DOCUMENT RESUME

ED 261 512 EC 180 581

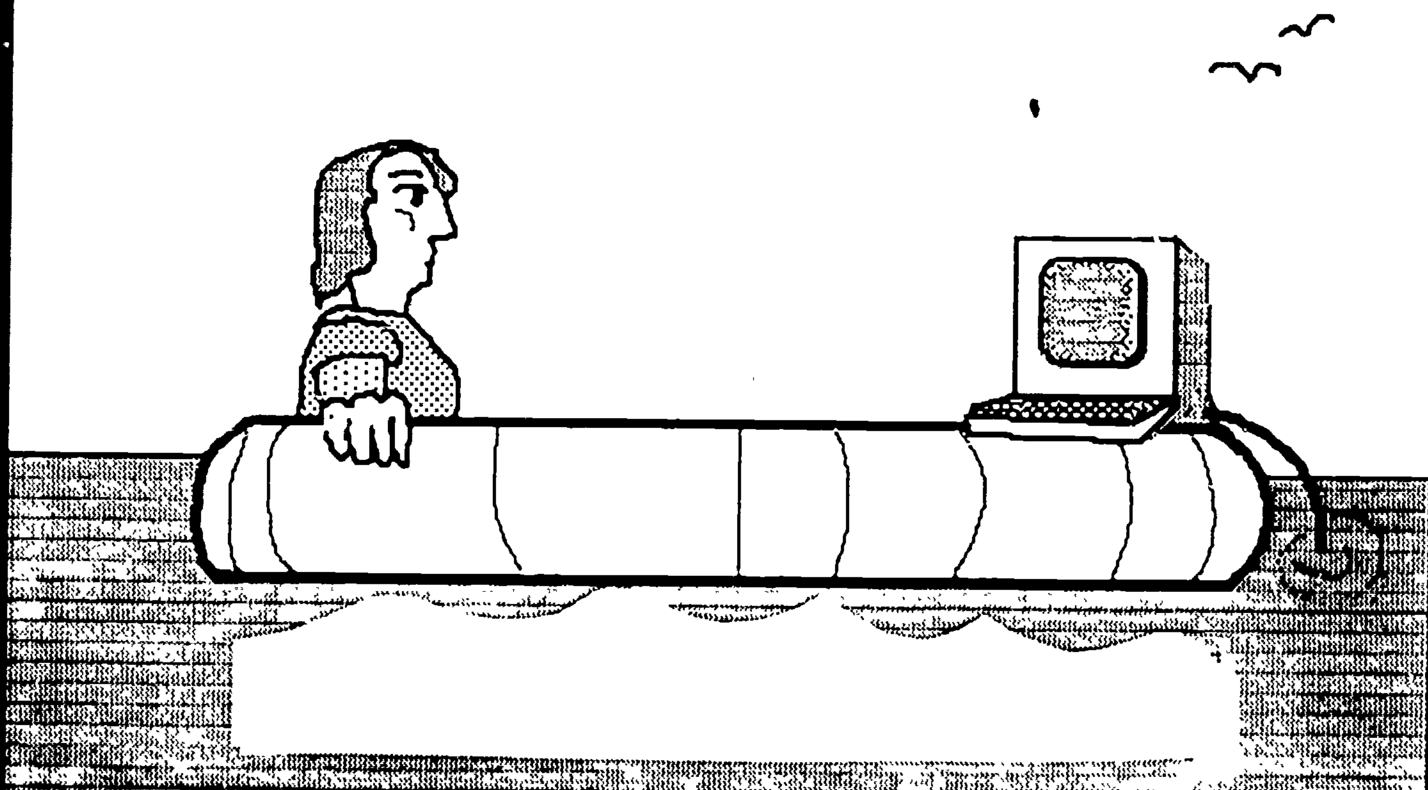
AUTHOR	Ankney, Barry
TITLE	Lost at Sea: Survival Manual on Microcomputers.
INSTITUTION	Illinois State Univ., Normal.
SPONS AGENCY	Department of Education, Washington, DC.
PUB DATE	[83]
GRANT	G008300313
NOTE	66p.
PUB TYPE	Guides - Classroom Use - Guides (For Teachers) (052)
EDRS PRICE	MF01/PC03 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Managed Instruction; Computer Oriented Programs; *Computer Software; *Disabilities; Elementary Secondary Education; *Microcomputers; *Special Education; Special Education Teachers

ABSTRACT

The information is offered to acquaint teachers with the scope and nature of microcomputer applications in special education. A brief history of computers traces the changes in the past 40 years. Information on basic computer operations covers such aspects as data storage, the nature of application programs, disk drives, and floppy disks. Step-by-step instructions are offered for using the Apple computer. Lessons are provided on the following topics: disk initialization, initialization of other diskettes from the initialized diskette, and disk operating system commands. Other topics addressed include basic programing, graphics, types of software (such as data base programs, word processing, and computer managed instruction programs), and software evaluation. A brief look at future trends and a list of selected programs of interest to special educators conclude the paper. (CL)

Lost at Sea:

Survival Manual on Microcomputers



Material prepared under U S D E Project No 023LH30056 by Barry Ankney, Project Associate, Illinois State University, Normal, IL 61761

INTRODUCTION

A quiet revolution seems to be occurring in American Education. The present revolution is the latest phase in the Computer Age. The Computer Age is generally said to have begun in the 1940's, and has continued to the present. From the 1940's to the present, the size of the computer has decreased, the cost of the computer has decreased, and the speed of the computer has increased. Beginning in the late 1970's, "micro"computers or personal computers entered the marketplace. Technology which was previously available to a select few persons became widely available. In the rush to sell microcomputers to homes and schools, companies often failed to provide adequate information regarding the use of the new microcomputer technology.

To intelligently utilize a microcomputer, one must more fully understand the capabilities of the microcomputer. One must also fully understand the present limitations of microcomputers. Simply purchasing a computer does not place one in the position of being computer literate. There are typically no directions which apply to the specific task one may wish to perform via the newly acquired microcomputer. There are certainly directions regarding how to turn on the machine and perform a variety of "cute" demonstrations. But the user is left to his or her own devices to understand the possible uses to which the computer may be applied.

In response to the need for training, most colleges and universities have taken giant strides toward offerring pre-service and in-service teachers courses in computer literacy. Many teachers are unable to commute to colleges and universities to enroll in computer literacy courses due to distance and time limitations. These materials are being offered in response to that need.

OVERVIEW

These materials are designed to:

1. Provide a frame of reference for the current uses and applications of microcomputer technology in special education.
2. Provide general microcomputer awareness and beginning microcomputer literacy.
3. Explore major types of application programs.
4. Provide a framework for evaluation of application programs.
5. Provide a list of resources for persons who wish to gain further information.

These materials <u>are not</u> designed to:

1. Train the reader to be a computer programmer.
2. Provide an exhaustive exploration of the hundreds of specialized application programs currently available for microcomputers.
3. Provide specific instructions on how to use specialized programs.

A BRIEF HISTORY OF COMPUTERS

A computer is defined as an Electronic Digital Computing device. The computer requires electricity to operate. As we shall see, the amount of electricity required depends upon the type of computer we are discussing. Secondly, the computer works with digital information. More about this later. Finally, the computer performs mathematical calculations, and some other types of calculations and comparisons. The computer can only operate on a series of switches that are either in the "on" or "off" position. Analyzing the status of these sequences of "ons" and "offs" is the main business of computers. Before we try to understand how this works, we will explore what changes computers have undergone in the past forty years.

The first computers were monsterous devices which filled large rooms; and often required several stories of a building. The first generation of computers contained vacuum tubes, not unlike the old vacuum tubes one can find in an antique radio or television. As you may know, the vacuum tubes created a great deal of heat due to the resistence of the filament. This heat buildup would often result in the tube "burning out". The first large scale computer, the ENIAC (Electronic Numerical Integrator and Computer) was completed in the mid 1940's. It contained approximately 18,000 vacuum tubes and was "programmed" by connecting various wires and setting about 6,000 switches in a prescribed manner. If one wished to calculate some other problem, the wiring would have to be changed, and the switches set to different positions. Considering the average life of a vacuum tube to be about 3,000 hours, much time was spent locating and replacing burned out tubes. The construction of the ENIAC was begun in 1943. The ENIAC remained in operation from 1946 through 1955.

In 1945, the development of another computer system, the EDVAC (Electronic Discrete Variable Automatic Computer), began. Although the EDVAC was not to become operational until 1951, it represented a major breakthrough in computing. A closely related machine, produced by a student of the inventor of the EDVAC, was the EDSAC (Electronic Delay Storage Automatic Calculator). Both the EDVAC and the EDSAC utilized a system of storing program instructions in electronic memory. With the ability to store instructions in electronic memory, the need to rewire and reset switches each time a new problem was to be calculated was overcome. What we today take for granted as a part of the computer system, that is, a computer program, was a major development in the late 1940's.

Two of the major developers of the ENIAC joined Remington-Rand Corporation in a venture to build a business computer. The resulting product, the UNIVAC (UNIVersal Automatic Computer) was the first commercially available computer. In 1951, a company which had specialized in electromechanical machines which used punched cards decided to enter the computer industry as an active force. International Business Machines Corporation (IBM) delivered its first computer system in late 1952. Building on their punched card accounting machine systems, IBM developed medium sized computers for use in business applications. In 1953, IBM announced their IBM 650 electronic computer. Remington-Rand and IBM were in fierce competition during the early 1950's. Remington-Rand held the lead in the computer business up until 1956, when their sales were surpassed by IBM sales.

The 1950's saw a tremendous growth in the demand for persons who could program computers. Instructions for the computer had to be written in a form that the computer could understand. This is called "machine language", and requires hundreds of thousands of steps to produce a "program". The search for a way to use symbolic notation in place of machine language was begun in the early 1950's. The first widely used "high level" computer language was written by a team of programmers from IBM. Their FORTRAN (FORmula TRANslation) language was introduced in 1957. By 1959, over 200 different programming languages had been developed. Nearly all of these languages were specifically developed for one or two identical computers, so a program which would function on one machine would not work on another machine. A meeting sponsored by the U.S. Government in 1959 resulted in the formation of a committee designed to write a computer language for business applications which would be machine-independent or work on any machine. The following year, the committee released COBOL (COmmon Business Oriented Language). COBOL remains one of the most widely used computer languages on large computer systems.

These first series of computers which utilized vacuum tubes for the computer circuitry were known as the "first generation" computers. The "second generation" of computers were started with the invention of the transistor at Bell Laboratories in 1947. Although several computers utilizing transistors in place of vacuum tubes were built in the early 1950's, it was not until 1958 that a computer system using transistors exclusively for internal circuitry was marketed. The use of transistors in place of vacuum tubes drastically reduced the size, the cost, and the maintainence of the average computer system. Consequently, more businesses and educational institutions could afford to purchase computers. The 1960's saw phenomenal growth in the number of computer systems. Some colleges began to train computer professionals, but most of the computer programmers were trained on the job. The glass enclosed, climate controlled computer room became the showplace of many colleges and industries.

The next major development occurred in 1964, when IBM announced the System/360 computer systems. The System/360 utilized a new technology called Solid Logic Technology. This technology involves using small chips to contain the controlling circuitry of the computer, rather than discrete pieces such as transistors and diodes. The announcement of the System/360 began the "third generation" of computers. The new technology was both more reliable and, more importantly, produced faster calculation times than the "second generation" of computers. While first generation computers could perform 2,500 calculations per second, the new third generation computers could perform 375,000 calculations per second.

The development of software for the new third generation computers involved design of both application programs and operating systems. The operating system software is the overall computer instruction set which tells the computer how to input data, how to process data, and how to interact with external storage devices, terminals, and printers. Application programs are specific task oriented programs designed to accomplish some task in an efficient manner, for example, a payroll program. The late 1960's and early 1970's saw considerable growth of a new software industry. Along with the growth of the industry came many horror stories about computer foul-ups. The general public came to distrust computers, and to question how they, the public, were to be protected from invasion of privacy by the new technology.

