. After the comparison, the diskettes can be removed or replaced as desired, in the usual way.

Note that because **COMP** is a transient DOS command, it normally must be issued before removing the master diskette from Drive **A:**; in the FEC Laboratories, however, the transient DOS command files are stored on the Ethernet fileserver, so this caveat doesn't apply. Do not shut off the computer while changing diskettes.

Try using this command to compare the two (presumably identical) files **OWNER.TXT** and **NEWOWNER.TXT** on your personal diskette by entering:

```
A:\>COMP B:OWNER.TXT B:NEWOWNER.TXT<cr>
```

2.17 Renaming Files (RENAME Command)

DOS allows you to rename your files with the resident command

```
RENAME [d:]oldname newname
```

where **oldname** is the current file name and **newname** is the new name you would like the file to have. For example, to change the name of the file on your personal diskette currently named **NEWOWNER.TXT** to **OWNER.SAV**, the DOS command is (try it):

A:\>RENAME B:NEWOWNER.TXT OWNER.SAV<cr>

DOS simply changes the name of the file in the file directory for the diskette, without modifying the contents of the file in any way.

Check the directory of your diskette with the **DIR** command to insure that DOS has changed the name.

A:\>DIR B:<cr>

2.18 Erasing Files (ERASE or DEL Command)

If you no longer want to keep a file named **filename** stored on your diskette, you can tell DOS to remove the file's name from the diskette file directory and free up the space occupied by the file on the diskette. The command prototype for the resident DOS command **ERASE** is:

ERASE [d:]filename

Now, enter the **CHKDSK** command to determine the total storage space occupied on your personal diskette, and write it down. Next, enter the command to erase the file **OWNER.SAV**, just renamed in the last section:

A:\>ERASE B:OWNER.SAV<cr>

Enter the appropriate **DIR** and **CHKDSK** commands to establish that the file **OWNER.SAV** no longer exists, and that the number of available bytes on the diskette has been increased as a result.

An alternate form of the **ERASE** command is **DEL** (for delete); DOS will accept either command.

Note: If you wish to erase all files present on a diskette, you can use a generic name, as in:

A:\>ERASE B:*.*<cr>

DOS will request confirmation of such a potentially damaging erasure with the message

Are you sure? (y or n)

before actually performing the operation. It is fairly easy to enter an equivalent command accidentally when not intending to do so, e.g., the command

A:\>**ERASE *PROG.*<cr>**

is effectively the same as **ERASE *.*** because of the DOS rules for analyzing generic names. Be careful when using the ERASE command !

2.19 Copying and Comparing Diskettes (DISKCOPY and DISKCOMP Commands)

You will probably have no need to completely duplicate your diskette. However, if you do, the DOS command prototype is:

DISKCOPY [d1:] [d2:]

Here, **d1:** and **d2:** are the source and target drives, respectively. In general, **d1:** should be different from **d2:**, although DOS allows the drive designations to be the same (in which case DOS will ask you to insert and remove the disks several times during the copying process, since the main memory is usually not large enough to store the content of an entire disk at one time). **DISKCOPY** is a transient command, and hence normally must be issued before removing the master diskette from Drive **A:**; in the FEC Laboratories, however, the transient DOS command files are stored on the Ethernet fileserver, so this caveat doesn't apply.

To copy your personal diskette in Drive **B:** to another diskette (to be inserted later into Drive **A:**), the command would be:

A:\>**DISKCOPY B: A:<cr>**

DOS will respond with the message:

Insert source diskette in drive B:

Insert target diskette in drive A:

Strike any key when ready

You can then remove the master diskette from Drive **A:** in a manner similar to that described in Section 2.16, and insert the diskette onto which you wish to copy your personal diskette. It is not necessary to format the target diskette before using the **DISKCOPY** command, since the command formats the diskette and copies at the same time. The target diskette is formatted identically with the source diskette, including any volume name (label). While copying, DOS displays the message:

```
Copying 2 side(s)

Formatting while copying

Copy complete

Copy another (Y/N)N<cr>
Insert DOS disk in drive A
And strike any key when ready
```

The DOS disk referred to here is the master diskette containing the principal DOS program files.

If you wish to compare two diskettes to see if they are identical in every respect, the DOS command is **DISKCOMP**. The command prototype is:

```
DISKCOMP [d1:] [d2:]
```

Here, as in the **DISKCOPY** command, the two drives should be different. If they are not, a cumbersome sequence of diskette insertion and removal operations is required. **DISKCOMP** is a transient command, and must normally be issued before removing the master diskette; in the FEC Laboratories, however, the transient DOS command files are stored on the Ethernet fileserver, so this restriction doesn't apply. If **d1:** and **d2:** are **A:** and **B:**, respectively, for example:

```
A:\>DISKCOMP A: B:<cr>
```

DOS responds with the message:

```
Insert first diskette in Drive A:

Insert second diskette in Drive B:

Strike any key when ready
```

The diskettes are compared on a track-by-track basis, and any mismatches are noted by specifying the side (immaterial for single-sided diskettes, 1 or 2 for double-sided diskettes) and track for which any mismatches occur. If the diskettes compare successfully, DOS will respond with:

```
Diskettes compare ok

Compare more diskettes (Y/N)?N<cr>
Insert DOS disk in drive A
and strike any key when ready
```

Remove the diskette from Drive **A:**, and reinsert the master diskette.

Note: A diskette whose files have been generated from from an original diskette using the **COPY** command, e.g.,

```
COPY B:*.* A:
```

will not necessarily be identical to the diskette that would be produced by the **DISKCOPY** command from the same original diskette; the storage sectors assigned to a given file may be different in the two situations.

2.20 Resetting the Date and Time (DATE and TIME Commands)

If you wish to reset the date and time after the initial DOS booting operations have taken place, you can do so with the DOS commands **DATE** and **TIME** which have the prototypes:

```
DATE [mm/dd/yy]
```

and **TIME [hh:mm:ss:xx]**

The entries **mm**, **dd**, **yy**, **hh**, **mm**, **ss**, and **xx** have the same significance as the same entries listed in Section 2.6.3, and will not be repeated here. If only **DATE** or **TIME** are entered, DOS will request the information exactly as shown in Section 2.6.3.

```
Examples:   A:\>DATE 9/14/87<cr>
            A:\>TIME 14:30<cr>
            A:\>DATE<cr>
            A:\>TIME<cr>
```

These commands are not needed for the IBM PCs in the Freshman Engineering Computing Laboratories, which are equipped with battery-powered calendar/clocks. The date and

time will be set automatically when DOS is booted, provided the clocks are themselves set properly. If, for some reason, you wish to reset the date and time from the AST clock, it is only necessary to load and execute the program file **astclock** by entering:

```
A:\>astclock<cr>
```

2.21 Tree-Structured File Directories

Most of the commands described in previous sections of this chapter are available in DOS 1.1. Exceptions that are only available in DOS versions 2.0 and greater are:

1. the **PRINT** command, and the file print queueing feature.
2. the /V parameter of the **FORMAT** command, the **VOL** command, and labels for volumes (e.g., diskettes).
3. the **CLS** command for clearing the screen.
4. the **GRAPHICS** command for printing graphics mode screen images
5. parameters for the **MODE** command related to color monitor control

Features available only in DOS 3.0 and greater are:

1. the **ATTRIB** command,
2. the **LABEL** command,
3. parameters related to quad-density floppy disk drives.

You will find that the DOS 1.1 commands plus the extra commands from DOS 2.0 and 3.0 listed above will be adequate for most of your computing needs, particularly if you are using a fairly standard IBM PC, However, all of the advanced features of DOS 2.0 and 3.0 are fully supported on the IBM PC (even those for any attached quad-density drives), and you may use them if you wish.

If your IBM computer has a hard disk , you will

probably want to use some of the more advanced features of DOS 2.0 and 3.0, related to tree-structured file directories described in this section.

File directories in DOS 1.1 are one-level directories, in that all of the files on a volume (e.g., a diskette) are stored under a single file directory. When you use the **DIR** command, DOS 1.1 displays the file directory information for all files on the diskette. This is usually adequate for diskettes, which have a limited capacity for storing files (no more than 64, 112, and 224 distinct files can be stored on single-sided, double-sided double-density, and double-sided quad-density diskettes, respectively). Even so, it is not easy to find a desired file from a large directory of say 100 names.