The year 1965 brought the announcement of the first "mini" computer. Digital Equipment Corporation (DEC) released the first "mini", and other companies were quick to follow. Some of the "mini's" were selling for as little as $5,000 for the central processing unit. As the size and cost of the computers decreased, more companies purchased computer systems. By 1970, there were about 100,000 computer systems in operation in the United States. Data entry devices were developed during the 1960's which permitted direct storage of information on magnetic tape or disk. During the 1970's, improvements in hardware and software designs made it possible to switch from a "batch" processing environment to a "transaction-oriented" mode of operation. In a batch system, all of the data is accumulated and entered into the system, then processed at one time. In the new transaction-oriented system, a single transaction could be processed, and the files would be updated. This capability was possible because of the technology to store large quantities of information in auxiliary storage devices from which the computer could directly read, process the transaction, and then write the updated information back to the storage device. Data communications, the ability to enter information at a terminal in one site, have the data transmitted over phone lines to a central computer, then have information transmitted back to the terminal, became a reality in the early 1970's.

The next major development in the computer industry was microelectronics. The technology was developed shortly after the invention of the transistor in 1947. IBM used the technology on the System/360, but advances in the technology occurred in the 1960's. The technology involves etching the circuits on thin wafers of silicon. In 1960, each wafer, or chip, approximately 1/4 inch square, could hold about 1,000 circuit elements. By 1970, the number of circuit elements on the same size chip numbered over 15,000. Today the same size chip may contain 70,000 or more circuit elements.

In 1969 the "microprocessor" was developed by INTEL Corporation. The "microprocessor" is a single chip which contains the arithmetic and logic processing units of a computer. The first microprocessor, contained on a chip smaller than a fingernail, contained almost as much power as the ENIAC, with its 18,000 vacuum tubes, and cost 1/16,000 as much to produce. The first "personal" computer was released in the mid 1970's. The microcomputer requires little electricity to operate. Most microcomputers operate on a 5 volt or 12 volt system, using a transformer to convert ordinary house current to the low voltage required by the computer. Some newer microcomputers operate via battery packs, making them portable. The technology of producing smaller and more powerful microprocessors seems to be continuing unabated. Today, many companies are producing small, powerful microcomputers. The leading manufacturers are Tandy Corporation (TRS 80), IBM (IBM PC and PC Junior), Commodore (Commodore 64), and Apple Computers (Apple II family, Macintosh, Apple III family, and Lisa). The use of microcomputers in educational settings has mushroomed over the past five years. Many schools are now requiring computer literacy as a requisite for high school graduation. The use of computers in special education settings has increased dramatically in the past five years. Before looking at some of the uses of microcomputers in special education settings, we will first explore how a computer processes and stores information.

BASIC COMPUTER OPERATION

A computer is a device which processes information stored in digital format. The computer has two basic means in which it can process information, by performing arithmetic calculations on the information, and by performing logical operations on the information. Arithmetic operations which are performed by the computer include addition, subtraction, multiplication, division, exponentiation, and a number of trignometric functions. Logical operations include comparisons of greater than, less than, equal to, not equal to, and a number of combined logical operations. All processing in the computer must by reduced to either a logical operation or an arithmetic operation. The proce. ing is accomplished by the central processing unit (CPU).

The data must be stored in an area adjacent to the CPU. This area is referred to as a register. The CPU can obtain information from the register, perform the prescribed calculation, and return the result to a register. Most microprocessors have several registers. For information to be stored prior to being sent to the register, and after being processed and returned to a register, some form of memory is required. In microcomputer systems, this memory is typically called random access memory (RAM) or read-write memory. This memory can store information which is to be processed by the CPU, and also stores results sent back from the CPU through a register. The read-write memory is capable of storing information only as long as the electricity flows to the computer. Once the electricity is turned off, the information is lost. There are other auxiliary means of making a copy of the contents of the read-write memory which is permanent, but more about those later.

The computer also needs a set of instructions to guide the input of information from read-write memory, the processing operation required of the CPU, and the output of the results to read-write memory. In addition, the computer needs a set of instructions to guide the linkage of the total system, including the monitor or TV, storage devices, and printer or other output devices. The instruction set for both the internal processing and the system functioning are usually stored in another type of memory, called read only memory (ROM). Read only memory is permanent memory which is etched into a silicon chip and connected to the CPU by direct circuits. Read only memory is not erased when the power to the computer is switched off. ROM remains intact and ready to send instructions to the computer as soon as the power is restored. The basic components of the computer itself are the CPU, and the two types of main computer storage (RAM and ROM). The computer is able to perform all of its operations with only these components.

In order for human beings to interact with and control the computer, several other components are required. There must be some way of entering new information and directing the computer to perform some specific task as it processes information. The typical computer has some form of keyboard, much like a typewriter keyboard, to accept input from the operator. Additionally, there must be some way for the computer to give instructions to the operator, and to prompt the operator for needed input of instructions or information. Typically, the personal computer uses some form of a monitor. This may be a television set hooked to the computer via a device which turns electrical impulses from the computer into radio frequency output (RF Modulator). Often, the monitor is a specially designed screen which may display color output, or only green or amber output.

To permanently store information which is to be processed by the computer, and to store the results of the processing in a permanent form, some auxiliary storage device is necessary. In effect, the auxiliary storage device makes a copy of the computers internal RAM and allows the information to be copied back into the RAM at some later time. If the results of the operation are to be prepared in written form, such as a report or the results of a student's lesson for that particular day, a printer will also be needed. A complete microcomputer system will typically include the processor and its main memory, the keyboard, a monitor or TV screen, a storage device, and a printer.

To understand how a computer processes information, it will be necessary to gain some basic understanding of how a computer stores data. Since a computer is an electronic digital computing device, let us explore briefly how digital information is stored, and how what we understand as numerals, letters, and punctuation marks are read by the computer. The computer senses information as a series of electrical impulses. The basic unit of storage in the main computer memory is a binary digit (bit). This single location in memory can be thought of as a switch, and can be either in the "on" position or the "off" position. The electronics of the computer hardware can sense if a particular switch is on or off. Obviously, a single switch with an on or off setting is not adequate to represent all of the letters, numerals, and other special characters we need to process. Consequently, the computer combines eight binary digits (bits) into what is called a byte. A byte is 8 bits. Using a standardized coding system, the position of the eight switches in an on or off pattern is sufficient to represent all of the letters of the alphabet (upper and lower case have a unique pattern), the numerals, and specially used characters that control input/output operations and commands to external devices. We will use the ASCII Code for illustrative purposes. ASCII stands for the American Standard Code for Information Interchange. A few examples may be instructive.

A numeral 1 represents an "on" position, while a numeral 0 represents an "off" position. The place values are written above each binary digit. The sequence of "ons" and "offs" necessary to represent several letters and other characters follows:

CHARACTER	128	64	32	16	8	4	2	1
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
1	0	0	1	1	0	0	0	1
2	0	0	1	1	0	0	1	0
?	0	0	1	1	1	1	1	1
a	0	1	0	1	1	1	1	1

In the above table, one can see that the letter A is represented by an ASCII code of 65 (the 64 place is on and the 1 place is on - 64 + 1 = 65). Similarly, the numeral 1 corresponds to an ASCII code of 49 (32 + 16 + 1 = 49)

In this manner, all characters can be represented by a series of eight switches, each of which is either on or off. Note that each character requires one byte of memory in which to be stored. Computers measure their internal memory in Kilo-bytes (approximately 1,000 bytes = one Kilo-byte - to be precise, one Kilo-byte = 1,024 bytes). People often refer to Kilobytes simply as K, so a 48K computer can store about 48,000 characters in read-write memory (a 48K computer has 49,152 bytes of RAM). The more RAM, the more characters which can be stored in main memory. The more characters you can store in main memory, the faster the microprocessor can process the information. Most personal computers today have a minimum of 64K memory.

As mentioned earlier, the CPU requires an instruction set in ROM in order to be told how to process data. The set of instructions which control the operation of the microprocessor is usually called the system monitor. This set of instructions is a computer program which resides permanently in the computer, and is usually written in a format that the microprocessor can directly interpret. In other words, the system monitor program is written in a binary format, commonly called machine language. Many microcomputers convert the raw binary code into a hexidecimal format. The hexidecimal number system has a base of 16. Remember that the binary format has a ones place, a twos place, a fours place, and an eights place. These four bits are one half a byte, often called a nibble. If we split a byte into two halves, or two nibbles, we can represent numerals in the range of zero through fifteen in one nibble. The hexidecimal number system is also designed to represent numbers from zero through fifteen, although the hexidecimal number system counts as follows: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. To return to our example of the representation of the letter A in binary, A was:

0 1 0 0 0 0 0 1		in binary
0 1 0 0	0 0 0 1	in split-binary
4	1	in hexidecimal

The hexidecimal 41 is much easier to represent in code than the binary 01000001. Both numbers are equal to the decimal 65, which is the ASCII code for the letter A. Machine code instructions are written in hexidecimal code and tell the computer how to function. Don't worry if you dont understand all of this, the information is presented for completeness, but it is not necessary for you to understand how this all works to program the computer. The reason you do not need to understand this is that all microcomputers also understand a different computer language, one that is much more meaningful to humans. The most popular language in use in microcomputers is BASIC, which stands for Beginners All-purpose Symbolic Instruction Code. Humans can, farily easily learn to use BASIC commands to control the computer. The computer has built into it (also in ROM) yet another program called an <u>Interpreter</u>, which converts the English-like commands in BASIC into the jibberish of machine code. An example:

The BASIC instruction, LET A = 19, Stores the value 19 in a memory location named A.

The machine code to accomplish the same task is:

```
A9 13 8D 25 03
```

Remember, you do not have to do this conversion yourself, the BASIC Interpreter does this for you. You only need to remember that when you issue a command in BASIC, the Interpreter converts the command into a form which the CPU can understand. The computer contains both the System Monitor and the BASIC Interpreter in Read Only Memory. There is one more program stored in Read Only Memory in some computers, that is the Operating System, the set of instructions that tell the computer how to accept input from the keyboard, how to store information on auxiliary devices, how to display information to the monitor or TV, and how to send information to a printer. Some computers store this program in read-write memory, rather than in ROM. There are then three programs available in nearly every computer that aid humans in interacting with the microprocessor, which "speaks" only binary code. Thank you, computer engineers, we would be lost without you!

Now that we have covered the basics of what goes on inside the computer, let's move on to some more practical information about the operation of the computer system. Remember that the computer needs specific instructions to perform different tasks. The instructions to have the computer present a drill and practice exercise in basic addition facts would be different from the instructions needed to have the computer present a graphical representation of the human heart. These different instructions are called application programs. They are typically written in BASIC, and are interpreted into machine code by the BASIC Interpreter. Programs come in several forms. They can be found in written form in magazines and books. In written format, they must be typed into the computer (stored in RAM) before they can be executed. This is a tedious job. You would not wish to type in a thirteen page tutorial program every time you wanted a student to use the program. Consequently, the storage devices we mentioned earlier in talking about permanently storing information can also be used to store programs. The auxiliary storage devices for microcomputers are of two main types.

Cassette recorders are the least expensive device to permanently store data and programs. Since information is stored sequentially on a cassette tape, information which is at the end of the tape is difficult to access, since the entire tape must be read first.

An alternative device is known as the disk drive, or floppy diskette drive. The disk drive and its accompanying floppy disk can store more information, and the information can be accessed directly, therefore, it can be retrieved more quickly. Most computer systems today are using some form of disk drive to store and retrieve data and programs.

The computer has, as part of its operating system, the ability to interact with the permanent storage devices to store and retrieve programs and data. Recall that the operating system is usually stored in Read Only Memory (ROM). Some computer systems utilize a system in which part of the operating system is stored in ROM, while the remainder of the operating system is stored on the permanent storage device, and is copied into part of the Read-Write Memory (RAM). Apple and IBM are among the computers which utilize this type of system. Since the typical computer system uses a floppy disk drive as the storage device, an understanding of the floppy disk is essential to computer usage.

If you have a floppy disk available, you may wish to look at it as it is described. A floppy disk, or floppy diskette, is a piece of mylar plastic which is cut in a circular shape. The mylar is coated with a thin layer of ferrous material, which can be magnetized. The disk is housed in a square plastic jacket, which is lined with a teflon type material, to allow the circular disk to spin inside the jacket. There is a large hole in the center of the disk, which is held by a special cone inside the disk drive. When the cone is rotated, the disk also rotates inside the plastic jacket. A small oval hole in the jacket allows the disk drive read-write head to scan the surface of the rotating disk. The read-write head can read the disk much in the same way a phonograph needle reads a record. The read-write head can also "write" information onto the surface of the disk, by arranging the ferrous particles is a certain pattern, magnetically.