On a hard disk, the problem is more serious, since there is capacity for storing thousands of files on a single disk. Clearly there is a need for breaking the disk directory into more manageable pieces. In DOS versions 2.0 and greater this is accomplished by using a "multi-level" file directory structure that has the appearance of an inverted "tree", and is called a tree-structured or hierarchical file directory. At the base of the inverted tree is the root directory. You can think of the root directory as corresponding to the entire directory in DOS 1.1.

The root directory has no name, but is designated by the backslash \. For example, to get a directory listing of the root directory on the hard disk (Drive **C:** of the PC-XT or PC-AT), the **DIR** command would be entered (assuming that Drive **C:** is the default drive) as:

C:\>DIR \<cr>

If **A:** (the floppy disk drive) were the default drive, then the entry would instead be:

A:\>DIR C:\<cr>

The root directory for a given drive can have a large number of subdirectories at level 1 of the tree structure. Each subdirectory is assigned a directory name consisting of from one to eight characters with an optional name extension of a period and up to three characters. Thus the rules for naming subdirectories are the same as for naming files (in fact, the subdirectories are files that contain naming and allocation information about other files assigned to the subdirectory).

The subdirectories can in turn have their own subdirectories, leading to a full tree structure. A file that is to be stored on the volume can be assigned to any subdirectory in the overall directory structure. Consider,

for example, the directory structure shown in Figure 2.3. This is a four-level directory in which subdirectories **FORTRAN**, **LETTERS**, and **SCHOOL** are all at level 1 with respect to the root directory, which is considered to be at level 0. The **SCHOOL** subdirectory contains just one file, **GRADES.F**, and has two subdirectories, named **PAPERS** and **EXAMS.UM**. Subdirectories **PAPERS** and **EXAMS.UM** are at level 2 with respect to the root directory and level 1 with respect to subdirectory **SCHOOL**.

A subdirectory need not have any user files; it can still have subdirectories of its own (e.g., **LETTERS.86** and **BUSINESS** in the figure).

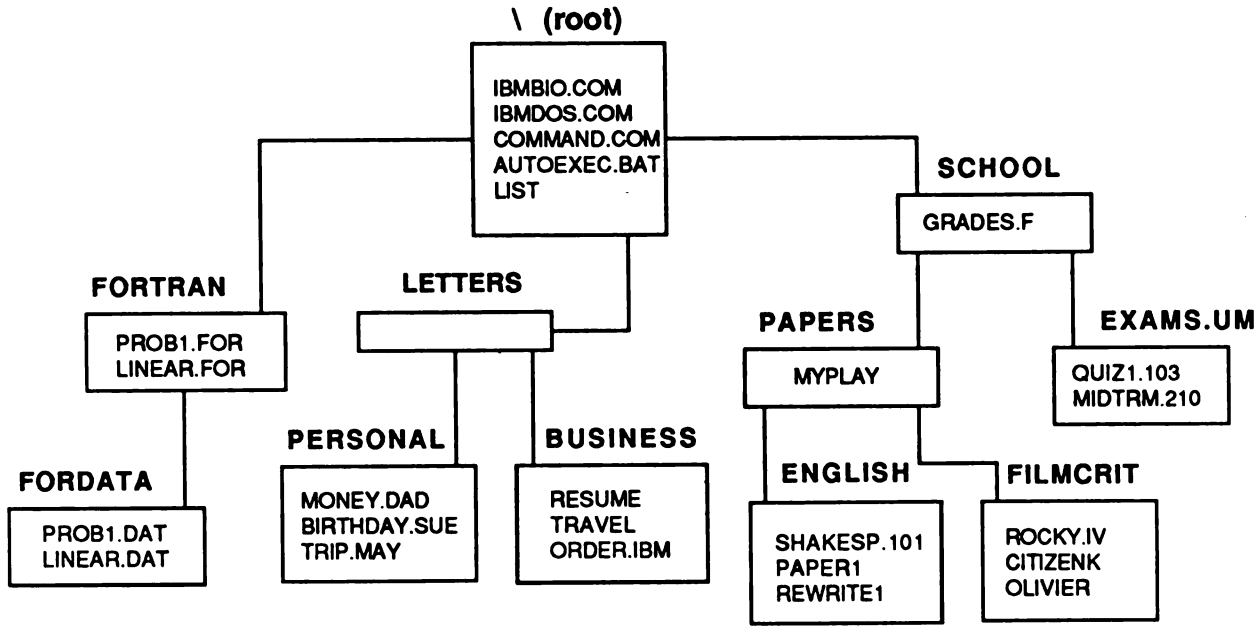

Figure 2.3

Example of a DOS Tree-Structured File Directory

One of the subdirectories on each volume (e.g., on the hard disk of the PC-XT or PC-AT) is denoted as the current directory for the volume. If any reference is made to a file on that volume without other information about the file's subdirectory location, DOS looks only in the current directory for the file. Consider the directory structure of

Figure 2.3. If **PAPERS** is the current directory on Drive C:, and Drive C: is the default drive, then entry of the command

C:\>**DIR**<cr>

will cause DOS to display directory information for just one file, **MYPLAY.**

On the other hand, if **FILMCRIT** were the current directory on drive **C:** but another drive, say **A:** were the default drive, then entry of the command

A:\>**DIR C:**<cr)>

would cause directory information for only files **ROCKY.IV**, **CITIZENK**, and **OLIVIER** to be displayed.

Every drive (volume) has a current directory, even though there is only one default drive at any given time. It is very important that you understand the difference between default drive and current directory if you intend to make use of hierarchical file organization.

2.21.1 File Directory Paths (TREE Command)

The commands for creating subdirectories and for making a particular subdirectory on a volume the current directory for that volume are discussed in the next section. Assume for the moment that a volume with the file structure shown in Figure 2.3 already exists and is stored on Drive C:, and that subdirectory **PAPERS** is the current directory. How does one locate files that are in other subdirectories on the volume?

DOS requires that you specify a path of directory names that leads to the target file, starting at either the root directory or the current directory for the volume. The technique for specifying a path from the root directory is probably best illustrated by an example. Suppose you are interested in the file **CITIZENK** in subdirectory **FILMCRIT.** Then the path based at the root directory is:

\SCHOOL\PAPERS\FILMCRIT

Here, the first \ indicates that the path starts at the root directory.

Since subdirectory **PAPERS** is the current directory, the path could also be specified more simply by starting at subdirectory **PAPERS:**

FILMCRIT

The absence of an initial backslash in the path means that the path starts at the current rather than the root directory.

The file **CITIZENK** itself would be referenced as either:

\SCHOOL\PAPERS\FILMCRIT\CITIZENK

or **FILMCRIT\CITIZENK**

For example, if you wanted to print a copy of the file **CITIZENK** on the printer, you could enter the command:

C:\>PRINT FILMCRIT\CITIZENK<cr>

If the default drive were **A:** instead of **C:**, then the printing could be initiated with either of the following commands:

A:\>PRINT C:FILMCRIT\CITIZENK<cr>

or **A:\>PRINT C:\SCHOOL\PAPERS\FILMCRIT\CITIZENK<cr>**

If the desired file is in the parent directory of the current directory, then it is also possible to "back up" one level by using two periods in the path description. For example, the file **GRADES.F** in the **SCHOOL** subdirectory could be accessed from the current directory **PAPERS** as:

..\GRADES.F

Even more obscure paths are possible. For example, if **ENGLISH** were the current directory, the file **QUIZ1.103** could be referenced as either:

..\..\EXAMS.UM\QUIZ1.103

or **\SCHOOL\EXAMS.UM\QUIZ1.103**

In general, we recommend that the file organization be kept fairly simple, with only one or two levels below the root level (remember, the tree is upside down!).

The DOS **TREE** command can be used to get a full listing of directory paths to all subdirectories on the volume. The listing is directed to the monitor screen, but can be redirected to another device or file, as described in Section 2.25). This transient command has the prototype:

TREE [d:][/F]

Here, **d:** is the drive containing the volume of interest (if absent, the default drive is assumed). If **/F** appears, then all of the file names in each subdirectory are listed under the corresponding directory path. For example, if the default drive is **A:**, then a complete picture of the file structure for the volume on Drive **C:** will be produced with the command:

A:\>TREE C: /F<cr>

2.21.2 Changing the Current Directory (CHDIR Command)

The current directory on any specified drive can be changed with the resident command **CHDIR** (CHange DIRirectory), which can be abbreviated to **CD**. The prototype for the command is:

CHDIR [d:][path]

Here, **d:** is the drive designator, and **path** is a directory path from either the root directory of the drive or from the current directory (at the time the command is issued). The specified path is exactly the one described in Section 2.21.1.

For example, suppose that Drive **A:** is the default drive, and that the tree-structured directory of Figure 2.3 is in Drive **C:** with **LETTERS** as the current directory. Then subdirectory **PERSONAL** could be made the new current directory on Drive **C:** with either of the commands:

A:\>CHDIR C:\LETTERS\PERSONAL<cr)>

A:\>CD C:PERSONAL<cr>

If you wanted to set the current directory of default Drive **A:** to its root directory, then the command would be:

A:\>CD \<cr>

If no path is specified in the command, then the path for the current directory of the indicated drive (the default drive, if the drive designator is not included either) is displayed on the monitor.

2.21.3 Setting the Directory Search Path (PATH Command)

When using a drive (or drives) with tree-structured directories, the task of remembering the various subdirectory paths for locating often-used program files and batch files becomes rather formidable. This is especially true for hard disks, which may contain hundreds or thousands of files and a large number of subdirectories. It is very useful to be able to refer to frequently used files, such as editors, compilers, electronic spreadsheet programs, and the DOS transient command files without including their file subdirectory paths or taking into account the currently assigned default drive and the current directory for each of the drives.

Thus, for certain commonly used files, one would like to refer to the file name alone, and let DOS find the file, even when the file is not on the current directory of the default drive (that is normally what would be implied when no directory path is supplied to locate the file in the overall directory structure).

Fortunately, DOS versions 2.0 and greater have a command called **PATH** that allows users to handle this problem fairly easily. The general prototype for this resident command is:

PATH [d1:][path1][;[d2:][path2]] ...

Here, **d1:**, **d2:**, etc. are drive designators, and **path1**, **path2**, etc. are associated file subdirectory paths. The **PATH** command allows you to assign the directory search path for DOS, where the search path consists of all of the individually specified file subdirectory paths appearing in the path list of the **PATH** command. A semicolon is used to separate individual subdirectory paths in the list.

Once the search path is set, DOS takes the following action when a command to process a program (**.EXE**) or batch (**.BAT**) file refers to a file name that has no specified subdirectory path (not even a drive assignment):

1. It first assumes that the file is in the current directory of the default drive. If the file is found, the directory search stops.

2. If the file cannot be found in step 1, DOS next checks the first file subdirectory path appearing in the **PATH** command parameter list. If the file is found in the corresponding subdirectory, searching stops.

3. If the file cannot be found in steps 1 and 2, the search continues with the next subdirectory in the list, etc.

The search continues from subdirectory to subdirectory until either: (1) the file is found, or (2) all subdirectories in the list are searched and the file is not found. In the latter case, DOS issues a "file not found" message.

As an example, consider the file structure of Figure 2.3 again. Suppose that the root directory, and subdirectories **SCHOOL** and **BUSINESS** are to be put into the directory search path, and that the volume is mounted on Drive **B:**. Let **A:** be the default drive. Then the appropriate **PATH** command would be:

A:\>PATH B:\;B:\SCHOOL;B:\LETTERS\BUSINESS<cr>

Now, the any program or batch files in the **SCHOOL** and **BUSINESS** and root directories of Drive **B:** will be found, even if only their names alone are specified, regardless of the current directory assignment for drive **B:** or the current default drive assignment **A:**. Should files with the same name be assigned to different subdirectories in the directory search path, the one in the first-mentioned subdirectory (in the **PATH** command) will be used.

If the **PATH** command is entered without parameters, simply as:

A:\>PATH<cr>

then DOS will display the current active search directory on the monitor screen (try it!).

If the **PATH** command is entered with a terminating semicolon and with no drive specification at all, as in

A:\>PATH;<cr>

then DOS resets the search path to null (i.e., only the current directory of the default drive will be searched for future file references having no drive designator or file directory path).

2.22 Creating and Removing Subdirectories (MKDIR and RMDIR Commands)

You can create a new subdirectory with the resident DOS command **MKDIR** for which the prototype is:

MKDIR [d:] path

Here, **path** is a directory path as described in Section 2.21.1. The command may be abbreviated to **MD**, and, if Drive **d:** is not specified, the default drive is assumed.

Suppose that you want to create a new subdirectory named **MONKEY** that will be a subdirectory of the subdirectory **BUSINESS** in the file directory structure shown in Figure 2.3. Assume that the default drive has been changed to **B:>** (see Section 2.7.3), that the volume is on drive **B:**, and that the current directory of the volume is **BUSINESS**. Then either of the following commands could be used:

B:\>MD MONKEY<cr>

B:\>MKDIR \LETTERS\BUSINESS\MONKEY<cr>

A subdirectory that has no files and no subdirectories can be removed from the directory structure with the resident DOS command **RMDIR**, which can be abbreviated to **RD**. The command prototype is:

RMDIR [d:] path

where **d:** is the drive designator and **path** is the directory path to the subdirectory that is to be removed; the last subdirectory in the path is the one that will be removed.

Note: Neither the current directory nor the root directory can be removed.

As an example, suppose that you wished to remove the subdirectory **PERSONAL** and to save the three files currently there in the subdirectory **LETTERS** before doing so. Assume also, that the volume is on default drive **C:**.

```
C:\>CD \LETTERS<cr>
C:\>COPY PERSONAL\*.*<cr>
C:\>ERASE PERSONAL\*.*<cr>
C:\>RMDIR PERSONAL<cr>
```

The first of these commands changes the current directory to **LETTERS**, the second copies the three files (using the generic file name ***.*** from subdirectory **PERSONAL** to the current directory **LETTERS** (the new files will have the same names), the third erases the three files in subdirectory **PERSONAL**, and the last removes subdirectory **PERSONAL** from the directory structure.

The remainder of this chapter will deal with some additional advanced features of DOS that you may find useful, once you have the basics in hand.

2.25 Redirection of Standard Input and Output

DOS allows you to redirect input and output from standard Input/Output devices (e.g., the keyboard **CON:** and display screen **CON:**). This means that output that would normally be sent to the screen, for example, can be routed to other devices such as the line printer or a file on a disk drive. For output the symbols > or >> are used to indicate the new target for the output.

For example, the command:

```
C:\USER>DIR A:<cr>
```

would normally cause the directory information for the current directory in the diskette in Drive A: to be displayed on **CON:**, the monitor screen. If you want the output to be sent to the printer or to a file on Drive A: named **DRIVEA.DIR** instead, you could enter

```
C:\USER>DIR A: > LPT1:<cr>
```

or

```
C:\USER>DIR A: > A:DRIVEA.DIR<cr>
```

respectively. If file **DRIVEA.DIR** does not exist, DOS will create it automatically. If **DRIVEA.DIR** already exists, the redirected information will be written starting at the beginning of the file; any previous contents of the file will be lost.

If you wished the output to be appended to information already written in the file **DRIVEA.DIR** (instead of starting

the writing operations from the beginning of an empty file), then the latter command would be written as:

```
C:\USER>DIR A: >> A:DRIVEA.DIR<cr>
```

Input can be similarly redirected using the symbol <. For example, if a program named **MYPROG** present in subdirectory **USER** on Drive **C:** expects input from the keyboard, but you choose to have the input taken from the file **MYDATA** in the subdirectory **USER** on Drive **C:** instead, and you want the output, normally written on the monitor screen, to be written to a file named **MYRESULT** in the current directory of a diskette in Drive **A:**, then the command would be:

```
C:\USER>MYPROG < C:\USER\MYDATA > A:MYRESULT<cr>
```

2.26 DOS Filters

DOS has a set of programs called filters that read data from standard input devices (or other devices or files specified with the redirection capabilities described in Section 2.25), modify the data in some way, and then write the results to standard output devices (or other devices or files specified with the redirection feature). The most important filters are **MORE**, **SORT**, and **FIND**. Since the filters are programs that must be loaded into memory from diskette or hard disk, they appear to the user to be transient DOS commands.

2.26.1 The MORE Filter

The **MORE** filter program reads data lines from the input device or file and displays them on the screen, one full screen at a time. After each full screen, DOS pauses with the message

```
--- MORE ----
```

Pressing any character key causes the next screen-full of information to be displayed.