Since both cassette tapes and floppy disks are designed to store information in magnetic form, on soft plastic, a few words of caution are in order. Floppy disks and casette tapes <u>should not</u> be exposed to magnetic interference. Do not place them on the top of the television or computer monitor, since the electromagnetic field surrounding the TV or monitor will surely destroy the media. Also, avoid magnets, clock radios, and any other appliance which has a small electric motor, and consequently a magnetic field surrounding the appliance. Extremes of heat and cold should also be avoided, with a range of 50 to 120 degrees Farenheit recommended for storage and operating temperatures. Care must be taken not to touch the surface of the magnetic material of the disk, since the natural oils in the skin can easily cause errors when the disk drive head attempts to read the disk surface. For the floppy disks, do not write directly on the disk label with a pencil or ball point pen. If you are to write on the disk label, be sure to use a felt tipped pen. Care in handling is important, as folds, marks made by paper clips, <u>any</u> moisture or dust, and excessive exposure to direct sunlight will destroy the disk surface. Diskettes should always be stored in the protective envelope when not in use. The most delicate part of the computer system is the magnetic material on which the programs and data are stored.

The floppy diskette can store programs and/or information. A floppy diskette which contains a specific program is, not suprisingly, referred to as a "program disk". A disk which stores information or data resulting from a program is called a "data disk". Most program disks are designed so that the user only need place the program disk in the disk drive and turn on the power to the computer. The program automatically takes over and begins operation. This type of program which runs automatically when you turn on the power is referred to as a "turnkey" program, since when you "turn the key" (turn on the power switch), the program runs.

Up to this point in our discussion, the information has been applicable to a variety of microcomputers. At this point, it is necessary to become very specific, in order to instruct on the actual use of a computer. The computer we will be discussing from this point forward will be the Apple][series microcomputer (Apple][, Apple][+, Apple][e, and Apple][c). The reason for focusing our attention on the Apple is twofold. First, there are more application programs written for the Apple computer for use with the handicapped than for any other computer available. Second, the Apple has been available for a longer period of time, and many schools have access to Apple computers.

We will now begin to look specifically at how to use the Apple computer, and later, we will look at the types of programs available for the Apple which are applicable to special education students.

Apple Computers: An Introduction

The Apple][series microcomputer was introduced as an expandable, affordable microcomputer. The Apple idea was to give the user options from which he/she could select, while allowing expansion at any time in the future. For our purposes, we need not discuss all of the configurations which are possible. We will assume that the specific Apple we are looking at has a minimum of 48 Kilobytes of RAM. If you happen to have a pre-1980 model, it is possible that you do not have 48K. If you are not certain, ask your local dealer. Also, if you have an Apple][e or][c, please keep the CAPS LOCK key in the engaged (down) position throughout our practice exercises.

While the original Apple][computer contained an INTEGER BASIC Interpreter, most persons have upgraded to an APPLESOFT BASIC Interpreter. The easiest way to see whether your Apple has the APPLESOFT Language is to do the following. Turn on the power to the TV or monitor. Turn on the power switch on the left rear corner of the Apple. The Apple should beep. If a disk drive is attached, the drive will make some noises (perfectly normal), and the red light on the front, left side of the drive should light. The words APPLE][should appear at the top of the screen. At this point, you should press the key marked "CONTROL", and while depressing that key, also press the key marked "RESET". The Apple will beep again, and something should appear on the screen. If the screen displays a reverse square bracket, like this], and a blinking square, you have APPLESOFT BASIC in your computer. If, instead, you see an asterisk, followed by the blinking square, or a greater than sign, >, followed by the blinking square, you probably do not have APPLESOFT BASIC. You should now turn off the power switch, in either case.

If you have the reverse square bracket (]), your computer will be prepared to execute programs in APPLESOFT BASIC everytime the computer is switched on. If you do not have the reverse square bracket (]), all is not lost. You may have what is called a language card, that is a special plug in card that allows your Apple to use other computer languages, those not contained in ROM. You may also have an APPLESOFT ROM CARD, which would also give you APPLESOFT BASIC by throwing a switch on the back of your computer. If you are unsure about having a Language Card, or an APPLESOFT ROM CARD, you should talk to your dealer, or someone who is more familiar with your computer.

Assuming that you have APPLESOFT in ROM, let's take a look at what else you have available when you first turn on the computer. If you have a DISK II disk drive, and the DOS SYSTEM MASTER diskette, you also have available the Apple Disk Operating System (DOS). Of course, the machine itself contains the system monitor (machine language).

We will procede in a step by step fashion to look at how the computer gets started when the power is switched on. Please read through this section, then sit in front of your Apple and repeat the steps as described. First, find the DOS SYSTEM MASTER disk. Insert the System Master in the disk drive (in drive number one, if you have more than one), and close the drive door. The door will fit flush against the front of the drive when it is closed. When you are inserting the disk into the drive, you should hear a faint click, and the disk will seem to seat itself in the drive. Now that the disk is in the drive, turn on the switch to your TV or monitor. Next, reach behind the Apple, on the left rear, and turn on the power switch. The Apple should BEEP, and the light on the disk drive should come on. The screen should briefly say, at the top, APPLE][, and then some other words will appear on the screen. The drive light will remain on for some time. When all is finished, the screen should look like this (or something quite similar - the dates may be different, and the line, "BE SURE THE CAPS LOCK IS DOWN" may be missing on your display):

```
                    APPLE ][
               JANUARY 1, 1983
 COPYRIGHT APPLE COMPUTER, INC.  1980,1982
          BE SURE CAPS LOCK IS DOWN
```

This backward bracket is the APPLESOFT BASIC prompt.

Let's stop here for a minute and talk about what has just occurred. On all 48K computers, two significant things have happened. First, the Disk Operating System has been copied from the diskette into RAM. Secondly, a program on the System Master Disk has been loaded and executed. If your computer has 64K memory, one other thing has occurred. The INTEGER BASIC language has been copied into the upper 16K of RAM. How could all of that happen simply by inserting a disk and turning on a power switch? Well, that all occurs because the Apple is designed to be <u>autostarting</u>. What that means is that every time the computer is turned on, the computer has a built in program (in ROM) which tells it to look for a disk drive. If it finds a disk drive, it turns it on, and loads a copy of the Disk Operating System from the disk to the Apple memory (RAM). Once DOS is loaded, the disk drive searches for the HELLO program on the disk, and loads a copy of that program into memory, again, RAM. The program, once loaded begins to execute. Part of what that particular HELLO program does is to search for a language card in the Apple, and if it finds on, to load a copy of INTEGER BASIC language into the Apple.

In much the same way, the Apple will load the DOS and the HELLO program frcm any disk. The program on the disk, in most cases, will begin to operate immediately. Consequently, for about 90% of what most people do with computers, that is all they need to know. Most persons can operate the programs on the Apple simply by inserting a disk, switching on the power, and following the directions on the screen. It is for the remaining 10% of the time you use the computer that you will need to know more. If you are certain at this point that all you care to know is how to get the Apple operating, than you can skip ahead to the section on types of software. You should take just one intermediate stop along the way. If you will turn to LESSON ONE and LESSON TWO, and complete those two simple exercises, you will then have formatted diskettes to use with your own application programs.

For those of you who want to learn just a <u>bit</u> more, here we go.

Disk Format

On the Apple][, the diskettes are formatted in a way that the surface of the disk is divided into 35 concentric tracks. Think of the tracks as circles of decreasing size starting near the outside edge of the disk, and proceeding toward the center of the disk. The tracks are numbered 0 through 34, with track zero closest to the outside edge of the disk. The <u>initializing</u> process formats the disk in this manner. Also, during initialization, the <u>D</u>isk <u>O</u>perating <u>S</u>ystem is copied onto the disk on tracks 0, 1, and 2. At the same time, Track 17, about the center of the disk, has written to it the diskette directory. The directory lists the names of the files contained on the disk.

Each track is further divided into <u>sectors</u>. The number of sectors per track is 16 for DOS Version 3.3, and 13 for DOS Version 3.2. Most newer disks are initialized with DOS Version 3.3. For DOS 3.3, there are a total of 560 sectors per disk. DOS and the directory track use 64 sectors, leaving 496 sectors available for use. Each sector can store 256 bytes of information (in other words, 256 characters) per sector. The disk can store up to 143,360 bytes or 143K of information. As you can see, each disk can store over twice as much information as the Apple memory can hold at any one time.

Now, if you have been following along with the lesson, your Apple should have the following screen display (or a display which is quite similar):

```
                         APPLE ][
                     JANUARY 1, 1983
       COPYRIGHT APPLE COMPUTER, INC.  1980,1982
                 BE SURE CAPS LOCK IS DOWN
]    This blinking rectangle is called a cursor.
     This backward bracket is the APPLESOFT BASIC prompt.
```

If your display looks like this, turn the page to lesson number one, read the first two paragraphs, then skip to step number seven (7). If you have turned off the Apple and/or your display differs from that above, turn to lesson number one and proceed through the lesson step by step.

LESSON ONE - DISK INITIALIZATION

A blank diskette cannot store information. Prior to use, a diskette must be initialized. Initialization is the process of writing the DOS (Disk Operating System) portion of the diskette. This Process involves mainly tracks 00 through 02 hexidecimal (that's 0 through 2 for those of us who speak decimal).

To initialize a diskette you should follow the directions given here step by step. Once you have learned more about Apple DOS, we will discuss other ways to initialize a diskette. For now, please follow the directions given here.

1. Turn on the Monitor or TV.
2. Insert the System Master Diskette in Drive # 1.
3. Close the drive door.
4. Turn on the Apple II (IIe) power switch.
5. The Apple will beep at you (This is perfectly normal; after all, you would expect a computer to beep and flash lights, wouldn't you ?).
6. The little red light on the Disk Drive lights. The Disk Drive makes noises like it has it is coming apart (again, perfectly normal). Finally, the drive light goes off, the disk stops the noise making, and the screen displays the folowing:

```
APPLE ][
JANUARY 1, 1983
COPYRIGHT APPLE COMPUTER, INC. 1980,1982
BE SURE CAPS LOCK IS DOWN
]
```

] This blinking rectangle is called a cursor.
This backward bracket is the APPLESOFT BASIC prompt.

7. Type the following: (Note: After typing each line press the return key.)

```
NEW
100 HOME
120 PRINT "HELLO PROGRAM"
130 PRINT
140 PRINT "THIS DISK BELONGS TO (YOUR NAME)"
150 PRINT CHR$(4);"CATALOG"
```

8. You have just entered your first computer program. To see a listing of your program, type the following:

```
LIST
(DID YOU REMEMBER TO PRESS RETURN ?)
```

9. At this point, Apple DOS is residing in the RAM memory along with your program that you entered above in number 7.

10. Now, the initialization of the blank disk gets underway:
    A. Remove the System Master Disk form Drive # 1 (please remember to place it in the paper sleeve).
    B. Insert your blank diskette in Drive # 1 (remember to close the drive door).
    C. Type the following: INIT HELLO
    D. When you press the return key, the Disk Drive # 1 activates.
    E. The Drive whirs and clunks for about two minutes. During this time, the program you wrote above is being saved onto the diskette in Drive # 1. More importantly, the Disk Operating System (DOS 3.3) is being written on the diskette in Drive # 1, and the diskette is being formatted into 35 tracks with sixteen sectors each.

11. Now wait until the blinking cursor returns to the screen.

12. To check the results of your work, turn off the power switch on the Apple. Wait a few seconds. Turn on the power switch.

13. After the appropriate Disk noises, the screen will look something like this:

```
HELLO PROGRAM
THIS DISK BELONGS TO <YOUR NAME> DISK VOLUME 254
 A 002 HELLO
]
```

Your diskette is now initialized. An initialized diskette can be used to store programs and other information generated by the computer. I suggest you place a label on the initialized diskette, indicating in some way that the diskette is initialized.

Now that you have completed LESSON ONE, let's check your understanding of this INITIALIZATION PROCESS by completing a brief self-test.

1. The initialization process:
   a. writes DOS to tracks \ , 1, and 2 .
   b. writes a directory an track 17.
   c. formats the disk into 35 tracks with 16 sectors per track.
   d. all of the above.

2. The command to initialize a disk is:
   a. FORMAT HELLO.
   b. INITIALIZE HELLO.
   c. INIT HELLO.
   d. none of the above.

3. Prior to being initialized, the disk:
   a. is ready to accept information.
   b. has 35 tracks.
   c. has a total of 560 sectors.
   d. is blank, and not usable by the Apple.

4. Before typing the command, INIT HELLO:
   a. you must insert the SYSTEM MASTER Disk and turn on the power to get a copy of DOS in the Apple memory,
   b. insert a new disk into drive one.
   c. both a and b.
   d. none of the above.

5. DOS Version 3.3:
   a. will divide each track into 13 sectors.
   b. will divide each sector into 16 tracks.
   c. will divide each track into 16 sectors.
   d. will divide each sector into 13 tracks.

6. To check the results of the INITIALIZATION process:
   a. you should turn the power off and remove your diskette.
   b. you should turn the power off, wait a few seconds, and then turn the power back on again.
   c. you should see the results of the program you have written as the HELLO program.
   d. first b, then c.

Your answers to the above exercise should be:

1. d
2. c
3. d
4. c
5. c
6. d

If you missed more than one of these questions, you should review the section on DOS Format before you proceed.

If you have finished LESSON ONE, and understand the process of initializing a disk, you are ready to learn a shortcut method of disk initialization. By using this method, it will not be necessary for you to use the SYSTEM MASTER Disk. Additionally, you will not have to type in the HELLO program, as you did in LESSON ONE.

Using this shortcut method, you can quickly initialize a number of diskettes. Each diskette will have the same HELLO program as your first initialized disk which you produced in LESSON ONE. Please turn the page now and complete LESSON TWO.

LESSON TWO
Initializing Other Diskettes from your Initialized Diskette

You may wish to initialize several diskettes at some point, so they will be ready to accept information when you need them.

Initializing a diskette is quite easy, once you have one initialized diskette. You may follow the procedure outlined in lesson one, but there is a much easier method. We will now use this shortcut method to initialize another diskette.

1. Insert your initialized diskette in drive one.
2. Turn on the Apple][(][e) power switch.
3. When the drive light goes out and your program appears on the screen, remove your diskette and insert a new diskette in the drive.
4. Type INIT HELLO.
5. When the drive light goes out, remove the newly initialized diskette.
6. To initialize another diskette at this point, insert another blank diskette in drive one and repeat steps 4 and 5.

You should now have an understanding of how to initialize a diskette. Any time a program you use calls for an initialized diskette, you will need to have one on hand and ready. Consequently, it is always a good idea to have a number of initialized diskettes handy at all times. Some programs will allow you to initialize diskettes directly from the program. For example, a program which stores the results of student's work for a particular day or week, will require a "data disk" to store the results, since, many times, the program disk is too full to store additional information. Some of these programs will allow you to interrupt the operation of the program to initialize a diskette, while others will not. It is probably best practice to have a few initialized diskettes on hand, just to be prepared.

We talked earlier about the directory track on the diskette, and said that the directory track contained a list of all of the programs contained on the disk. We will now learn how to "read" the directory track of a diskette. When the directions refer to "your diskette", they are referring to the diskette you have initialized in LESSON ONE.

Lesson Three
Disk Operating System (DOS) Commands
(Part 1)

There is only one program saved on your diskette at this time. You entered the program in lesson one, and saved it to the diskette in the process of Initializing the diskette. Soon, other programs will be saved on this diskette. To help keep track of the programs that are contained on the diskette, the Apple maintains a list of the programs you have saved on any particular diskette. The list of programs is known as a Catalog. Your diskette will automatically show you your current catalog whenever your diskete is "booted" because the program you typed in LESSON ONE contains the command necessary to automatically catalog the diskette when booted. Not all programs will have this feature built in to them, so we will now look at how to examine the directory on other disks. We can look at your diskette, or another diskette catalog by entering a DOS Command from the keyboard.

Insert your diskette in drive one and turn on the power. After you get the Applesoft prompt ( ]) and the blinking cursor, type:

```
TEXT <RETURN>
HOME <RETURN>
CATALOG <RETURN>
```

Here's what you should see:

```
VOLUME 254
 A 002  HELLO

 ]
```

The Apple has displayed the CATALOG of your initialized diskette. Your diskette currently contains only one program, called HELLO.

You now know about two DOS Commands:

1. INIT - initialize a diskette
2. CATALOG - displays the directory of the diskette

We will now learn about two additional DOS Commands that are closely related to each other (LOCK and UNLOCK).

Now type:

1. LOCK HELLO <RETURN>
2. The disk drive activates briefly, the the Applesoft prompt an' cursor returns.
3. If you CATALOG your diskette again, you can see what the LOCK command

26

has accomplished.

4. You should see:

```
VOLUME 254
*A 002 HELLO
```

5. As you can see, the only difference in the CATALOG is the asterisk in front of the A. Any time you catalog a diskette and see the asterisk in front of a file, you know the file is locked.

Now, let's reverse the process and unlock the file.

1. Type UNLOCK HELLO <RETURN>
2. The disk activates again, the the prompt and cursor return.
3. CATALOG again.
4. You should see:

```
VOLUME 254
A 002 HELLO
]
```

5. The asterisk has disappeared, indicating the file is UNLOCKED.

Locking a file protects against you or someone using your diskette. Your file could still be unlocked and then deleted, but you would be protected against it happening accidentally. In other words, your student cannot look at you and say, "I don't know how that could have happened." Deleting a file takes a conscious effort on someone's part.

Lesson Four
DOS Commands
(Part 2)

At this time you have learned about the following Dos Commands:

1. INIT (FILENAME)
2. CATALOG
3. LOCK (FILENAME)
4. UNLOCK (FILENAME)

In this lesson you will learn about:

5. LOAD (FILENAME)
6. RUN (FILENAME)
7. SAVE (FILENAME)

In the next lesson you will learn about:

8. DELETE (FILENAME)
9. RENAME (FILENAME 1), (FILENAME 2)

Insert your initialized diskette and turn on the power. Now type:

TEXT <RETURN>
HOME <RETURN> (This will clear the display screen.)
NOW:

1. Type: LOAD HELLO <RETURN>
2. The disk activates, then the cursor returns to the screen.
3. Type: SAVE HELLO 1
4. The disk activates, then the cursor returns.
5. Type: SAVE HELLO 2
6. Again the disk activates, followed by the return of the cursor to the screen.
7. Type: SAVE HELLO 3
8. Disk activates, cursor returns.
9. Now type: CATALOG to see the results of the SAVE Command
10. You should see:

```
DISK VOLUME 254
A 002 HELLO
A 002 HELLO 1
A 002 HELLO 2
A 002 HELLO 3
```

Here's what has happened. When you start the Apple, it automatically loads and runs the HELLO program. The HELLO program is designed to display the CATALOG of your diskette. We first cleared the display screen with the commands; TEXT and HOME. Then we asked the computer to find the program called

HELLO on the diskette, and to LOAD that program into the computer's memory. Once a program is in memory, you can save it to the diskette simply by typing the DOS Command, SAVE (Followed by the file name). When you save a program to the disk, you do not affect the computer's memory at all. The program is copied to the diskette, computer's memory. In the exercise above, we saved the program originally called HELLO, three additional times, each time under a different name.

In the process of learning about the save command, you inadvertently learned about the LOAD Command. When you LOAD a program, you place a copy of the program into the computer's memory, but you do not execute (RUN) the program. The original program is still in the Apple's memory right now. You can look at the program by asking for a listing of the program (by typing LIST). Do that now. Now to begin the program execution, type RUN. If you type RUN without a filename after it, you are simply asking the computer to execute the program that is currently in the computer's memory. (Remember, if you type RUN (followed by a filename) you are using a Disk Operating System Command, which requests the Apple to LOAD <u>and</u> RUN a file from the disk specified by FILENAME.) When you type RUN, the screen should look just like it did when you originally "booted" the disk.

Now to demonstrate the DOS Command, RUN (FILENAME), type RUN HELLO. This time the disk drive activates. The DOS Command, RUN LOADS a copy of the file named HELLO, and begins execution of the file named HELLO.

Before you turn off the power, LOCK the HELLO program.

Lesson Five

DOS Commands
(Part 3)

There are two additional DOS Commands you need to learn at this time. At the end of this lesson you will be familiar with the following commands:

1. INIT (FILENAME)
2. CATALOG
3. LOCK (FILENAME)
4. UNLOCK (FILENAME)
5. LOAD (FILENAME)
6. RUN (FILENAME)
7. SAVE (FILENAME)
8. DELETE (FILENAME)
9. RENAME (FILENAME)

We have covered all of these commands, except the last two. We will learn about the RENAME Command first.

1. Boot your initialized disk.
2. Type: TEXT:HOME <RETURN>
3. Type: CATALOG
4. You should see the following if you have been following along with the lessons.

```
VOLUME 245
 *A 002 HELLO
  A 002 HELLO 1
  A 002 HELLO 2
  A 002 HELLO 3
```

5. Type: RENAME HELLO 1, NEW ONE
6. The disk will activate, then the cursor will return.
7. Type: CATALOG
8. You will see:

```
VOLUME 254
 *A 002 HELLO
  A 002 NEW ONE
  A 002 HELLO 2
  A 002 HELLO 3
```

As you can see, you can RENAME a file with the RENAME Command.

Now, let's try to RENAME a LOCKED FILE.

1. Type: RENAME HELLO, NEW HELLO PROGRAM
2. The disk activates, the Apple "beeps" and the message:
   FILE LOCKED appears.
3. You have just learned that a LOCKED file cannot be RENAMED.

Now, let's look at the DELETE Command.

1. Type: DELETE HELLO 2
2. CATALOG
3. You will see:

```
VOLUME 254
*A 002 HELLO
 A 002 NEW ONE
 A 002 HELLO 3
```

4. HELLO 2 has been erased from the diskette.
5. Now type: DELETE NEW ONE
   DELETE HELLO 3
6. CATALOG
7. You will see:

```
VOLUME 254
 *A 002 HELLO
```

8. NEW ONE and HELLO 3 have been completely and permanently erased from the disk.
9. Now type: DELETE HELLO
10. The Apple "beeps"; FILE LOCKED message is displayed.
11. You cannot DELETE a LOCKED file.

Let's take a moment to check on your understanding of the DOS Commands we have covered by taking a brief self test.

1. DOS stands for:
   a. Don't Open Slot on the disk drive.
   b. Dynamic Off-line Storage.
   c. Disk Operating System.
   d. none of the above.

2. The command, CATALOG:
   a. displays a list of catalogs on the screen.
   b. displays a list of file names included on the disk.
   c. displays a list of disk volumes you currently own.
   d. none of the above.

3. The command, DELETE, followed by a file name:
   a. will delete the directory from the diskette.
   b. will erase the program currently in memory.
   c. will erase all programs from your disk.
   d. will erase the program named after DELETE.

4. All DOS Commands except this one need a file name added after the command.
   a. CATALOG
   b. DELETE
   c. LOAD
   d. RUN

5. The Command RUN is a DOS Command <u>only</u> :
   a. when it is entered by itself.
   b. when it is followed by a file name.
   c. both a and b.
   d. none of the above.

6. You can change the name of a file by using the command:
   a. DELETE
   b. CATALOG
   c. SAVE
   d. RENAME

7. The LOAD Command:
   a. loads a copy of the program named into RAM.
   b. loads a copy of the program from memory to disk.
   c. erases the Apple memory.
   d. both b and c.

8. To initialize a disk:
   a. Load the System Master.
   b. Type a HELLO program.
   c. Type INIT HELLO.
   d. All of the above.

Your answers should have been:

1. c
2. b
3. d
4. a
5. b
6. d
7. a
8. d

We will now look at Apple filenames in more detail. The Apple will allow you to store four different types of files on a DOS diskette. These are:

CODE	Meaning
A	APPLESOFT BASIC PROGRAMS
B	BINARY IMAGE FILES (MACHINE LANGUAGE FILES)
I	INTEGER BASIC PROGRAMS
T	TEXT FILES

When the diskette directory is examined, by CATALOGing the disk, you see something like the following:

```
VOLUME 254
*A 002 HELLO
 B 150 DATA BASE PROGRAM
 T 003 DATA FILE.DB
*I 123 GAME PROGRAM
```

When we examine this, we can get a great deal of information about that particular disk. First, the disk has four files, one of each type, as designated by the letter codes. Secondly, two of the files are locked, and two are unlocked, as designated by the asterisks in front of the file code. Thirdly, by looking at the three digit numeral following the file code letter, we can see how many sectors each file uses on the disk. For example, the HELLO program takes only two sectors storage space on the disk, while the DATA BASE PROGRAM takes 150 sectors of storage space. If we wished, we could add the numbers, to determine the number of sectors in use on the disk. In this case, the four programs occupy a total of 276 sectors on the disk. Since we know that there are 496 sectors available for use, we could also determine that there are 220 "free" or unused sectors remaining on this disk. The CATALOG command also gives us the VOLUME NUMBER of the disk. Unless we set the volume number, it will always be 254, by default. To specify a different VOLUME NUMBER, we would have to do so at the time the disk is initialized, by typing the command:

```
INIT HELLO,V 234
```

This would assign the number 234 as the VOLUME NUMBER. The VOLUME NUMBER must be in the range of 1 through 254. Or, you can specify 0 (zero), if you want DOS to ignore the VOLUME NUMBER.