This filter is a good one to use in place of the **TYPE** command of Section 2.13, which continues to scroll the output until all lines are displayed. An example, which would cause all lines from the file **A:MYFILE** to be displayed one full screen at a time is:

```
C:\USER>MORE < A:MYFILE<cr>
```

2.26.2 The SORT Filter

The **SORT** filter is a DOS program that is invoked with the prototype command

```
SORT [/R] [/+n]
```

SORT sorts its input into alphabetical or numerical order (reverse order if the parameter **/R** appears) starting in a specified column **n** (column 1 if **/+n** does not appear).

For example, the command

```
C:\USER>SORT /R/+7 < A:MYFILE >> A:OUT<cr>
```

will cause DOS to sort lines in the file **A:MYFILE** into reverse numerical or alphabetical order based on characters in the file starting in column 7. The sorted output will be redirected and appended to the file named **A:OUT.**

Note: The **SORT** filter does not distinguish between upper and lower case letters in alphabetical ordering, and the output file must be different from the input file.

2.26.3 The FIND Filter

The **FIND** filter has a DOS prototype of the form

```
FIND [/V][/C][/N]"string"[[d:]filename]
```

This command causes DOS to search for lines in the file named **filename** on Drive **d:** that contain the characters in **string** (enclosed in double quotes). If **/V** appears, the lines that do not contain "string" will be displayed. If **/C** appears, the count of the number of lines containing "string" (but not the lines themselves) will be displayed. If **/N** appears, the line number of each matching line will be displayed with the content of the lines from the file.

For example, if the command

```
C:\USER>FIND /N "GOD" A:GENESIS > LPT1:<cr>
```

is entered, DOS will print the line number and content of all lines from the file **GENESIS** on Drive **A:** that contain the word **GOD.**

2.27 Piping of Standard Input and Output

DOS also has a powerful piping feature that allows the screen output of one program or filter to be used in place of the keyboard input of another program or filter. DOS creates temporary intermediate files on the root directory of the default drive to hold the information that is being piped from one program to another. Since redirection (see Section 2.25) is allowed, the output from virtually any program can be used as input for another. The character | is used to indicate piping of Input/Output. Some examples should help to clarify this feature of DOS.

Example 1:

```
C:\USER>DIR A:|FIND "DIR"|SORT /R > LPT1:<cr>
```

This command will cause the directory information for Drive **A:**, (normally it be sent to the screen) to be piped as input to the **FIND** filter instead. The **FIND** filter looks for lines in the directory that contain the characters "DIR" (these are the lines that contain the names of subdirectories in the tree-structured file directory). The output from the **FIND** filter, which would normally be sent to the screen is instead piped as input to the **SORT** filter. The **SORT** filter will sort the subdirectory names (the names start in column 1 of the lines) into reverse alphabetical order. The output from **SORT**, normally displayed on the screen, is redirected to the line printer **LPT1:**.

Example 2:

```
C:\USER>A:MYPROG1|SORT|A:MYPROG2 > A:ANS<cr>
```

This command will cause the screen-directed output from the program in the file **A:MYPROG1** to be written to a temporary file in the root directory of drive **C:**. This output file is then sorted (starting in column 1) by the **SORT** filter; the sorted information is then written into another temporary file in the root directory of drive **C:**. Finally, the sorted file is used in place of keyboard input for the program from the file **A:MYPROG2**; screen-directed output is redirected to a file named **ANS** on drive **A:**.

2.28 DOS - The Future

Because of its adoption by IBM as the preferred operating system for the IBM PC, PC-XT, PCJr, PC-AT, and PC Convertible computers, PC/DOS is now the most-used operating systems for microcomputers. With support for hierarchical file directories, redirection of input and output, filters, pipes, queueing of file printing, and other features not described in this text, the latest versions of DOS have taken on some of the attributes of operating systems for much more powerful computers. In particular, many of the newer features show the influence of the UNIX operating system on the PC/DOS system designers.

The future of DOS seems assured because of the very large customer base already in place and continued support by both Microsoft and IBM. More powerful versions of PC/DOS that involve support for multitasking (the running of more than one computing task at the same time by a single user), multiuser operation (the simultaneous use of a single PC by more than one user), and distributed computing (access and simultaneous use by a single user of more than one PC attached to a local area network) will certainly be introduced in the relatively near future (a one to two year time frame).

Stay tuned!

CHAPTER 4

RUNNING FORTRAN-77 PROGRAMS ON THE IBM/PC

4.1 FORTRAN - A History

FORTRAN (short for FORmula TRANslation) was the first symbolic language of the procedure-oriented type (see Section 1.3). The earliest version of the language, now called FORTRAN-I, was developed at one of IBM's research laboratories in the mid-1950s. An improved version, called FORTRAN-II, was introduced in 1958 for use on IBM's first large scientific computer, the IBM 704.

Virtually every other computer manufacturer produced translating programs called compilers (see Section 1.3) for versions of the language for use on their own machines, and by the time the IBM 360 family of computers came onto the market in the mid-1960s, FORTRAN was well entrenched as the major computing language for scientific and engineering applications (another language, called COBOL, played a similarly dominant role for business applications).

IBM released translators for a much-improved, but compatible, version of the language called FORTRAN-IV with the introduction of the 360 computers. There were (and still are) many variants or dialects of the FORTRAN-IV language in use on different computers or by different industrial and academic groups. In 1966, ANSI, a professional standards group, prepared definitions for two FORTRAN standard languages, called the full language (intended for implementation on large computers), and the subset language (for implementation on small computers). The subset language is "upward" compatible with the full language (i.e., programs written in the subset language also satisfy all the syntactical rules for the full language). These standard languages are called FORTRAN-66, but the labels "FORTRAN-IV" and "FORTRAN-66" are used more or less interchangeably.

4.2 FORTRAN-77

In 1977, the ANSI group formulated a new set of standards for a considerably improved structured version of FORTRAN. Here, "structured" refers to a programming concept first formulated by Prof. Niklaus Wirth of the Technical University of Zurich (the designer of the procedure oriented language PASCAL and of the newer Modula-II language).

Roughly speaking, the structured approach to programming consists of breaking the overall problem into individual tasks or "modules" based on notions of iteration, sequencing, and selection. Individual modules are arranged in a hierarchial structure, each module having just one entry point (starting place) and one exit point. Control is passed from module to module in a downward sequence, with few if any unconditional branches to higher levels of the structure. Structured programs have, for example, very few GOTO statements, and tend to be much easier to "read" (in the sense of reading text in a natural language from top to bottom of a page) than nonstructured ones. Flow diagrams are not much used for describing structured algorithms because of this natural forward sequencing in the logic.

The structured version of FORTRAN defined by the ANSI group is now called FORTRAN-77. Again the standard consists of two definitions, a full implementation language and an upwardly compatible subset language. This version of the language is now the most popular one.

Interestingly, the FORTRAN-77 language standards are defined in such as way that, with few exceptions, programs written in standard FORTRAN-66 (an unstructured language) are upward compatible with FORTRAN-77 (a structured language). Therefore, it is possible to write unstructured as well as structured programs using FORTRAN-77 (the syntax for languages such as PASCAL tends to enforce the writing of structured programs). For this reason, many computer scientists consider FORTRAN to be an archaic programming language. Nevertheless, FORTRAN is still the most-used computing language for numerical calculations in engineering and the sciences.

Most "good" programmers use structured approaches to program formulation insofar as possible, regardless of the programming language used. In this context a "good" program is by definition a program that not only "works" (i.e., produces correct answers), but one whose logic is clear and can be understood easily by others. This latter characteristic is critically important when one programmer (or group of programmers) writes programs but different people maintain and improve them later on (a common

situation in almost all industrial, governmental, and academic organizations).

4.4 FORTRAN-77 Compilers for the IBM/PC

FORTRAN compilers for personal computers are not very common, BASIC being the most used programming language for microcomputers. However, quality compilers have come onto the market during the past few years. The most popular of these are: (1) Microsoft FORTRAN compiler, for the subset version of FORTRAN-77, (2) Professional FORTRAN compiler (by Ryan-McFarland), for a full implementation version of FORTRAN-77, and (3) Microsoft FORTRAN Optimizing compiler, another full-implementation version of FORTRAN-77.

We have adopted Microsoft's Optimizing FORTRAN 77 compiler for use in the introductory computer course for engineering students because: 1) it supports all of the structured features of ANSI standard FORTRAN 77, 2) it is compatible with previous versions of FORTRAN (still very popular with scientists and engineers, with an enormous base of well tested programs), 3) it is upwardly compatible with Microsoft's subset FORTRAN compiler used on the CAEN Laboratory IBM/PCs, and 4) it is fully compatible with the MTS version of the IBM mainframe FORTRAN-77 compiler (the ***FORTRANVS** compiler available on the IBM 3090).

We have authored a companion text, FORTRAN-77 (with MTS and the IBM PC) that covers algorithm development and programming using the ANSI Standard FORTRAN-77 language, with emphasis on the Microsoft implementation for the IBM PC, PC-XT and PC-AT. Henceforth, when we refer to Microsoft (MS) FORTRAN we will be referring to the optimizing full-implementation compiler.

The intent of this chapter is to familiarize you with use of this compiler on IBM/PCs.

Throughout this chapter, we assume that your FORTRAN-77 program has already been written, and that it has been stored on a diskette in a DOS file with the file-name extension **.FOR**. The particular extension **.FOR** is required by the MS compiler (it also helps to point out which files contain FORTRAN programs). Throughout this chapter, we will call the DOS diskette file containing the source FORTRAN-77 program **PROG.FOR**.

4.5 Compilation, Linking, and Execution

The translation of your FORTRAN source program and its eventual execution by the computer is a three-step process, as described briefly in Section 1.3 and illustrated in Figure 1.3.

Step 1: Compilation is the process that translates an algorithm represented in the symbolic FORTRAN language (the source program) into an equivalent algorithm in the machine's language called the object program. There may be several individual source programs (called subprograms in FORTRAN parlance) involved in a "package" or "suite" of programs required to solve a problem. Typically, the compiler treats each subprogram separately to produce individual object programs (the individual source programs can even be translated at different times).

Step 2: Linking is the process that gathers together the individual object programs, along with any library programs, and produces a single interconnected machine language program called the object module. Library programs perform commonly needed tasks like trigonometric function evaluation, and are already available in object form, usually on disk storage in a library with name extension **.LIB**.

Individual machine language programs are usually compiled into relocatable form, meaning that they can be moved (relocated) as blocks of instructions to any block of addresses in the main store, provided there is sufficient unused memory available. The process of linking the individual programs together to create the object module involves a significant amount of address recalculation by the linker, since any addresses associated with operands in the program must be modified to account for the relocation of the programs in the memory.

Step 3: Execution involves the loading of the object module produced by the linker into the computer's main store (the program that does the memory loading is called a loader) and then turning over control of the central processor to it. The executing program reads the user's data from input devices or files and writes results for display or printing on output devices or for storage in a file(s).

The three steps, viewed from the standpoint of the programs involved and their inputs and outputs are:

Compilation	Program:	compiler
	Input :	source language programs
	Output :	relocatable object programs
Linking	Program:	linker
	Input :	relocatable object programs from compilation and library
	Output :	object module
Execution	Program:	object module
	Input :	user's data (from keyboard, disk files)
	Output :	user's results (to monitor, printer, disk files)

The responsibility for actually loading each of the programs (compiler, linker, object module) into the main store is assumed by yet another program called the loader, one of the operating system (e.g., DOS) programs. In some cases (almost always, in large computing systems), the linking is done in conjunction with the loading, and the program that does the dual job is called a linking loader. For microcomputers, the linker tends to be tied more closely to the compiler than to the operating system, and the compiler manufacturer distributes a linker with the compiler as part of the software package. This is the case for Microsoft's FORTRAN-77 compilers.

4.6 The Microsoft Compiler/Linker for the IBM PC

In Microsoft's implementation of the compiler/linker, the compilation task involves either two or three passes, i.e., the compilation "step" is in fact more than one step involving processing by two or more programs, all of which constitute the compiler. The two (or three, depending on some user options, such as code optimization) passes are:

Pass 1 Translates the source program, stored in the file **PROG.FOR** for example, into an intermediate

form, consisting of some temporary files that will be used by later passes of the compiler. It also produces a program-listing file, whose name has the extension **.LST** (**PROG.LST**, for example), which is important because it identifies and explains, as far as possible, any syntactical (language) errors in the source program.

Pass 2 — Takes the intermediate files produced in Pass 1 and generates the relocatable object code. The output from Pass 2 is an object program (or programs) that is (are) stored in a file whose name has the extension **.OBJ** (**PROG.OBJ**, for example). Also deletes the intermediate files from Pass 1, and generates more intermediate files that may be needed by Pass 3.

Pass 3 — Optimizes the object code, improving the efficiency of calculation of certain expressions, particularly if they have elements in common. Can also produce an optional listing of the object program (in hexadecimal code) in a file with the name extension **.COD**, or of an assembly language version of the program in a file with the name extension **.ASM**, and/or of a file containing detailed information about all of the programs in an object module in a file with name extension **.MAP**. Since very few programmers need such listings, we will ignore this latter feature.

The linker then takes the individual relocatable object programs produced by the compiler, stored in the file (or files) with extension **.OBJ**, links it (them) with the necessary library programs (present in one or more library files with the name extension **.LIB**), reassigns memory addresses, and generates a final object module that can subsequently be loaded into the fast memory of the IBM/PC and executed. The output from the linker is in a file whose name has the extension **.EXE** (**PROG.EXE**, for example).

As supplied by Microsoft, the compiler files are stored on six double-sided diskettes. Although it is possible use floppy disk versions of the compiler and linker on a dual-floppy drive IBM PC, it is a very tedious process. In essentially all cases, one wants to have the compiler/linker files stored on the hard disk of a PC-XT or PC-AT, or for networked machines to have the files stored on the hard disk of the network file server(s). The most important of these files are listed in Table 4.1.

Table 4.1 Important Files in the MS FORTRAN-77 Compiler

File Name	Size (bytes)	Purpose
F1.EXE	150,343	Pass 1 of the compiler.
F1.ERR	15,776	Error messages for pass 1.
F2.EXE	185,117	Pass 2 of the compiler.
F3.EXE	126,667	Pass 3 of the compiler.
F23.ERR	2,914	Error messages for pass 2 and 3.
FL.EXE	25,011	Compiler manager.
FL.HLP	1,584	Help messages for manager.
FL.ERR	1,817	Error messages for manager.
F3S.EXE	82,629	Alternate pass 3 when optimization is disabled.
LINK.EXE	50,531	Linker.
LLIBFORE.LIB	194,048	Library file that supports coprocessor (e.g., Intel 8087) if present; otherwise, floating-point calculations are done using software.

Depending on the options desired, there are several ways of calling the various programs that constitute the Microsoft compiler. The sequence of compilation activity is controlled by the "manager" program in the file **FL.EXE**, in response to input from the user.

Some of the more important files that are used or produced (some optionally) by the compiler are shown in Figure 4.2.

Table 4.2 Symbolic Names Used by Compiler

Name	Meaning
progfile.FOR	File containing FORTRAN source program.
progfile.LST	File that will contain the program listing and diagnostics generated by the compiler.
progfile.OBJ	File that will contain the relocatable object program generated by the compiler.
progfile.ASM	File that will contain the assembly language version of the program.
progfile.COD	File that will contain the machine language version of the program.
profile.MAP	File that will contain memory mapping information about the object programs generated.

Here **progfile.FOR** is your original FORTRAN source program file, e.g., **PROG.FOR**. The file **progfile.LST** will contain a listing of your FORTRAN program statements, any diagnostic error messages describing errors in FORTRAN syntax, and other information such as a listing of all FORTRAN variables, arrays, and functions and subroutines referenced in the source program and total memory space required. The other files are in general not of interest to ordinary mortals.

If your source program contains an error (usually caught by pass 1 of the compiler), then the compilation/linking process will be aborted, and you can examine the listing file to determine the nature of any errors found. An Editor can then be used to make corrections in the file **progfile.FOR**, and the compilation/linking operations can be attempted again.

Table 4.3 lists the symbolic names of the most important files involved in the linking operation (when **LINK.EXE** is being executed).

Table 4.3 Symbolic Names Used in Linking

Name	Meaning
progfile.OBJ	File(s) that contains the object program(s) produced by the compiler.
library.LIB	File(s) containing object versions of the library programs. The major library file is LLIBFORE.LIB, but other library files may also be used under some circumstances, depending in particular on the availability of a coprocessor.
progfile.EXE	File that will contain the final executable object module. available.

The most important file from the user's standpoint is the final executable load module file **progfile.EXE**. The file **progfile.OBJ** may be of interest if other object program files, produced with a different execution of the compiler, are to be linked together.