A few words about filenames themselves is also in order at this time. There are a few simple rules to follow in naming files. A filename must begin with a letter of the alphabet. A filename must be from one to thirty characters in length. Finally, a filename can contain any character you can type from the keyboard, with the exception of the comma. Some examples follow:

acceptable filenames	unacceptable filenames
HELLO-PROGRAM	1ST-HELLO PROGRAM
DB345.54	FILE NUMBER 1, PART 2
TEXT.FILE-1	ABCDEFGHIJKLMNOPQRSTUVWXYZ12345

The first unacceptable filename starts with a number, the second contains a comma, and the third is longer than thirty characters.

BASIC PROGRAMMING

A program is a set of instructions written in a language the computer can interpret which tell the computer what to do, and in what sequence to perform the requested operations. BASIC stand for Beginners All-purpose Symbolic Instruction Code. A BASIC Program, then, is a set of instructions written in BASIC. In our case, it is Apple's brand of BASIC, called APPLESOFT BASIC. We can program the computer in two ways. One way is to type each command, one at a time, and have the computer carry out each instruction. This is called immediate mode, since each command is executed immediately after it is entered. While this is not very efficient, it is useful for learning how some commands work. The second, and more frequently used method of programming the Apple is by programmed mode. In programmed mode, the computer stores all of the commands in the sequence you specify, and executes them one after the other when you instruct the computer to do so. The main advantage, other than speed of execution, is that you can store the program in memory, and use the DOS Command, SAVE, to store the program to disk. When a program is SAVED, there is no need to retype the instructions each time you wish to execute the program.

Let's take a look at some BASIC instructions, or Commands, in immediate mode first. If you will once again "boot" your system, we'll have you do some things as we go along. ("Booting" simply means inserting a disk containing DOS, like your diskette, and turning the power on. "Booting" gets its name from the idea that the computer can pick itself up by its bootstraps when the power is turned on. In other words, the computer is "smart enough" to find a disk drive and get the program on the diskette started, all by itself.) So, if you will "boot" your initialized disk, we will get started.

Once your disk has been booted, type the command, NEW, and press the return key. NEW is a command which erases the current program from the computer's memory. You must press the return key to "tell" the computer that you are finished with your command. The command, NEW, does not erase the copy of your program from the disk (remember, DELETE does that). NEW only erases the copy in the RAM memory. After typing NEW, there is no program in the computer's memory. To prove this, type the command, LIST, and press return. You do not see anything except the APPLESOFT prompt ( ]) and the cursor on the line below the word LIST, that you just typed. There is nothing to list, because you just erased the program in RAM.

To see if the copy of the program still exists on the disk, type CATALOG. You will still see your HELLO program listed in the directory. To get a copy of that HELLO program back into memory what must we do? Right, type LOAD HELLO. Now after the disk light goes off and the cursor returns, type LIST, and the program will be listed just as you typed it in back in LESSON ONE.

Type NEW again to erase the memory once more. You now know how to erase the computer's RAM and how to erase a file from the disk. Next let's learn how to erase the display screen of your monitor or TV. Type HOME, followed by return. The screen clears and the APPLESOFT prompt and cursor move to the upper left hand corner of the screen. HOME clears the screen.

To put some information on the screen, we can use the PRINT command. Type the following:

```
PRINT "MY NAME IS"
PRINT "(TYPE YOUR NAME HERE)"
PRINT "AND I CAN PROGRAM THE APPLE"
```

Please press the return key after each line. As you see, the computer does everything immediately. The message that we had hoped would look like this:

```
MY NAME IS
BARRY R. ANKNEY
AND I CAN PROGRAM THE APPLE
```

turned out like this instead:

```
PRINT "MY NAME IS"
MY NAME IS
PRINT "BARRY R. ANKNEY"
BARRY R. ANKNEY
PRINT "AND I CAN PROGRAM THE APPLE"
AND I CAN PROGRAM THE APPLE.
```

Not very good, is it. Immediate mode leaves much to be desired. You see though, that the Apple will print anything you tell it to print by placing the message in quotation marks following the command PRINT.

Since immediate mode is rather limiting, let's look at programmed mode, and try to improve the above message. First, we will have to understand how to get all three lines of instructions arranged in the proper order to display the message as we really wanted it in the first place. To do this we use <u>line numbers</u>. A line number can be any number from zero through 63999, in integers. That leaves the possibility of having 64,000 separate lines of instructions which are numbered in a way that the Apple knows the order in which to process the instructions. We will not be using all 64,000 line numbers. It is a good idea to get in the habit of numbering the lines of instructions in increments of tens. In that way, if you decide later to add some other instructions between steps in the program, there will be room to do so. The computer does not care what numbers you use (within the stated range of 0 - 63999). The computer takes the lowest numbered line number as a starting point, and executes the instructions in the order of ascending line numbers. No matter what the intervals are between the numbers, the computer simply goes in ascending order.

Now, back to our message. If we want it to look like:

```
MY NAME IS
BARRY R. ANKNEY
AND I CAN PROGRAM THE APPLE
```

We will have to arrange the instructions in this order:

```
10 PRINT "MY NAME IS"
20 PRINT "(TYPE YOUR NAME HERE)"
30 PRINT "AND I CAN PROGRAM THE APPLE"
```

When you have typed the above, type RUN, followed by return. The command RUN, when not followed by a filename, is a BASIC Command, which causes the program currently in memory to execute. Now, after RUNning the above program you should see:

```
MY NAME IS
BARRY R. ANKNEY
AND I CAN PROGRAM THE APPLE
```

We can also see part of the commands we entered toward the top of the screen (above our message). Let's see if we can put together what we know up to this point to display our message on the screen, with no other distracting information. Any ideas? How about the command HOME? If we were to add the command HOME to our program just before we PRINTed our message, do you think we could get rid of the clutter? Let's try it. To add this command, simply type:

```
5 HOME
```

right where the cursor currently is. DO NOT type NEW, since we do not want to erase the other three lines from the memory. After you type the above line, LIST your program. You should see:

```
 5 HOME
10 PRINT "MY NAME IS"
20 PRINT "(TYPE YOUR NAME HERE)"
30 PRINT "AND I CAN PROGRAM THE APPLE"
```

As you can see, the computer places the line in the proper sequence for you automatically. You can add a line of instructions at any time that you see the APPLESOFT prompt and the cursor. Just remember one thing, if you add a new line of instructions with the same line number of a line that already exists in your program, you will erase the old line and replace it with the new line of instructions. If you can't remember what numbers you have used, simply LIST and check. Now to check the results of adding line number 5, type RUN. (You will no longer be reminded to press return.) When you RUN the program, the screen is cleared prior to our message being displayed.

Now we will try to make the message look a little better by centering it in the screen. The Apple display screen will display 40 characters across, by 24 lines vertically. (The Apple with the optional 80 Column Display Card, but we will only deal with the standard configuration.) After issuing the command, HOME, the cursor is placed at position 0 horizontally, and 0 vertically. The next character will be printed in that HOME position unless we move the cursor prior to issuing the PRINT command. One way to accomplish this is to have leading,blank spaces in front of the message by pressing the space bar following the leading quotaton mark the number of spaces you wish the message to be moved to the right. For example, if we want 'MY NAME IS' centered on the screen, we could use this command:

```
10 PRINT "               MY NAME IS"
          (15 spaces).
```

We add fifteen spaces since there are a total of forty columns, and our message takes 10 spaces. Forty minus ten is thirty, and half of thirty is fifteen. Another way to accomplish this would be to reset the cursor directly with the command HTAB X, where X is the number of spaces you wish to indent before printing your message. To illustrate the above example,

```
10 HTAB 15 : PRINT "MY NAME IS"
```

would accomplish the same results. Note that we have separated the two commands, HTAB X, and PRINT with a colon. Any time you wish to use more than one command on the same line, simply separate the two commands with a colon. You can string together as many commands on the same line as you like, as long as you do not exceed a maximun of 255 characters per numbered line. For example, our enhanced program could be written on only one line, like this:

```
10 HOME : HTAB 15 : PRINT "MY NAME IS" : HTAB 12 :
    PRINT "BARRY R. ANKNEY" : HTAB 6 : PRINT "AND I
    CAN PROGRAM THE APPLE"
```

This one line program will clear the screen, then display the message on the screen, with each line centered on the screen. Each line is printed on a separate screen line, since the Apple "sees" a carriage return following each PRINT statement and its message enclosed in quotation marks. To stop the computer from reading the carriage return, you must insert a semicolon prior to the colon which separates commands on a single line, or before you press carriage return at the end of the line, since the Apple also reads the carriage return at the end of the line as an indication that the next message is to be printed to the next screen line. To return to our example, first as multiple line numbered program:

```
 5 HOME
 8 HTAB 8
10 PRINT "MY NAME IS ";
20 PRINT "BARRY R. ANKNEY"
25 HTAB 6
30 PRINT "AND I CAN PROGRAM THE APPLE"
```

This version would result in the following display:

```
        MY NAME IS BARRY R. ANKNEY
      AND I CAN PROGRAM THE APPLE
```

The print statement on line 10 does not send a carriage return, because we placed a semicolon at the end of the line, before pressing the carriage return key. The message on line 20 follows immediately after the message produced by line 10.

Now the same display, produced by a one line program.

```
10 HOME : HTAB 8 : PRINT "MY NAME IS ";: PRINT
    "BARRY R. ANKNEY" : HTAB 6 : PRINT "AND I CAN
    PROGRAM THE APPLE"
```

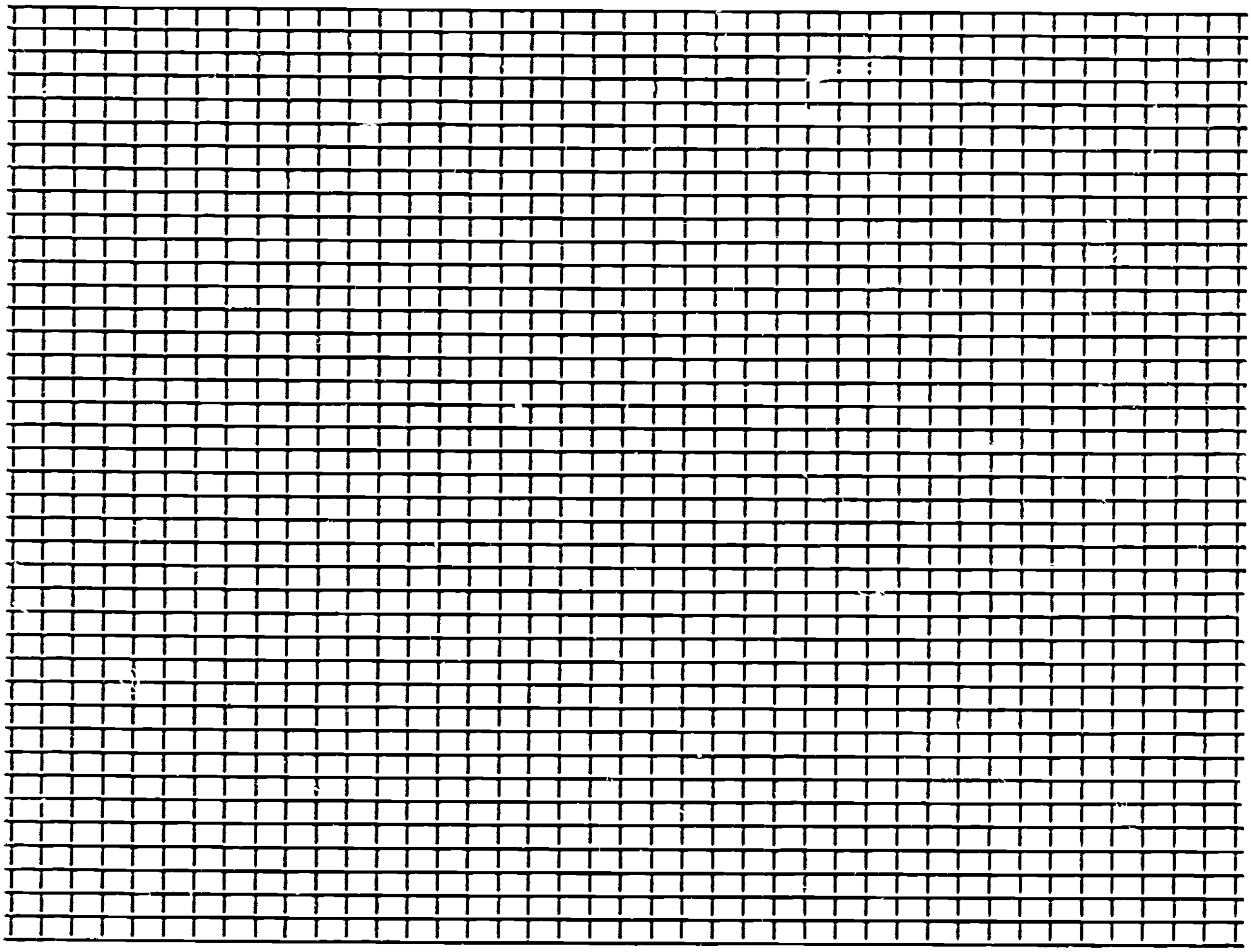
A LOOK AT GRAPHICS

We will leave the PRINT statement now and look at another capability built into APPLESOFT BASIC. The Apple has, in addition to the TEXT mode we were using to print our message to the screen, several graphics modes. The TEXT screen we used before has 40 columns by 24 lines. The LOW RESOLUTION GRAPHICS screen has 40 columns by up to 48 rows of graphics blocks (although we normally use only 40 rows for graphics, and leave four lines of text at the bottom of the display screen. The HIGH RESOULTION GRAPHICS screen has 280 columns by up to 224 rows of graphics dots (although we normally use only 192 rows for graphics, and leave four lines of text at the bottom of the display screen).

Lets look at LOW RESOLUTION (LORES) GRAPHICS here. LOW RESOLUTION GRAPHICS are popular in educational programs, because of the ease of programming LORES versus HIGH RESOLUTION Graphics. The LORES graphics screen has the 40 columns numbered from 0 to 39. The rows are also numbered 0 to 39 in the normal LORES display. We can look at the screen as a grid in which the 1600 blocks can each be painted in any one of sixteen colors. The block in the upper left is numbered 0,0 and the block in the lower right is numbered 39,39. (Always refer to the column number first then the row number.) Each block is actually somewhat rectangular in shape, with the horizontal side slightly longer than the vertical side. On the next page is a rough grid which will aide us in exploring LORES graphics.

40

LOW-RESOLUTION GRAPHICS PLOTTING GRID



LOW-RESOLUTION GRAPHICS COLORS

COLOR	NUMBER	COLOR	NUMBER
Black	0	Brown	8
Magenta	1	Orange	9
Dark Blue	2	Gray # 2	10
Purple	3	Pink	11
Dark Green	4	Light Green	12
Gray # 1	5	Yellow	13
Medium Blue	6	Aqua	14
Light Blue	7	White	15

The LORES graphics grid on page 39 shows 40 columns across the page, and 40 rows down the page. Also, the LORES colors are listed at the bottom of page 39 according to the color number. In order to place a pink block in the upper left hand corner of the LORES page, we need to tell the computer to switch to LORES graphics, and color the upper left rectangle pink. To call up the LORES graphics page, we use the command, GR. GR will clear the screen to black and give us 40 by 40 rectangles to color. To specify what rectangle we wish to color, we call for the rectangle by the column number first, followed by the row number. Remember, the columns run up and down the page and the rows run across the page. The color must be specified before we attempt to plot the point, otherwise, the color of the last plotted point will be used. If there were no previous points plotted, the point will be plotted in black. It becomes quite difficult to see a black point on a black background.

Again, first we call for the LORES screen, then we specify the color, then we use the column-row coordinates to specify which point we want to plot. The APPLESOFT commands to do this, in programmed mode, are:

```
10 GR
20 COLOR = 11
30 PLOT 0,0
```

Line 10 switches to LORES graphics display. Line 20 specifies the color pink. Line 30 specifies the block to be plotted.

We can plot points individually by specifying C and R in the statement, PLOT C,R - where C is the column number, and R is the row number of the block. There is also a built in way of plotting a line in either the horizontal <u>or</u> vertical direction. To plot a horizontal line we use the command, HLIN C1,C2 AT R, where C1 is the beginning column number, C2 is the ending column number, and R is the row on which you wish the line drawn. For example, to draw a line across the top of the screen, the command, HLIN 0,39 AT 0 is used. To draw a horizontal line across the bottom of the screen, the command, HLIN 0,39 AT 39 is used. In programmed mode:

```
10 GR
20 COLOR = 11
30 HLIN 0,39 AT 0
40 HLIN 0,39 AT 39
```

We can also draw vertical lines using the command, VLIN R1,R2 AT C, where R1 is the beginning row number, R2 is the ending row number, and C is the column number. This time we will draw a vertical line from the top, center of the screen, to the center of the screen. To do that we will use the command, VLIN 0,19 AT 19. The line starts at row 0 (the top of the screen) and ends at row 19 (the middle of the screen), and is drawn at column 19 (the middle column). Try your hand at this for a few minutes.

To change colors while you draw, simply issue the COLOR command with a new color number just prior to the point or line you wish to have plotted in a different color. For example, to draw the folowing lines in different colors, follow this kind of procedure:

```
10 GR
20 COLOR = 11
30 HLIN 0,39 AT 0
40 COLOR = 6
50 HLIN 0,39 AT 39
60 COLOR = 1
70 VLIN 0,19 AT 19
```

We will now be introduced to variables, and then use the variables to do some drawing on the LORES screen. Up to this point we have done all of our programming with <u>constants</u>. That is, we have used actual values for numerical quantities. It is also possible to use a <u>variable</u>. Think of a variable as a place in which we can store different values. Each variable will have a variable name associated with it. When we assign something to a variable, we place that something in the box with the proper variable name. There can only be one value associated with any variable at a given time, but we can change that value at any time in our program. Let's look at a specific example. We have a variable named, A. We assign the value 10 to the variable A. We do this with the BASIC statement:

```
A = 10
```

While the above expression is read as, "A equals 10" in algebra, in BASIC, it is read, "assign the value 10 to variable A". It is important to read the expression in this way, since we can perform arithmetic operations on variables. For example, the statement:

```
A = A + 1
```

is not a valid algebraic equation. If we read the above as, "assign the current value of variable A + one more to the variable A." In other words, "take whatever value is currently in box A, add one to that value, and assign the sum to variable A." If A held the value 10 prior to the above statement, A would hold the value of 11 after the above statement. We can also subtract, divide, multiply, and exponentiate the values in a variable with similar statements.

Now, we will discover the value of variables in improving the speed and performance of a program. We will first look at one way of plotting every point on the screen using constants. You should type the following program and then RUN it to see how it works.

```
 10 GR : COLOR = 3
 20 HLIN 0,39 AT 0
 30 HLIN 0,39 AT 1
 40 HLIN 0,39 AT 2
  .  .  .  .  .  .
  .  .  .  .  .  .     CONTINUE EACH LINE WITH ONE
  .  .  .  .  .  .       ADDED TO THE VALUE
  .  .  .  .  .  .        OF THE ROW NUMBER
  .  .  .  .  .  .
400 HLIN 0,39 AT 38
410 HLIN 0,39 AT 39
420 END
```

That is rather tedious, isn't it. When you run the program, the screen clears to black, and then very quickly is colored in one row at a time starting at the top row, and continuing to the bottom row. It serves our stated purpose in plotting every point on the screen using constants, but there must be a better way. If we add only two additional BASIC commands, GOTO (LINENUMBER), and IF (condition) THEN (statement). These commands work in this way. The GOTO command stops the Apple from using its built-in way of starting at the lowest numbered line number and proceeding in ascending order to the highest numbered line number, by causing the program to "branch" to the specified line number. Once the program has made the jump to the specified line number, the built-in way of proceeding sequentially takes over again. The IF - THEN statement is BASIC's way of performing logical calculations. IF the given condition is true, THEN the statement is performed. IF the statement is not true (false), THEN the statement following the THEN is ignored, and the computer goes on to the next line numbered instruction.

We will look at an example which does the same exact thing as our tedious example above. Please type in this program and then RUN the program after you are certain you have made no typographical errors.

```
10 GR : COLOR = 3
20 R = 0 : C1 = 0 : C2 = 39
30 HLIN C1,C2, AT R
40 R = R + 1
50 IF R < 40 THEN GOTO 30
```

```
60 END
```

Here is what the program does:

Line 10 sets LORES graphics, and specifies the color purple.
Line 20 sets the values of three variables; R, C1, and C2.
Line 30 draws a horizontal line from column 0 to column 39 at the row number determined by the current value of R.
Line 40 increments the value of R by one.
Line 50 checks to see if the current value of R is less than 40. If R < 40, the program branches to Line 30. If R = 40, the program ignores the GOTO statement, and goes to Line 60.
Line 60 ends the program.

We have set up what is called an "IF - THEN LOOP". The program loops back to perform some statement IF certain conditions are met. IF in Line 50, the value of R is 1, as it is the first time through the program, the condition, R < 40, is true. Since the condition is true, the statement following the THEN is executed. The program goes to Line 30, where a horizontal line is drawn from column 0 to column 39 at row R. Since R now has the value of 1, the line is drawn at row 1. Line 40 adds one to the value of R, making R now 2. Line 40 checks to see if R < 40 is still true. Since it is true, the statement following the THEN is again executed, sending the program back to Line 30, to plot another line. The program continues in this fashion until R is incremented to the value 40. At that time, the condition, R < 40, is false. Since the condition is false, the statement following the THEN is ignored, and the program continues to the next line number. At that line number, 60, the program ends.

Certainly the second approach is much faster than the first approach. There is one other way in which the same program can be written. This final approach utilizes a different kind of "loop". Rather than the "IF - THEN" loop, this last approach uses a "FOR - NEXT" loop. FOR - NEXT is a handy, built-in way for the computer to do automatic incrementing of a variable. It works like this:

```
10 GR : COLOR = 3
20 C1 = 0 : C2 = 39
30 FOR R = 0 TO 39
40 HLIN C1,C2 AT R
50 NEXT R
60 END
```

Here is how this loop works:

Line 10 again, sets LORES graphics and the color of our choice.
Line 20 sets the values of variables C1 and C2.

Line 30 starts the FOR - NEXT loop, by specifying the starting value of the variable, R, and also specifies the upper limits of the varialbe, R.

Line 40 draws the horizontal line at the row specified by R.

Line 50 increments R by one, and automatically causes a branch to the FOR statement in Line 30, where the value of R is checked to see if it remains within the specified limits, i.e., between 0 and 39. If R is within the limits, the program goes on to the next line and executes the statement, in this case, draws a line at row R. If R exceeds the limits, i.e., if R > 39, then the loop is ended, and the program branches to the next numbered line following the NEXT statement, in this case, Line 60.

Line 60 ends the program.

Here we have three ways of accomplishing our stated goal of coloring all 1600 blocks on the LORES screen. Each is slightly better than the one preceeding it. We have spent some time on LORES graphics, because many educational programs employ LORES graphics. If the reader is interested in developing graphics for use in his/her own programs, the grid on page 64 may be useful. The grid may be copied and used in the following manner. First draw your pictures on the grid in the colors you wish to use, then translate the colored grid into BASIC commands.

While we have just scratched the surface of BASIC programming, it is time we go on to other things. As mentioned in the overview, one cannot become a programmer by following these materials. We have explored enough of the basic BASIC to have a level of literacy sufficient for many teachers. If there is a need to explore BASIC programming further, and it is not possible to attend college classes, there are a number of books available which will take the interested reader further into BASIC programming. Some of these will be listed in the list of resources at the end of these materials.

Before we begin to look at types of software, test your understanding of BASIC by writing a program which will do the following:

1. Draw an outline around the LORES screen in the color white.
2. Draw a line in the color green from the upper left hand corner of the screen to the lower right hand corner of the screen (the line will actually look like steps).
3. Place a small rectangle, two blocks by two blocks on the sides, in the very center of the screen in the color of your choice.