APPENDIX A

QUICK REFERENCE GUIDE FOR PC/DOS FILES, DEVICES, AND COMMANDS

A.1 Introduction

This appendix summarizes the most important features of PC/DOS 3.1 (sometimes called MS/DOS 3.1), the disk operating system for the IBM PC, PC-XT, and PC-AT used on most PCs at The University of Michigan. DOS file and device naming conventions, DOS commands, and the booting (startup) procedure are covered only briefly here. For detailed descriptions of the characteristics of DOS, see the appropriate sections of Chapter 2. The description here assumes that the default drive (see Section A.5) starts out as **A:**, the usual case for the IBM PC. On the PC-XT and PC-AT, the initial default drive is almost always **C:**, and the DOS prompt characters in the examples should be modified accordingly.

A.2 DOS Files

DOS files are stored on diskettes or hard disk (PC-XT, PC-AT) and can be classified into the following principal types:

text file Contains lines of up to 255 ASCII encoded characters (source programs, data, documentation, etc.).

program file Contains object (machine language) program code in binary form.

batch file A text file that contains DOS commands. DOS reads the commands from a batch file and executes them (i.e., the commands are entered from the file rather than the keyboard). If an **autoexec** batch file (a batch file with the file name **AUTOEXEC.BAT**) is present on the master diskette in Drive **A:** of the IBM PC or in the root directory (see Section A.3) of the PC-XT or PC-AT when DOS is booted,

all of its commands will be executed immediately, before DOS will accept any user commands from the keyboard.

A DOS file name consists of from one to eight characters, preferably alphanumeric. There is no distinction between upper and lower case letters. The name can be followed by a file name extension consisting of a period and from one to three alphanumeric characters. Most users consider the file "name" to consist of both the file name and file name extension. The following are valid DOS file names:

```
MYFILE.103
X.Y
COMMAND.COM
```

In some cases (particularly for copying and erasing more than one file at a time), global or generic file names such as

```
AB*.E*
```
and
```
PROG?.FOR
```

may be used. Here, ? stands for any single legitimate character and * stands for one or more legitimate characters.

A file name may be preceded by a drive designator **d:** (either **A:** or **B:** or **C:**, etc.), to specify the disk drive containing the diskette where the file is stored, e.g.:

```
B:MYFILE.103
A:VEDIT
```

A.3 DOS Directories

DOS maintains a file directory for each volume (diskette or hard disk drive) of secondary storage, including file name, date of last change, size in bytes, and track and sector allocation on the disk. In many cases, particularly for diskettes, there will be a single directory for all files on the volume. However, DOS allows a hierarchical tree-structured file directory in which each volume has a base-level root directory denoted by \, and any number of subdirectories whose naming conventions are the same as for files. These are especially useful for hard disks with large numbers of files.

The root directory can have any number of subdirectories at "level 1"; these subdirectories can, in turn, have subdirectories of their own, which will be at "level 2" with respect to the root directory. These "level 2" subdirectories can, in turn, have subdirectories of their own, etc. A file can be assigned to the root directory or to any subdirectory in the overall structure.

Either the root directory or one of the subdirectories is denoted as the current directory for that volume; every volume has a current directory. Each file on a particular volume can be be located in the overall directory structure by specifying a directory path for it; here, a directory path is a string of subdirectory names that leads to the appropriate subdirectory from either the root directory or the current directory, as described in Section 2.21.1. Any file named without an associated directory path is assumed to belong to the current directory for the volume.

A.4 DOS Devices

The most important DOS device names are:

LPT1: or **PRN:** The attached Line PrinTer

CON: The CONsole, either the keyboard for input or the monitor screen for output.

A.5 Booting DOS

To perform a cold boot (starting up with the IBM/PC initially turned off):

1. Open the drive covers and remove any diskettes already present.
2. Insert the master diskette into Drive **A:** (on the left), and close the drive door.
3. Insert your personal diskette into Drive **B:** (on the right) and close the drive door.
4. Turn on power to the main unit, printer, and monitor.
5. The IBM PC will make some internal checks of its memory and circuitry, and then load (boot) the

major DOS program files from the master diskette automatically.

6. Enter the time and date if requested (in the form, for example: 9/14/87 (for September 14, 1987) and 13:22 (for 1:22 PM). If the computer is equipped with a battery-powered calendar/clock, this step will be omitted.
7. DOS will respond with the prompt character **A:\>**, indicating that Drive **A:** is the default drive.

To reboot the DOS program files when the computer is already turned on (called a warm boot), the master and personal diskettes should be in Drives **A:** and **B:**, respectively (see steps 1 and 2 above). Then type

<Ctrl-Alt-Del>

(hold down all three keys simultaneously). DOS will reload, request the date and time as indicated in step 5 above, and display the **A:\>** prompt, indicating readiness to accept a DOS command.

The procedure for booting the PC-XT or PC-AT is similar, except that the system programs are normally loaded in the root directory of the hard disk, Drive C:. Your personal diskette should not be in the floppy drive when the power is turned or (or when a warm boot is attempted). After the booting of the operating system is complete, insert your personal diskette into the floppy disk drive (Drive **A:**).

A.6 Changing the Default Drive

When DOS is loaded into the IBM/PC memory, its program files are read from the Drive **A:** (master) diskette. Subsequently, DOS displays the **A:\>** prompt whenever it is expecting a DOS command to be entered from the keyboard. The **A:\>** prompt indicates that any reference to a file without a drive designation, for example,

A:\>VEDIT<cr>

means that the file is present on Drive **A:**. In addition, the program files for any transient DOS commands (see next section), are assumed to to be present on Drive **A:**. Any files on Drive **B:** must be prefaced by a drive designation (e.g., **B:PROB3.DAT**), as in

A:\>TYPE B:PROB3.DAT<cr>

To change the default drive to **B:**, simply enter **B:** immediately following the **A:\>** prompt.

A:\>B:<cr>

DOS will then display the prompt **B:\>**. When **B:** is the default drive, any files (including transient command program files) on the Drive **A:** diskette must be prefaced by the **A:** drive designation, for example,

B:\>A:VEDIT<cr>

The default drive can be switched back to **A:** by entering **A:** following the **B:\>** prompt as in

B:\>A:<cr>

A.7 Executing a DOS Program File or Command

To execute a program file named **progfile** when **A:** is the default drive, simply enter

A:\>progfile<cr> if the file is on Drive **A:**

or **A:\>B:progfile<cr>** if the file is on Drive **B:**.

It should be noted that many of the most-used DOS commands are resident in memory once DOS is loaded (these are called resident or internal commands) and can be processed regardless of the current default drive assignment.

Other commands, called transient or external, require the loading and execution of program files (normally from the default diskette) for their processing. The default drive should contain the master diskette when these commands (**CHKDSK**, **COMP**, **DISKCOMP**, **DISKCOPY**, **FORMAT**, **MODE**, and **PRINT** are the most important ones) are entered; otherwise the transient commands normally must be prefaced by a drive designator for the non-default drive containing their corresponding program files. For example, if **B:** is the current default drive, the DOS file **CHKDSK.COM** is on the diskette in Drive **A:**, and the diskette to be checked is in Drive **B:**, then the DOS command **CHKDSK** would appear as:

B:\>A:CHKDSK<cr>

or **B:\>A:CHKDSK B:<cr>**

Most of the difficulties concerning when a drive or directory path designation is required can be alleviated by use of the **PATH** command (see Section 2.21.3) which allows the specification of a directory search path (a global current directory in a sense).

A.8 Redirection of Input and Output

DOS allows you to redirect input and output from standard input/output devices (i.e., the keyboard **CON:** and display screen **CON:**). This means that output from a DOS command or an executing program that would normally be sent to the screen can be routed to other devices or files instead. The output redirection symbol is >. To route directory information generated by the **DIR** command from screen to printer, for example, the command would be:

```
A:\>DIR B: > LPT1:<cr>
```

Output can be redirected to files in one of two ways, erase and write (>) or write append (>>) as in:

```
A:\>DIR B: > B:DIRFILE<cr>
A:\>DIR B: >> B:DIRFILE<cr>
```

Here, the directory information will be written starting at the beginning of file **B:DIRFILE** in the first case, and appended at the end of file **B:DIRFILE** in the second case.