If you are able to complete this assignment, you have come a long way toward being computer literate. A possible solution to this program is printed at the end of these materials. Try to solve the problem on your own, then check the listing at the end of these materials.

Types of Software

There are four general types of software (programs) which we will examine as having potential for use in special education. They are: Word Processing Programs, Data Base Programs, Computer Assisted Instruction Programs, and Computer Managed Instruction.

We will look at some characteristics of each major type of software, and some examples of each will be offered. The mention of these programs by name should not be construed as an endorsement of the program, but is offerred as a starting point in one's search for appropriate software for their computer.
When looking for software, perhaps the best way to judge whether the program will work for you, is to ask the dealer for the names of some persons who are currently using the program. Most dealers are happy to provide a potential customer with this information. There are many users who do not mind answering questions about their experiences with the programs. In addition, most dealers have a policy of allowing a person to try a program before buying the program. For a complicated program, sitting down at the dealership for a few minutes is not very useful in many cases, but the programs which are designed for classroom use can often be previewed in fifteen minutes, at least in enough detail to ascertain if the program meets the needs of the user. We will look at one way of formally evaluating software before purchase a little later. Right now, we will look at the types of software.

A Word Processing Program is a program which turns the computer into a word processor. A word processor has electronic storage of words that are typed, and adds the capability of moving, reformatting, rearranging, and deleting entire words, paragraphs, and pages of printed material. Most Word Processors also include powerful modules which allow the user to print the information to a printer in any number of formats. Word processors are time savers, if the person who is using the program is a good typist. If one does not type well, the word processor does not really save time. The primary advantage for teachers is the ability to store documents (lecture notes, class handouts, worksheets, tests, etc.) which are used frequently. There is no need to retype materials each semester. The teacher can edit and update any information and quickly produce another copy. A real advantage to a printer in concert with a word processor is its ability to produce crisp, error free ditto masters. Most printers which will accept single sheet paper can accept the ditto master inserted in place of the paper.

There are numerous Word Processing Programs on the market for the Apple. In choosing a Word Processor, one should look for the features which are best matched to the user. For example, a Word Processor which boasts the most features may be appropriate for heavy use in an office environment by persons specially trained in using all of the features of the program. This same program may be overwhelming for the occasional user, who would have much difficulty remembering the special commands associated with the program. The occasional user is usually better served by a less complex program, which may have fewer features. If there is not a need to do "fancy" things like automatically preparing an alphabetized index from a document, why buy a program with this capability? There is often an attitude that bigger and faster is better, but the expense, both in dollars, and in time to learn to use the program, may offset the benefits.

There are a number of low to medium priced word processing programs which will fit the needs of most teachers. The Word Processing Program which was used to prepare the present document is The Executive Secretary (TM). Another Word Processing program with which the writer has had significant experience is the Applewriter Program (TM) from Apple Computers, Inc. While this should not be construed as an endorsement of these programs, the writer does know many teachers who routinely find these programs helpful and adequate to meet their needs.

There are also a number of Word Processors available for student use. One of the more popular is The Bank Street Writer (TM). The Bank Street is easily learned by even young students (the writer has seen eight year old students of average ability use the program with little difficulty), but is also a useful program for adults.

The second major type of program we will explore is the Data Base Program. The data base program can be thought of as an electronic filing system. The system contains records in a file. A number of records can then be stored in a filing system, and can be rearranged, accessed, edited, and used to print a variety of reports. Using a file cabinet metaphor, each record can be thought of as a card. On each card, there are a number of different entries in standard fields (lines). Each card (record) is stored on a file (drawer) in a specified order. Rather than actual paper cards in a file drawer, the "cards" are stored in an electronic file. The major advantage of use of the Data Base Program is the ability to store routinely used information about students in a format that can easily be accessed. Once the system is established, records are easier to maintain on a Data Base Program system. The major disadvantage is the amount of time required to establish and initially enter the records onto the Data Base Program. There are also numerous Data Base Programs available. One of the features to look for in a Data Base Program is the ability to link the Data Base Program with your Word Processing Program. For management of information, such as that required of state and federal regulations stemming from P.L. 94-142, a Data Base Program can be a real time saver, especially when it will link to a Word Processing Program to prepare reports.

Some of the Data Base Programs that the writer has seen used effectively in special education management settings include the Information Master (TM), the Quick File II (TM) and the DB MASTER (TM) Data Base Programs. The former is toward the lower end of the price range, the second is toward the middle, and the later is toward the upper end of the price range. Generally, the features offerred are compatible with the prices. What was said earlier about talking with current users is also important for searching for a Data Base Program. Each of the above programs will link with different Word Processing Programs.

The greatest variety of programs will be found in the next category, Computer Assisted Instruction . In the field of special education, many programs have been developed for children with specific handicapping conditions. Programs are available to do everything from displaying large print to the screen for visually handicapped, to programs which utilize voice input for persons unable to use a standard keyboard. In addition, there are a number of adaptive devices which are used as component parts of the computer to allow alternative input through switches of various types. Speech synthesizers are available to give speech to the voiceless. The finest source of information regarding computer applications for the handicapped is the International Hardware and Software Registry, available from Trace Research and Development Center at the University of Wisconsin, Madison, Wisconsin. The Registry lists all of the latest programs and adaptive interface devices designed to aid the handicapped, including brief descriptions of the programs and hardware, availability, and cost. For computer applications for the low-incidence handicapped students, the Registry is the source.

For the mildly handicapped, there are any number of Computer Assisted Instruction programs designed for non-handicapped learners which are useful to mildly handicapped learners as well. Programs are available that can be tailored to the individual learner's rate of progress, and that provide adequate visual and auditory reinforcement often so necessary to maintain the attention of the mildly handicapped. Additionally, there are a number of simulation programs (programs that simulate a real life situation) that have been specifically developed for the adolescent handicapped student. They cover such areas as job readiness, interviewing, money management, etc.

Academically oriented programs of many types are available as well. Programs which give drill and practice in spelling, math, and reading recognition and comprehension are readily available. As with most programs, some are quite good, and some are, quite frankly, poor at best. The teacher or curriculum worker charged with finding and acquiring appropriate software for handicapped students is facing a major task. The task requires a great deal of time and energy. There are no clear choices, since the needs of the individual students are so diverse. The teacher will, of necessity, have to rely on his/her own knowledge of the student's strengths and weaknesses and sift through the volume of programs available to find the closest match. In the next section, some guidelines for assessing software programs will be offered.

The final category of software which we will discuss is the Computer Managed Instruction programs. Here we will consider programs which manage the instructional process in the broadest sense. Included are programs which aid in scoring tests and test batteries, programs which monitor student progress, and place students at the appropriate levels of instructional materials, and programs which combine a diagnostic component with a remedial component at the appropriate level, based on the student's performance. As with the Computer Assisted Instruction programs, some are quite good, and other programs are quite poor. Many are adequate for specific settings, while at the same time are not appropriate for other settings. There have recently been a number of programs designed to aid in the development of the Individualized Educational Program (IEP). Many of the programs are well designed and efficient, but lack the flexibility needed to be useful in a variety of settings. Again, the teacher will need to take the time to explore the program in enough detail to ascertain the usefulness in his/her own particular setting.

One final word about the software available for special education. At this point in time, it appears that the best software, and, consequently, the best use of microcomputers, is designed for management of instructional programs, and for record keeping and associated report writing. The lack of good quality software for handicapped students is due mainly to two factors. Primarily, since software programs are so expensive to develop, it is difficult to find professional programmers who are willing to invest their time and efforts into developing programs which have, by definition, a limited market. Secondly, even when good programmers do write software specifically for the handicapped, they tend to do one of two things. They write a specialized program for a specific individual or small group of individuals which proves to be extremely effective. However, since the program was developed for a specific client, or small group of clientele, the program lacks the generality necessary to be effective with other handicapped students. In many cases, it does not take too many trials of spending 200 to 300 hours developing a program for a select few, which is not marketable, for the programmer to realize that, while charity is emotionally gratifying, it does not "pay the rent". Also, programmers are not typically very familiar with handicapped students and their unique needs.

The story of a programmer who was volunteering his time to an agency serving retarded citizens comes to mind. It seems this fellow was dating one of the teachers, and had spent some time on outings with some of her students. When asked if he could write a simple program for the agency's microcomputer to accomplish task X with several students, he was pleased to lend his talents. The teachers described what they thought they would like the students to be able to do with the computer, not knowing a lot about the capabilities of the computer. As the story goes, the programmer went away with the teachers' ideas, and returned a few weeks later with a finished program, not knowing alot about handicapped students, other than his few outings with some of the students. When the teachers sat excitedly at the computer to test out the program before introducing it to the students, they gained a little more knowledge of what the computer could do, and developed a list of "suggestions" for improvement. The programmer, who was still dating the teacher, agreed to make the changes. A few weeks later, he returned with his "improved" program. The teachers were pleased with the results, and introduced the program to the students. After a few days with the program, the teachers began to see some "minor" problems with the program, but not knowing much about programming, could only tell the programmer what the would like to have in place of what currently existed. As the story was related to the writer, the "boyfriend", programmer, stuck it out through about two more revisions before beginning to date a woman who did not teach.

While the story may have been embellished in the telling, the point is valid. Teachers, at least most teachers, do not know about programming. Programmers, at least most programmers, do not know about teaching. Until the teachers and programmers are able to communicate effectively with each other, progress in developing top quality software will continue to be slow. Over the past several years, major publishing houses are hiring educators to design programs, and programmers to write programs. This trend does not seem to have carried over into the special education field at this point in time. Again, the numbers of students who represent the market for the software are insufficient to excite the major publishers into production of programs designed specifically for the handicapped.

Since teachers in special education are faced with problems in finding appropriate software for classroom use, it is important that the teacher be able to evaluate existing software to determine if the software can be utilized in the special education setting. The next section will deal with software evaluation.

SOFTWARE EVALUATION

The process of evaluating a program for the purpose of deciding if the program is worth purchasing is perhaps one of the most important tasks of the teacher who intends to utilize microcomputers in the classroom setting. The evaluation should proceed in a systematic fashion using a standard format, so the teacher uses similar criteria to judge the value of competing programs. There have been many suggested formats for use in this evaluation process. The specific format one chooses is probably dependent upon the nature of the students the teacher faces every day. Certainly, the teacher of mildly handicapped students will be looking for different elements than will the teacher of the physically impaired.

An example of a software evaluation format used by the writer and a number of "pioneering" teachers, those who were among the first to utilize microcomputers in their classrooms, is presented here as a guide for others who face the task of software evaluation. The reader should feel free to either adopt this format for their own use, or modify the format to more closely fit their own needs. We will first look at the form, then examine the rationale for inclusion of the information asked for in the form.

SOFTWARE PACKAGE EVALUATION FORM

Name: Date:

Program name: Cost:

Copyright date:

Vendor's name, address, phone:

Available for (brand, model(s), memory):

Peripherals required:

Primary objective(s) of program:

Prerequisite skills needed:

Protection - ease of modification:

Program type:

___Communication aid	___Environmental Control
___Drill and Practice	___Entertainment
___Tutorial	___Motor Development
___Word Processing	___Data Base Management
___Reinforcement	___Other ______________

Names, addresses, and phone numbers of other users:

Rating scale: 4 = Highest 1 = Lowest NA = Not applicable

	Documentation	Circle one				
A.	Clear explanation of purpose	4	3	2	1	NA
B.	Clear explanation of expected teacher/student input	4	3	2	1	NA
C.	Listing of alternative input devices previously utilized	4	3	2	1	NA
D.	Provides examples of questions or problems	4	3	2	1	NA
	Program					
A.	Vocabulary level appropriate	4	3	2	1	NA
B.	Graphics clear and easy to comprehend	4	3	2	1	NA
C.	Record of student progress/ achievement results	4	3	2	1	NA
	Overall usefulness					
A.	For teacher	4	3	2	1	NA
B.	For handicapped learners	4	3	2	1	NA

Comments:

The Evaluation Form on the previous page is entitled a <u>Software Package</u> Evaluation Form. A software package is the floppy diskette <u>and</u> the accompanying user directions or <u>documentation</u>. The floppy disk containing the program itself, is often useless without the documentation, which explains the use of the program. It is often simply not possible or feasible to incorporate all of the information about the program into "on screen" directions. Often there are special control characters which access parts of the program which are otherwise invisible to the user. Frequently, the user needs to read several pages of explanation in order to properly customize the program to the use within a specific classroom.

Documentation may be brief, as in most CAI programs, or it may be extensive for CMI programs which combine a diagnostic component with an instructional component. Most Data Base Programs and Word Processing Programs have extensive documentation, since the user needs to learn somewhat complex procedures to utilize the program. Regardless of the length of the docmentation, it is an essential part of the software package.

The form provides space for recording the evaluator's name, and the date the program was evaluated. The name of the program, copyright date, and approximate cost should also be recorded for reference at a later time. The name of the vendor, and where the vendor may be contacted to place an order, should you decide the package is appropriate should be recorded to facilitate filling in a requisition form through your district purchasing department. It is important to determine and record which particular brand and model computer, and what memory capacity is needed by the program, since you may have several computers with different hardware configurations on which the program might be utilized. Also, any special peripheral devices which are required of the program should by recorded, to ascertain if the program will operate on your existing hardware, or if additional components would be required.

The stated objectives (or implied objectives) should be located and recorded, and used as a basis upon which the program should be rated. Remember that programs differ in their intent. One math tutorial program may have as an objective the introduction of a concept, while another may have as an objective the mastery of a concept. Clearly, if the teachers goal were to find a program which strives for mastery of a concept, the program which only claims to introduce the concept would not fit the teacher's needs. It will be necessary in many cases to thoroughly read the accompanying documentation to find these pieces of information. Some documentation booklets may also give you the prerequisite skills expected before the program is to be used, but most program documentation will not offer this information. When the prerequisites are not listed, it will be up to the teacher and the teacher's general knowledge of instructional sequence and the teacher's experience to determine what skills are prerequisite to a specific program. To make this determination, it will likely be necessary for the teacher to spend sufficient time examining the program by going through the program as the student would.

For special educators with some advanced knowledge of BASIC, the degree of program protection, and the ease which the program could be modified or tailored to meet the individual's needs is an important factor to consider. Also, the degree of protection is important to know, since you will undoubtedly want to make an archival copy of the program after purchase. The copy will then be stored away in a safe place, and used only if the original copy is destroyed, or wears out with useage. Many companies are now offering low cost backup copies at the time of purchase. One should find out what the publisher's policy is regarding these backups. A few publishers offer low cost replacement of diskettes which are damaged or wear out. If the publisher does not offer low cost backups, then find out how difficult it will be for you to make your archival copy.

Categorize the program into one of the types listed under program type, and then file your evaluation forms according to the program type for easy reference in the future. As mentioned previously, ask the dealer for names and phone numbers (or addresses) of other persons who have purchased the program. Record this information, and then take the time to contact these persons to seek some evaluative comments from them. Remember to try to get a specific idea of their usage of the program, since all programs are good for some students, but not all programs are good for all students.

Use the rating scale to score the program on a scale of one to four on the listed items. First rate the documentation, then rate the program. Finally decide how the program rates overall, either for teacher's use, or for classroom use with students. Some explanation of the listed items is likely in order. The first point under documentation addresses whether the doumentation gives a clear explanation of the purpose of the program. Be cautious of excessive claims made in the documentation. The second point refers to how well the documentation explains to the teacher and/or the student, what types of responses they are expected to make to inquiries as prompted on the screen. The third point is more specific to more severely handicapped learners, and refers to the idea of alternatives to standard keyboard input to interact with the program. If there have been previous experiences with these methods, and the potential learner needs one of these alternatives, rate the stated success of those methods. The last point regarding documentation is whether the documentation provides clear realistic examples of the problems presented by the program, or samples of the program content.

In rating the program itself, look at the level of the vocabulary used both in the directions given to the student, and in the problems or content itself. If graphics displays are used in the program, look at the quality of the graphics. Many educational programs use low resoultion graphics to display large letters and numerals, and many teachers to whom the writer has talked do not feel the quality is adequate. Also, sometimes the graphics displays do not seem to fit well with the content of the program, as if the graphics are inserted for interest, without being well integrated into the purpose of the program. Finally, for CAI programs, determine what kind of (if any) record of student responses and/or student progress is maintained by the program. If the program does not make any record of the student's responses, the teacher will have to determine the value of a program which does not allow the teacher to monitor progress. If the teacher has to watch what the student does, the program is going to create more work than it will save.

The final, global rating can be used to compare the present program to similar programs, in order to develop some system of comparing competing programs. If the teacher will use this, or a similar format of evaluating all programs, the critical choices among similar competing programs will be easier, and probably wiser. As a rough guideline, based on the writer's experience, the average time required to evaluate a CAI program is about one hour. Other programs generally take longer to evaluate.

FUTURE TRENDS

The future holds continued change in the field of microcomputers. The "latest generation" of microprocessors are designed to handle information at speeds ranging from six to twenty times faster than their predecessors of only a few years ago. The lastest generation of Apple Computers, The LISA and Macintosh employ greatly enhanced methods of interaction with their users. Both of these machines use icons, screen pictures, and a device called a mouse, a small box with a button which you move across the desk, to select from different program choices. They allow writing, drawing, calculations, and graphical representation of data by selecting the appropriate icon using the mouse. All of these features can be running in the computer simultaneously.

A new Operating System was introduced for the Apple %bb%fb series in the winter of 1984. PRODUS is designed to facilitate the use of a hard disk, a device which utilizes a permanent metal disc enclosed in a protected cabinet, and which usually stores 5 million to 20 million bytes of information. The Apple %bb%fb family of computers now have available "integrated" software packages. These software packages allow the user to combine word processing, data base management, and calculations within the same program. Also, the Apple %bb%fb family now has available a mouse, and a drawing program which emulates the Macintosh drawing program. There is continued talk of a new microprocessor for the %bb%fb Series. Rumor has it that the new processor will be a 16 bit, as opposed to the present 8 bit, processor. Speed would be dramatically improved with this addition. Also, for persons who want more speed out of their Apple, there are presently several "speed-up" boards on the market that increase processing speed by two to six times the normal speed.

In the field of hardware for use with handicapped, voice input modules with a built in recognition vocabulary of one hundred to two hundred words are becoming reasonably priced. Voice synthesizers are increasingly more lifelike, and less expensive. Many enhancements for printers are available that use a dot matrix printer to produce Braille. The rehabilitation engineering field is producing prototypes which may become more affordable in the future.

Today, the use of computers in education in general, and special education in particular, is in its infancy. The length of the life of the computer in the classroom, as well as the quality of that life depends on teachers becomming more aware of the capabilities and current limitations of microcomputers. As teachers become more familiar with the currently available programs, and what could be produced, they will need to communicate their wishes to the publishing houses. The publishers can have a great impact on what types of programs are developed by programmers. If educators refuse to settle for second rate programs, and gain the committment of the community to fund computer usage in education, the computer has a chance to provide a meaningful enrichment in the education of our youth. If teachers do not become involved, or if the community does not choose to lend its support, the computer is likely to face the same fate as did educational television. The challenge awaits all of us.

SELECTED PROGRAMS OF INTEREST TO SPECIAL EDUCATORS

WORD PROCESSING PROGRAMS

The Executive Secretary
SOF/SYS, Inc.
4306 Upton Avenue South
Minneaplois, Minnesota 55410

Applewriter (II and IIe versions)
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

The Bank Street Writer
Bank Street College of Education and Broderbund Software
17 Paul Drive
San Rafael, California 94903
(also available from)
Scholastic Inc.
50 West 44th Street
New York, New York 10036

DATE BASE PROGRAMS

Information Master
High Technology Software Products, Inc.
P.O. Box 14665, 8001 N. Classen Blvd.
Oklahoma City, Oklahoma 73113

Quick File II (for the IIe only)
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

D B MASTER
Stoneware Microcomputer Products, Inc.
50 Belvedere Street
San Rafael, California 94901

COMPUTER ASSISTED INSTRUCTION PROGRAMS

For younger students:

Face Maker, Story Machine, Snooper Troops (1 and 2),
In Search of the Most Amazing Thing,
Hey Diddle Diddle, and others
Spinnaker Software
26 Brighton Street
Belmont, MA 02178

Micro Mother Goose, Alphabet Beasts & Co.,
Learning With Leeper, Dragon's Keep,
and Troll's Tale
(available from) Scholastic Family Software
1290 Wall Street West
P.O. Box 645
Lyndhurst, New Jersey 07071

Juggle's Rainbow, Bumble Plot,
Gertrudes Puzzles, Gertrudes Secrets,
and Bumble Games
The Learning Company
4370 Alpine Road
Portola Valley, California 94025

Apple Logo
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

For older students:

Shell Games
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

Game Show and TIC TAC SHOW
Computer Advanced Ideas
1442A Walnut Street, Suite 341
Berkley, California 94709

Alien Addition, Minus Mission, Alligator Mix,
Meteor Multiplication, Demolition Division, and Dragon Mix
Developmental Learning Materials (DLM)
One DLM Park
Allen, Texas 75002

Counting Bee
Edu-ware Services
Box 2222
Agoura, California 91301

Punctuation, How to Read in the Content Areas,
and How to Read and Solve Math Problems
Educational Activities, Inc.
Box 392
Freeport, New York 11520

Language Arts and Math Sequences
Milliken Computer Courseware

Mastertype
Lightning Software
P.O.Box 11725
Palo Alto, California 94306

Special Needs and Elementary Volume 7
Minnesota Educational Computing Consortium
(MECC)
2520 Broadway Drive
St. Paul, Minnesota 55113
(In Illinois)
Project Micro-Ideas
2941 Linneman
Glenview, Illinois 60025

Consonants and Blends, Vowels, Clocks,
Word Families, Cassette Control Device
Hartley Courseware, Inc.
Box 431
Dimondale, Michigan 48821

<u>For the Severely Handicapped</u>

<u>International Hardware and Software Registry</u>

The programs available through this source are too numerous to list. If you work with severely handicapped, this book is your best source of programs for use with your students.

Trace Research and Development Center
314 Waisman Center
1500 Highland Avenue
Madison, Wisconsin 53706

The Voice

Muse Software
(available from)
Scholastic Family Software
1290 Wall Street West
P.O. Box 645
Lyndhurst, New Jersey 07071

BOOKS ABOUT THE APPLE %bb%fb COMPUTER

The Applesoft Tutorial
Applesoft Reference Manual
Applesoft Tool Kit
DOS Programmer's Manual
DOS User's Manual

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014

Educational Software Directory
Apple %bb%fb Edition

Swift Publishing Company
P.O. Box 188
Manchaca, Texas 78652

The Apple Blue Book

WIDL VIDEO
5245 W. Diversey Avenue
Chicago, Illinois 60639

The Book of Apple Computer Software

The Book Company
16720 Hawthorne Blvd.
Lawndale, California 90260

Apple %bb%fb User's Guide
(the author uses this book as a text
in his classes)

Osborne/McGraw-Hill
630 Bancroft Way
Berkley, California 94710

USEFUL MAGAZINES ABOUT THE APPLE

Creative Computing
P.O. Box 789-M
Morristown, New Jersey 07960
- A good source for general information about the latest developments in computing

inCider
80 Pine Street
Peterborough, New Hampshire 03458
- an excellent magazine for beginning Apple enthusiasts

Nibble
P.O. Box 325
Lincoln, MA 01773
- if you want to learn how to make the most of your Apple, don't miss this one.

Personal Computing
50 Essex Street
Rochelle Park, New Jersey 07662
- another good general magazine which reports on the latest trends in computing

Softtalk
11021 Magnolia Blvd.
North Hollywood, California 91601
- reviews of the latest software for the Apple

Answer to programming exercise.

The following is one of many ways you could satisfy the assignment. As you recall the assignment was:

1. Draw an outline around the LORES screen in the color white.
2. Draw a line in the color green from the upper left hand corner of the screen to the lower right hand corner of the screen (the line will actually look like steps).
3. Place a small rectangle, two blocks by two blocks on the sides, in the very center of the screen in the color of your choice.

```
10   GR
20   COLOR = 15
30   C1 = 0 :C2 = 0 :R = 0
40   REM THE OUTLINE
50   HLIN C1,C2 AT R
60   HLIN C1,C2 AT R + 39
70   VLIN R,R+39 AT C1
80   VLIN R,R+39 AT C2
90   REM NOW THE DIAGONAL LINE
100   COLOR = 4
110   FOR R = 0 TO 39
120   PLOT C1,R
130   C1 = C1 + 1
140   NEXT R
150   REM NOW THE RECTANGLE
160  COLOR = 11 : REM PINK
170   PLOT 19,19 : PLOT 19,20 : PLOT 20,19 : PLOT
       20,20
180   END
```