Input can be redirected with the symbol <. For example, if a program from file **B:PROG.EXE** normally expects input from the keyboard and displays its results on the screen, input can be redirected from a file named **B:DATA** and output can be redirected to the line printer with the DOS command:

```
A:\>B:PROG < B:DATA > LPT1:<cr>
```

A.9 DOS Filters

DOS has three commands called filters that can read data from standard input devices (or other devices or files specified with the redirection facility), modify the data in

some way, and then write the output to standard output devices (or other devices or files specified with the redirection facility). The filters are (1) **MORE**, usually used to read input file lines and then to display them one full screen at a time, (2) **SORT**, usually used to read unsorted lines and to write the lines in sorted order, and (3) **FIND**, usually used to find and to display those lines in a stream of input lines (for example, from a file) that contain a specified character string.

A.10 Piping

DOS has a feature called piping that allows the screen output of one program or filter to be used directly as input by another program or filter. DOS creates temporary files on the root directory of the default drive to store the information that is being piped from one program to another. The piping symbol is |, as in:

A:\>DIR B: | FIND "DIR" | SORT > LPT1:<cr>

Here, the directory information for drive **B**, produced as output from the **DIR** command, will be piped to the **FIND** filter, which will locate those lines containing the characters **DIR**. The lines containing the characters **DIR** will be piped to the **SORT** filter, which will sort them into alphabetical order (in this case, starting with column 1). The output from the **SORT** filter will be redirected to the line printer.

A.11 DOS Commands

The following are the most important of the PC/DOS 3.1 commands, their prototypes and the corresponding actions taken by DOS. The entries **filename**, **filenam1**, **filenam2**, **sorcfile**, **textfile** and **trgtfile** appearing in the DOS prototype commands are DOS files following the naming conventions (here, the names are assumed to contain optional name extensions) described in Section A.2.

The file name may be preceded by a disk drive designator, (**A:** or **B:**), e.g., **B:MYFILE.103**, or by a directory path, e.g., **B:\SUBD1\SUBD2\MYFILE.103**. Drive names **d:**, **d1:** and **d2:** may be either **A:**, **B:** or **C:**. Square

brackets [ ] indicate that the entry is optional; if the information is omitted, then a default value will be assigned; e.g., if no drive name **d:** is specified, it is assumed to be **A:** whenever the DOS prompt **A:\>** appears and **B:** whenever the DOS prompt **B\>** appears. In the command prototypes and examples that follow, the prompt character **A:\>** and the carriage return **<cr>** are not shown.

Command	Prototype	DOS Command Type and Action

ATTRIB **ATTRIB [+R|-R][d:][filename]**

transient command

Used to set (or reset) the file **filename** to Read only status. If **+R** appears, then the file may be read but not written to (i.e., altered by other DOS commands). **-R** restores the file to read/write status. If neither **+R** nor **-R** appear, then the current read/write status of **filename** is displayed.

Example:

```
ATTRIB +R B:SACRED.FOR
ATTRIB -R B:MYFILE.BAS
ATTRIB B:MYFILE.BAS
```

BREAK **BREAK [ON|OFF]**

resident command

Under normal circumstances DOS checks for entry of the key combination **<Ctrl-Break>** only when it performs input or output operations (the default is **BREAK=OFF**). If you want to interrupt execution of a program that requests few input or output services from DOS, **BREAK** should be set to **ON**. If only **BREAK** appears, DOS displays the current **BREAK** status.

Example: BREAK ON
BREAK

CHDIR **CHDIR [d:] path**
or **CD [d:] path**

resident command

Changes the current directory on the specified drive **d:** (or the default drive if **d:** is not entered) to that specified by the DOS **path** (See Section 2.21.3).

Example: CD C:\FORTRAN\FORDATA
CHDIR \

CHKDSK **CHKDSK [d1:][filename][/F][/V]**

transient command

Checks the diskette or hard disk in Drive **d:** and displays information about the total storage space (bytes) used and still available, total number of files, etc. It also shows the total amount of main memory (RAM) and the unoccupied main memory space. If **/V** appears, then a list of all files on the drive will also be displayed.

If **filename** appears, then DOS will display the number of non-contiguous sectors that are occupied by the file (indicates the extent of fragmentation for the file contents).

If **/F** appears, then "lost clusters" (contents of sectors whose directory or allocation information is inconsistent) can be gathered together and written into files named FILE0000.CHK, FILE0001.CHK, etc.

Example: CHKDSK B:
CHKDSK B: /V

CLS **CLS**

resident command

Clears the screen.

Example: CLS

COMP **COMP [d1:]filenam1 [d2:]filenam2**

transient command

Compares **d1:filenam1** with **d2:filenam2** on a byte-by-byte basis. The program will note any unequal bytes found, and will stop comparison automatically after finding ten mismatches.

Example: COMP B:PROG.NEW B:PROG.OLD

COPY **COPY [/B][d1:]sorcfile [d2:][trgtfile][/V]**

resident command

Copies all bytes in source file **d1:sorcfile** into target file **d2:trgtfile**. If **d2:trgtfile** already exists, it will be erased and then overwritten. If **trgtfile** is omitted, then the target drive name must be different from the source drive name, and the target file will have the same name as the source file. If **d1:** and **d2:** are the same, then **trgtfile** cannot be the same as **sorcfile.**

The source "file" name may be **CON:** (the console) in which case characters entered at the

keyboard will be copied to the target file.

The target "file" may be **CON:** (console) in which case the source file contents will be copied to the monitor screen; to stop the scrolling of the display temporarily, enter **<Ctrl-S>** or **<Ctrl-NumLock>**; type any character in the typewriter section of the keyboard to resume.

The target "file" name may be **LPT1:** (the line printer), in which case the source file contents will be copied to the printer.

sorcfile can be a concatenated file (Section 2.15). If the files being concatenated are binary files, then the optional entry **/B** should appear.

The **/V** entry causes copy verification (see the **VERIFY** command) to be performed on the target file for the **COPY** command.

Examples:

```
COPY B:EXAMPLE.FOR B:ABC.1
COPY A:PROG B:
COPY B:PROB1.FOR B:PROB1.BAK
COPY B:PROG.SAV B:PROG /V
COPY B:MYFILE LPT1:
COPY B:*.*    (assumes A: is the
                 default drive)
COPY CON: B:OWNER.TXT
COPY B:PROG CON:
COPY /B B:F.BIN+B:G.BIN B:H.BIN
```

DATE **DATE [mm/dd/yy]**

resident command

Assigns a new value for the date where **mm** is a one or two digit month, **dd** is a one or two digit day, and **yy** is a two or four

digit year. If no assignment appears in the command, the system will request entry of the appropriate information.

Examples:
```
DATE 5/16/83
DATE
```

DIR **DIR [d:][filename][/P][/W]**

resident command

Displays on the monitor screen the file directory entry for **filename** (may be a generic file name) on Drive **d:**. If the file name is omitted, the directory entries for all files on the diskette in Drive **d:** are listed. **/P** causes the listing process for long directories to pause when the screen is full (to resume, press any key). **/W** causes the filenames only to be listed in horizontal format, five per line.

Examples:
```
DIR
DIR B:
DIR B:/P
DIR B:MYFILE.FOR
DIR B:*.FOR
DIR B:/W
DIR /W/P
```

DISKCOMP **DISKCOMP [d1:] [d2:]**

transient command

Compares the diskettes in the specified Drives **d1:** and **d2:** on a track-by-track basis, noting the track number and diskette side (1 or 2) in which any mismatches occur. **d1:** must be different than **d2:**.

Example: DISKCOMP A: B:

DISKCOPY **DISKCOPY [d1:] [d2:]**

transient command

Copies the complete contents of the source diskette (in Drive **d1:**) to the target diskette (in Drive **d2:**). If the target diskette is unformatted, the **DISKCOPY** command will format it while copying. Therefore, it is not necessary to format the target diskette before using the **DISKCOPY** command.

Example: DISKCOPY A: B:

ERASE **ERASE [d:]filename**
or **DEL** **[d:]filename**

resident command

Erases or Deletes the file **filename** from the file directory of the diskette in Drive **d:**; the file no longer exists so far as DOS is concerned.

Examples: ERASE B:MYFILE.FOR
ERASE B:*.BAK

FIND **FIND [/V][/C][/N] "string" [[d:]filename]**

transient command

The **FIND** filter causes DOS to search file **filename** on drive **d:** for all lines containing the character string **string** (the quotation marks must appear in the command) and displays them on the monitor screen (redirection can be specified). If **/V** appears, the lines not

containing **string** will be displayed. /C causes only a count of the number of lines containing **string** to be shown. /N causes the line number as well as the content of each line to be displayed.

Example:

```
FIND /N "DOS" B:PC.DOC
FIND "DIR" FILE > DFILE.LST
```

FORMAT **FORMAT d:[/S][/1][/8][/4][/V][/B]**

transient command

Formats (initializes) the contents of the diskette on the indicated Drive **d:** (the drive designator **d:** must appear). If /S is specified, the operating system files **IBMBIO.COM**, **IBMDOS.COM** (called "hidden" files) and **COMMAND.COM** will be copied to the new diskette from the default (master) diskette.

If /B appears, appropriate directory entries will be made for the hidden files **IBMBIO.COM** and **IBMDOS.COM**, but the files themselves will not be copied. This allows for distribution of application programs on diskettes to which the hidden files can be copied later using the **SYS** command.

If /V appears then the user can assign a volume name (up to eleven characters) to the diskette. Hard disks must also be formatted, but this is usually done just once.

If /1 appears, the diskette will be formatted in single-sided format (irrelevant if **d:** is a single-sided drive). If /8 appears, the diskette will be formatted with 8 sectors per track (compatible with DOS 1.0

and 1.1) rather than the usual 9 sectors per track (for DOS 2.0 and later). If **/4** appears (allowed only when formatting on a quad-density drive), the diskette will be formatted in double-density double-sided format with 9 sectors per track.

Examples:
```
FORMAT B:
FORMAT B: /V
FORMAT B: /S/V/1
```

GRAPHICS **GRAPHICS [/R]**

transient command

Allows for the subsequent printing of graphics screen images on a dot-matrix printer with entry of **<Ctrl-PrtSc>**. Normally, white on the screen will be printed as black. If **/R** appears, the white=black, black=white printing pattern is reversed to black=black and white=white.

Example:
```
GRAPHICS
GRAPHICS /R
```

LABEL **LABEL [d:][label]**

transient command

Changes the label of the volume (diskette or hard disk) on drive **d:** to **label**. If **label** is not specified, DOS will request interactive entry from the keyboard.

Example:
```
LABEL B: NEW NAME
LABEL
```

MKDIR **MKDIR [d:] path**

or (MD) **[d:] path**

resident command

Creates (makes) a new subdirectory on drive **d:** (default if not specified) having the directory path **path.** If **path** does not start with the root directory (\), the named directory will be a subdirectory of the current directory on drive **d:.**

Example: MD C: \USER\NEWSUBD
MD SUBCURNT.DIR

MODE **MODE [LPT1:][n][,m]**

transient command

If **LPT1:** is specified, this command sets the mode of operation for the line printer **LPT1:** to **n** characters per line (either **80** or **132**) and **m** lines per inch (either **6** or **8**). For the Freshman Engineering Computing Laboratories, the power-on options to the printer are **132** characters per line and **6** lines per inch. In other instances, the settings are likely to be **80** characters per line and **6** lines per inch.

If **LPT1:** is not specified, the command sets the display mode for the monitor screen to **n** characters per line (either **40** or **80**). **n** may also be **BW40**, **BW80**, **CO40**, **CO80**, in which case the color monitor will display in black and white (**BW**) or color (**CO**). In systems with both color and monochrome adapter/displays, **n** may be **MONO**, in which case the monochrome

monitor becomes the active display.

Examples:
```
MODE LPT1:132
MODE LPT1:132,8
MODE LPT1:80,6
MODE 40
MODE 80
```

MORE **MORE**

transient command

The **MORE** filter reads data lines from the standard input device (redirection is permitted) and displays one full screen at a time on the monitor (redirection is permitted). To continue with the next full screen, hit any key in the typewriter section of the keyboard.

Example:
```
MORE < B:MYFILE.TXT
```

PATH **PATH [d1:][path1][;[d2:][path2]] ...**

resident command

Sets the directory search path to include directories **d1:path1**, **d2:path2**, etc. If DOS is asked to process an executable file (i.e., a file having extensions **.COM**, **.EXE** or **.BAT**), which cannot be located in the current directory of the specified drive, each directory in the search path is searched in turn for the missing file. If **PATH** is entered with no parameters, the current search path will be displayed. If **PATH;** is entered, the current search path will be deleted.

Example:
```
PATH A:\;C:\FORT77;C:\MCACHE\LINEQ
```

```
PATH ;
PATH
```

PRINT **PRINT [d:][textfile] [/T][/C][/P]**

transient command

Queues one or more files **textfile** on drive **d:** for printing. Printing is performed in background mode while other DOS commands are entered and processed. **/T** terminates printing of all files in the printing queue; **/C** cancels only the named files from the queue; **/P** initiates printing (default case). By default, up to 10 files may be in the print queue at the same time.

Example:

```
PRINT B:FILE1 B:FILE2
PRINT /C B:FILE2    (removes
                B:FILE2 from queue)
```

PROMPT **PROMPT [prompt text]**

resident command

Allows you to change the form of the DOS prompt by entering appropriate characters in **prompt text.** [See a DOS manual for all possibilities.] In this text, we assume that the **PROMPT** command shown as the example below is included in the **AUTOEXEC.BAT** file. The **$P** and **$G** entries cause the current directory of the default drive and >, respectively, to be displayed as the DOS prompt, e.g., **C:\USER>**

Example:

```
PROMPT $P $G
```

RENAME **RENAME [d:]filenam1 filenam2**
or **REN** **[d:]filenam1 filenam2**

resident command

Renames the file on Drive **d:** currently named **filenam1** to **filenam2.**

Examples:
```
RENAME B:MYPROG.F NEWPROG.FOR
RENAME B:PR*.FOR B:PR*.WAT
```

RMDIR **RMDIR [d:] path**
or **RD** **[d:] path**

resident command

Removes a subdirectory from the hierarchical file structure. The subdirectory must be empty (have no files) and have no subdirectories of its own.

Example:
```
RD C:\USER\NEWSUBD
```

SORT **SORT [/R][/+n]**

transient command

The **SORT** filter reads input lines from a standard input device (redirection is allowed), sorts the lines into alphabetical or increasing numerical order (unless **/R** is specified, in which case sorting is performed in reverse order) starting in column **n** (column 1, if not specified). Sorted output is displayed on the monitor screen (redirection is allowed).

Example:
```
SORT < A:UN.FIL > A:SOR.FIL
SORT /R 7 < UNSORTED.FIL
```

SYS **SYS d:**

transient command

Copies the hidden files **IBMBIO.COM** and **IBMDOS.COM** to a diskette previously formatted using the **FORMAT** command with the **/B** parameter.

Example: SYS A:

TIME **TIME [hh:mm[:ss[:xx]]]**

resident command

Assigns a new value to the time where **hh** is a one or two digit hour, **mm** is a one or two digit minute, **ss** is a one or two digit second, and **xx** is a one or two digit hundredths of a second (only the **hh** entry need appear). If no assignment appears, the system will prompt for entry of the appropriate information.

Examples: TIME 10:12:3:57
TIME 14:37
TIME

TREE **TREE [d:][/F]**

transient command

If **/F** appears, the complete hierarchical file structure including all subdirectories and file names on drive **d:** (default if **d:** is not specified) are displayed. If **/F** does not appear, only the subdirectory names are displayed.

Example: TREE
TREE C:
TREE C: /F

TYPE **TYPE [d:] textfile**

resident command

Displays the contents of the file **textfile** on the monitor screen. To stop the display temporarily, enter **<Ctrl-S>** or **<Ctrl-NumLock>** (type any character in the typewriter section of the keyboard to resume).

Note: Press **<Ctrl-PrtSc>** before entering the **TYPE** command if you want to print the file contents while they are being displayed on the screen.

Example: TYPE B:MYFILE.FOR

VERIFY **VERIFY [ON|OFF]**

resident command

ON causes all disk writing operations (e.g., from the **COPY** command) to be "verified" by rereading, comparing with the original, etc. Under most circumstances such checking is unnecessary and time consuming. If no parameters are given, the current verify status is displayed. The default status is off.

Example: VERIFY ON
VERIFY

VER **VER**

resident command

Displays DOS version number.

Example: VER

VOL **VOL [d:]**

resident command

Displays volume name (label) of diskette or hard disk in Drive d:.

Example: VOL B: