If you are thinking of buying a computer, if you have just bought one, or if you are simply trying to understand what the computer explosion is all about, then The Personal Computer Handbook is the book you need.

Written in the language you understand, not the one the machines speak, it de-codes the jargon and offers a practical guide to understanding and using computers – whether you want to store information, to handle accounts, to plot statistics, to teach your children, to play computer games, or to write your own programs. From the mysteries of RAM and ROM or bits and bytes to those of bootstraps, handshaking and nibbles, The Personal Computer Handbook is a straightforward, step-by-step guide to the process of buying and learning to use a computer at home or at work.

Fully and clearly illustrated with numerous specially commissioned photographs and drawings, The Personal Computer Handbook is the first book on the subject that is truly “user-friendly”.

PETER RODWELL is one of the most respected figures in the field of microcomputing today. As editor of Personal Computer World, he has played a key role in establishing it as the largest selling and most prestigious micro magazine in Britain. He has been professionally involved with computers since the advent of the first micros, and, as a journalist, he has taken a special interest in word processing and electronic publishing. In 1981, he helped set up the ComputerTown UK! network of computer literacy centres, and he now runs his own information technology consultancy.

ISBN 0 86318 014 0

£12.95
THE PERSONAL COMPUTER HANDBOOK

THE FULLY ILLUSTRATED, FULLY COMPREHENSIVE GUIDE FOR EVERY MICRO USER

How computers work – from RAMs and ROMs to bits and bytes

A jargon-free guide to computer software – for games, business and education

An inside look at the hardware – what goes on beneath the lid

Contents include: What computers can do ★ How computers think ★ Setting up and getting going ★ Running programs ★ Fault-finding ★ The hardware and the software ★ Writing programs in Basic ★ Debugging ★ Games and graphics ★ Word processing ★ Buyer’s guide to choosing a computer ★ and much, much more

“It is unlikely that any of this year’s micro intro books will outsell The Personal Computer Handbook”

Micro Business, October 1983

DORLING KINDERSLEY
Dorling Kindersley Computer Books

The Personal Computer Handbook heralds a series of quality computer books from Dorling Kindersley. With the emphasis on superb graphics and clear, integrated text, these books will form a new benchmark in computer publishing.

Watch for:

- A complete course of screen-shot programming manuals for the Sinclair Spectrum, the BBC Micro, the Commodore 64, the Acorn Electron, and the other best-selling personal computers
- Peter McWilliams – “the Dr Spock of computers” – phenomenally successful US author, now published in book form for the first time in the UK

AND

Dorling Kindersley Software

A list of personal computer programs offering strong editorial content, excellent graphic presentation, and great value for money.

Watch for:

- High-quality arcade games
- Big-name adventure games
- Classic strategy – with a difference
- Educational software opening up exciting new avenues
THE
PERSONAL
COMPUTER
HANDBOOK
THE
PERSONAL
COMPUTER
HANDBOOK
PETER RODWELL

DORLING KINDERSLEY • LONDON
## CONTENTS

<table>
<thead>
<tr>
<th><strong>INTRODUCTION</strong></th>
<th><strong>BEGINNING COMPUTING</strong></th>
<th><strong>HOW COMPUTERS WORK</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>8 What is interesting about computers?</td>
<td>34 Setting up</td>
<td>50 How computers “think”</td>
</tr>
<tr>
<td>9 What is a computer?</td>
<td>37 Familiarization</td>
<td>54 Computer logic</td>
</tr>
<tr>
<td>14 What is a computer system?</td>
<td>40 Using cassettes</td>
<td>56 Electronic logic</td>
</tr>
<tr>
<td>18 Inside a computer</td>
<td>42 Using disks and cartridges</td>
<td></td>
</tr>
<tr>
<td>22 Computer applications</td>
<td>44 Programming tips</td>
<td></td>
</tr>
<tr>
<td>24 The history of computers</td>
<td>47 Fault finding</td>
<td>60 Putting the bits together</td>
</tr>
</tbody>
</table>
Welcome to the world of the computer. Every day, it seems, new and wonderful technological advances are announced and frequently there is a computer playing some part in the story. Increasingly, everyday and familiar objects are gaining new and unexpected abilities, again thanks to computer technology. You could be forgiven if at times you feel surrounded and not a little overwhelmed by the speed and scope of the advances taking place. But despite any preconceptions you may have to the contrary, you will discover in this book that it is more a case of the computer adapting to your world than you adapting to its.

The purpose of this book is to try to impart to you a sense of the excitement and change that the widespread use of computers in the home and work place will inevitably bring. If you feel you should know more about this new technology, there is no better way than to go out and buy yourself one of the popular home computers now available so cheaply. Even if you start off by playing prepackaged games, you will still be developing some degree of familiarity with computer technology and an appreciation of what it involves.

Traditionally, in science fiction and in the brochures and advertising put out by some elements of the computer industry itself, we have all been conditioned to think of the computer as an all-powerful entity. As you will soon discover for yourself, this concept is very far from the reality of the situation.

How to use this book
Computing is a vast subject with a whole multitude of byways, each of which would – and does – need an entire book to explain fully. What I have done in this book is to answer a few basic questions:
- what is a computer?
- what can a computer do?
- how does it work?
- what relevance does it have to me at home and work?
And many more.

The book has an overall structure to guide you through the subject, but the structure is not so rigid as to stop you deciding how you go about reading it. And each section of the book is color coded for easy reference. You can, if you wish, start here and read through from the first self-contained module to the last, although in a subject like computing there are so many parallel "streams" of knowledge that any sequence is open to debate. Or you can jump around from one module to another, picking up streams and finding that there are plenty of cross-references to explanations of terms you might not understand. Unfortunately, jargon is a necessary evil. It is a verbal shorthand that allows us to convey a concept quickly, without the need to go into lengthy explanations every time. You will find yourself rapidly picking up the terminology, but if you come across an unfamiliar term or expression – look in the glossary where all is explained.
WHAT IS INTERESTING ABOUT COMPUTERS?

Computers are now almost everywhere in one form or another. Years ago, a computer took up a large room, weighed several tons, required expensive air conditioning, and was attended to by a whole staff of highly trained experts. It was a remote, powerful beast which demanded that things be done its way and caused havoc when it occasionally went wrong.

Today, the chances are high that you have some computers in your home without even realizing it. At the very least, you probably operate a computer unknowingly several times a day. Technology has advanced to the point that it is possible to make a computer no larger than the nail of your little finger—it may not be very powerful, but it can be used in a surprisingly large number of applications. And wherever it is used it brings more versatility or greater ease of use and, at times, more bother and inconvenience when the technology is mis- or badly applied.

Just take a quick look around your home and see whether you are a secret computer user. How about your television set—has it a remote control? If so, chances are there is a tiny computer inside it. Or your telephone—is it the type that can remember numbers and dial them for you at a press of a button? Even if it is an ordinary telephone, when you use it you are controlling a substantial amount of computerized equipment, which is dedicated to routing your call to the person whose number you have dialed.

Of course, it could be that you have a true microcomputer in your own home—several million people worldwide now have home computers, and sales of these compact but versatile machines are soaring. At work, too, more and more people are coming into contact with computers. Since almost the start of the computer industry, business, science, and education have all been big users of computers. Now, though, you will find computers sitting on desks with people who have had no formal training in computer science using them as part of their work. Computers help managers plan, they provide access to information, they help people produce letters, reports, and books. Provided care is taken in choosing the right system, these little machines can considerably improve not only the quality of your work but, more importantly, the quality of your working life, relieving you of much boring drudgery which most of us sometimes suffer.

Now, more specifically to the question posed above, it is difficult to put your finger on exactly what it is about computers that is so interesting. But fascinating they are, once you have overcome your original conditioning. They can also be highly addictive. It is not uncommon to hear tales from friends who have been up half the night trying to work out the complexities of a particular computer program that has been causing them problems. This is not a job of work, but a pleasure or a way of relaxing.

Part of the fascination of computers must surely lie in their complexity. As you shall see later in the book, a computer is a complex collection of basically pretty simple components, but it is this complexity that can cause you problems when you are trying to make the computer do something. One of the principal problems is learning a common language—one that both you and the computer can understand. Once this hurdle is overcome, the way is clear for you to come to terms with this new tool.

Computers are also interesting because they are becoming so widespread. And this proliferation is due to another major point of interest about them—a computer when you buy it or come into contact with it for the first time is a "blank" machine. It needs to be told exactly what to do, step by step, and, within certain constraints, it can be programmed to perform an extremely wide range of different tasks. As an example of the type of applications you can put a computer to, imagine the scenario where your computer helps you to write by becoming a supremely versatile typewriter; later, you might decide to face the awful truth and ask the computer to give you a summary of your financial state; having recovered from this shock you might feel that some form of relaxation is definitely called for, so you challenge your computer (with the appropriate program) to a game of chess. And, unless you are a very good chess player, you will probably lose. And these types of applications are only a few of the hundreds possible.

There is no simple answer to the fascination of computers. Reading this book should, hopefully, give you some idea of what they are all about. But buying one will show you very quickly indeed.
WHAT IS A COMPUTER?

We are today surrounded by remarkable electronic devices – television sets, video recorders, hi-fis, washing machines, microwave ovens, push-button telephones, and countless other consumer goods. Increasingly, these goods have the words “computer”, “micro”, or “digital” prominently displayed on them. The very words have developed their own mystique. In the jargon of high technology they imply efficiency, reliability, and a state-of-the-art quality. Yet how do these products really relate to computers? What exactly is a computer?

Each of the devices mentioned is designed to perform a specific task – and that task alone. The computer, however, is different. It is a “blank” machine which, by itself, can do nothing. Basically, it is a collection of electronic circuitry arranged so that it can respond in a predictable way to any impulse or input it is given. Yet the nature of the response can be altered by the user. It is this “alterability” that distinguishes the computer from the other electronic devices.

Computers need to be told what to do. They must be given a list of instructions (or “program”), specifying in detail each step of the job they are being instructed to perform. Once they have received them, they will follow them exactly, doing precisely what each one says – no more and no less. The nature of the program means that if you make a mistake in your instructions then the result will not be what you want. The computer has no way of knowing what you intend it to do.

However, having successfully instructed the computer to perform one task, you are quite able to go on and instruct it to do something else. You just need to give it a new set of instructions. The computer is a multi-purpose device, then. It is able to perform a number of different tasks, not just one.

Devices like modern washing machines sometimes do actually contain a tiny computer. And, in fact, the only real difference is that the washing machine’s computer is dedicated to just one task – that of controlling the machine. This is because it has been given one permanent set of instructions. You could, theoretically, remove it, program it with a new set of instructions, and make it perform an entirely different job.

The different types of computer
There is a convention that classifies computers according to their size and computing power. At the top are the mainframes, the giants used in large companies, government departments, and scientific

MAINFRAME TO MICRO

There are three categories of computer. The mainframe is the largest and is found in big companies, universities, and government departments. Mainframes are immensely powerful, capable of handling hundreds of users at the same time.

A minicomputer is basically a scaled-down mainframe. It has the capacity to allow several people to work simultaneously. This possible limitation is offset by the lower cost and the fewer number of people that are needed to run and maintain a minicomputer system.

A microcomputer is one that uses a low-cost, mass-produced microprocessor chip as its central processing unit. Few microcomputers can handle more than one user at a time, but because they are so cheap each user can be given an individual machine.
IBM'S FIRST MICROCOMPUTER

The entry of IBM into the personal computer market was probably the most significant event in the industry's short history. Previously, most personal computers were the products of small, new companies, set up to exploit the growing interest in computing. The lack of involvement of the big computer companies also created a suspicion in the minds of many potential customers that the micro industry was not to be taken seriously. The arrival of IBM on the micro scene, then, made the industry "respectable" in many people's eyes, and prompted most other large computer manufacturers to go "down-market".

The machine itself has had a tremendous impact on the micro business. It pioneered new ground by incorporating a 16-bit processor, one of the first micros to do so. In America, the low price of the basic model in the range made it a strong contender in the upper end of the hobby/home market as well as for business use, and a massive industry has grown up around it, providing not only software but also plug-in expansion cards for the "PC" as it is commonly known. IBM waited 18 months before launching the PC in Europe, and this delay allowed other 16-bit systems to move in.

Monitor The standard IBM PC machine comes with a monochrome monitor. If this is too limiting, with the addition of an optional plug-in card, a color monitor can be used instead. The value of a color display is now well recognized by both business users and home hobbyists (see Module 20).

Main unit The main "box" houses the microprocessor (see Module 17), memory chips (see Module 19), disk drive units (see Module 22), and all the necessary electronics that go to make up a modern microcomputer. Also, inside is a row of slots into which extra cards can be plugged to expand the machine's RAM capacity (see Module 19) and provide additional input/output ports (see Module 20).

Keyboard As major office equipment manufacturers, IBM knows more than most others about keyboards and keyboard layout (see Module 20). The PC's keyboard is one of the better ones in the industry, although it has received a little criticism for some aspects of its design. Generally, though, anyone familiar with a traditional IBM typewriter should feel immediately at home with the keyboard of the PC.
and educational establishments. Mainframe computers are extremely powerful. They are capable of handling many different jobs at the same time and will allow as many as several hundred people to carry out computing work simultaneously. They are, however, very expensive, and require not only highly skilled staff to operate them but also special air-conditioned, dust-free rooms.

Next in line in terms of computing power is the minicomputer. This is a smaller version of the mainframe, using similar design techniques. Most minis can cope with fewer users simultaneously, but many do not need full-time specialist attention or a special environment in order to work properly.

These days it is rather difficult to define a mini in exact terms. At the top of their range, the most powerful out-perform the bottom-end mainframes. At the other end, certain minis outperform minis. It has been said that a minicomputer is no more than a computer made by a minicomputer manufacturer.

The days of the mini as it currently exists are numbered. It is the microcomputer or personal computer that is taking its place. Some micros already offer greater power and more facilities at lower prices than some small minis, and we will soon see micros challenging the mid-range minis as well.

A microcomputer is a machine based around a microprocessor or "silicon chip". And it is because micros can be mass-produced, while minis and mainframes are handbuilt, that they are so much cheaper. This, together with the continuing technological advances in chip design and manufacture, has been responsible for the micro revolution. Each new generation of micros — and new machines are introduced almost weekly — seems to offer greater power in a smaller box at a lower cost.

Because micros are now so cheap, almost everyone who needs access to a computer can afford to have one on their desk top. The cost of providing, say, 20 people with micros works out at far less than the cost of installing and maintaining a minicomputer capable of serving the same number of users. Moreover, to expand the system, new people are simply given a new micro. Minicomputers not only have an upper limit to the number of people who can use them, but also require elaborate and expensive programs to handle a large number of users.

THE POCKET COMPUTER

The development of microelectronics has led directly to the invention of the microprocessor (see Module 6), but miniaturization has not just produced the desk-top computer. Manufacturers are now marketing computers that have been reduced to pocket size (although, to be fair, you do need rather large pockets). The machine here, called the PCI, is made by Sharp and marketed by Tandy in the UK and by Radio Shack in the States. It incorporates a processor, a small but useful amount of RAM (see Module 19) and a full Basic interpreter (see Module 27) in ROM. You can use the PCI as simply a pocket calculator, or you can write programs in Basic and store them in RAM. With these RAM chips, the memory is retained even when the machine is turned off. An extra option is a plug-in cassette interface (see Module 21), which allows you to save programs on tape.

Liquid crystal display This single-line display can be scrolled up and down on the screen, and from side to side so that you can read lines too long to fit on to the display.

Keyboard A full alphanumeric keyboard (see Module 20) is provided.

Numeric keys These keys have been grouped together for easy use as a calculator.
The most notable physical effect of the microelectronics revolution has been the shrinking size of computers. From the roomfuls of equipment weighing several tons, which were the computers of ten or fifteen years ago (see Module 6), we now have machines that can be carried around anywhere.

There are two types of portable machines: some, like the computer here, are reduced versions of desk-top machines and include full-sized keyboards, small screen, and disk drives. They also need an external power supply. Others, like the machine on the facing page, have batteries for complete portability. Both, however, are full computers.

**RS232C interface** This allows the machine to connect to serial printers, or any device using this popular standard interface.

**IEEE-488 interface** This connects the machine here to the standard instrumentation bus for data communication with test instruments.

**Disk drives** The twin disk drives (see Module 22) are incorporated into the machine. It uses the popular CP/M operating system, making a very wide selection of software available.

**Case** The rigid plastic case snaps together to form a completely waterproof container that is readily transportable.

**Screen** The tiny screen (see Module 20) is just about legible if you sit close. You can, however, use an add-on monitor for office use.

**GO-ANYWHERE PORTABLES**

The face of portable computers is changing rapidly. The machine below was designed to be both cheap and compact, yet it still has a useful selection of ready-to-use software. Its caselike exterior unfolds (see right) to reveal a full keyboard and tiny screen (there is an add-on screen for office use). It also has two disk drives and there is space to store about 20 disks. Being Z80 based (see Module 17), it can run the CP/M operating system (see Module 33) and so there is a wide range of software available for it. The basic version comes with a word processor, spreadsheet, and database management system.

**The suitcase portable** The face of portable computers is changing rapidly. The machine below was designed to be both cheap and compact, yet it still has a useful selection of ready-to-use software. Its caselike exterior unfolds (see right) to reveal a full keyboard and tiny screen (there is an add-on screen for office use). It also has two disk drives and there is space to store about 20 disks. Being Z80 based (see Module 17), it can run the CP/M operating system (see Module 33) and so there is a wide range of software available for it. The basic version comes with a word processor, spreadsheet, and database management system.

**Keyboard** The keyboard (see Module 20) is built into the computer's lid. The keyboard can be detached to allow you to position it more comfortably. A cable connects the keyboard to the computer. Unlike some portables, this one has a full-sized keyboard.
The attaché-case portable

Epson is a Japanese company that has become world famous in the micro industry for its very reliable, low-cost dot matrix printers (see Module 23)—even IBM went to Epson for the printers for its now famous PC microcomputer.

Epson entered the market with a desk-top CP/M-based machine, the QX-10, and this, the HX-20—a fully portable computer. Rather than produce a machine like the one on the facing page, with a heavy screen and disk drives, Epson opted for a much smaller configuration—the machine is the same size as an ordinary sheet of typing paper, with a four-line LCD display.

The keyboard, though, is full sized. A tiny dot matrix printer is built in and you can add a microcassette recorder for storing data and programs. It can also be interfaced with a full-sized display using an RS232 interface.

LCD display

This shows four lines at a time, but an extra interface allows you to use a larger, conventional display screen.

Microcassette

As an option, a microcassette can be plugged into the machine. This can be used to record and play back programs and data.

Loudspeaker

Keyboard

The HX-20 has a full-sized typewriter-quality keyboard making portable word processing possible.

Paper feed button

Printer

A tiny printer is built into the computer and, although rather too small for word processing, allows figures and tables to be produced on the spot.

Monitor interface

Connects the machine to a monitor.

Storage space

If you have disk drives, you need disks, and these need to be stored somewhere when the machine is being carried around. This machine has storage space for up to twenty 5½ in disks (see Module 22).
WHAT IS A COMPUTER SYSTEM?

It is misleading to think of a computer as a single machine. Computers are actually systems. They comprise several components, linked together, each with its own job. Although it is not vital to know in detail how each of these works in order to use a computer, it is helpful to know something about the interaction between the various elements. First, we will look at components that perform the actual computing. These are the central processing unit (or CPU) and the system's internal memory. Next, we will look at the elements that allow the computer to communicate with you, and vice versa – the input/output (I/O) facilities.

The part of the system that “thinks”

The most important component of a computer system is the central processing unit. The CPU is the “brain” of the computer. It does all the hard work, and all the other elements in the system are connected to it in some way.

TYPICAL HOME SYSTEM

The minimum you need to put together a computer system is the computer itself – CPU and memory – plus some method of allowing communication between computer and user – keyboard and screen. It is also convenient to have a method of storing programs when the computer is turned off.

Here you can see a typical home computer system, with the electronics and keyboard contained in one unit. Rather than use a special screen, most home computers are designed to use a domestic television set. To store data and programs, another piece of familiar domestic hardware is used – a cassette recorder. Most home computers can use just about any kind of recorder, although you may find that some computers are more fussy than others about the kind of recorder they work best with. Some manufacturers, though, insist on the use of their own tape machine instead. This costs more, but is usually more reliable.

Screen Output from the computer is displayed on a domestic television set. Most modern home computers are capable of displaying color, although to take advantage of this facility you will have to link your computer to a color television set.

Cassette recorder One of the popular methods of storing output from the computer and of loading data and programs into the computer is an ordinary domestic cassette recorder (see Module 21). Although slower than disks (see Module 22), you do not have to buy a specialist piece of equipment.

Main unit The CPU (see Module 17), RAM (see Module 19), and other electronics are all contained in the main unit of the computer for tidiness and compactness. The main unit also houses the keyboard (see Module 20), used to communicate with the computer.
In a mainframe system the CPU is the size of a large cabinet, but in a micro it is a single component—the microprocessor or silicon chip. Two types of CPU are used in modern micros—8-bit and 16-bit chips. 16-bit processors are usually more powerful and can use more memory than 8-bit CPUs.

The parts of the system that "remember"
The extra memory capacity of the 16-bit machines makes them expensive, so most home machines use 8-bit processors—but on its own the CPU is powerless. It will not do any computing unless it receives instructions (the program) and the information (the data) on which it is to work. The program and data must be stored in the computer system, and this store is called a memory (see Module 19).

In short, there are two types of memory—RAM and ROM. RAM is used to store programs and data temporarily—that is, only when you need to use them and only for as long as the computer is turned on, as making security or "back-up" copies of programs and other data can be cumbersome with only one drive. This measure is essential in case a floppy disk is damaged and the information on it cannot be extracted.

For technical reasons, a domestic television cannot display text in more than 16 lines of 64 characters. For business use, this is really insufficient and a standard display of 24 or 24 lines of 80 characters is considered essential. In turn, this requirement demands the use of a professional-quality display, one that is not only capable of showing the extra amount but which has a crisper, easier-to-read appearance to its characters.

This last point is particularly important if you have to work with the screen display for hours every day. Most business users need to have information printed out on paper from time to time—text from a word processor, say, or invoices. Generally, a dot matrix printer is perfectly adequate for the latter type of job where print quality is not paramount. But where high-quality output is needed, a daisywheel printer, with a typewriter-quality print-out, is used (see Module 23).
on. ROM stores things permanently. Its contents are fixed during manufacture and, therefore, cannot usually be altered.

ROM and RAM chips have only a limited memory capacity. For storing long programs and large amounts of data when the computer is switched off, various types of mass-storage peripherals are used. These include cassette tape recorders, floppy disks, hard disks, and cartridges. Cassette recorders are extremely slow, however, and unsuitable for business use. Floppy disks are, therefore, the most widely used storage medium. They can do exactly what the tape does, but much more easily and quickly. Hard disks are even more efficient, and have a greater capacity, but they are still expensive and mostly restricted to business use.

The parts of the system that "communicate"

All computers need a way in which to communicate with the outside world. Without this facility you would be unable to put programs or data into the computer, and the computer would not be able to communicate its results to you. To overcome this the keyboard is the standard device for inputting data. The computer keyboard is like a typewriter keyboard, but with the addition of a few extra keys. There is, however, no paper — as you type, the characters appear on a monitor. This is a screen similar to a television set. While the monitor saves on paper, it cannot produce a permanent output. For this, you need a printer (see Module 23).

Modern printers come in two basic types — the dot matrix and the daisywheel. Dot matrix printers form characters on paper as a series of tiny dots. They are low-cost, high-speed units, but the quality of their output is not as high as that produced by daisywheel printers. Daisywheels are, however, slower and more expensive.

There are other, more exotic I/O devices available (see Module 24). One that you will come across often is the joystick or games paddle. This device is mostly used for games, but other, more serious "exotica" is available for things like long-range communication and graphics print-outs.

FAMILY TREE

A computer's CPU and its memory can be interfaced to a wide range of different devices (see below). Some are essential and are found on all computers — keyboard, screen, and some form of storage system. Others are designed for specialist use, such as graphics, speech input and output, or software cartridges. Often, too, computers are connected to other computers, either through network systems or via the telephone system using couplers and modems.
NETWORKS

When computers were large and expensive, it was considered to be uneconomical to allow only one person at a time to use the machine. It was discovered that, with the help of some clever software, the CPU could be made to switch quickly between tasks, performing a little work for one user, then the next, and so on.

Although some microcomputers have been made with this multi-user capability, generally the CPU is not powerful enough to carry out the job convincingly. Delays occur and this makes for inefficiency. The CPU is, anyway, cheap enough for users to have their own, individual units. However, expensive parts of the system, such as hard disks and daisy-wheel printers, are better used if several people can have access to them. To provide this facility, micros can be connected together to form a “network” of units.

The idea of a network is very simple - a group of machines are interconnected with a length of wire. Included in the network are microprocessor-based units, which handle things like hard disks and printers, or even allow access to other networks or to the telephone system. When a computer wants to communicate with another object in the network, it sends out a signal to identify the recipient. The latter has been “listening” to the network, and when it hears its signal responds that it is ready.

In practice, the whole business of joining up computers is more complex, and several methods have been devised - the bus, the star, and the ring.

The bus system uses a piece of wire to which all devices are connected via an interface. These interfaces are constantly listening to the signals being sent along the bus, and when one picks up its own signal it transmits the information on to its computer. When a computer wants to transmit information along the bus, its interface listens in case any other computer is using the bus. If it is clear, it will allow the computer to transmit.

The star network relies on a central device that controls the entire network - the network controller. This decides which computer is to receive a particular communication, and then sends it out in the appropriate direction. But if the controller breaks down, the whole system is inoperable.

The ring system suffers from a similar problem. The interface for each computer sits in the path of data traveling around the ring. Data is sent in “packets”, each of which begins with an identifying code of the device that is to receive it. Each packet is intercepted by the computer, and if it is not the intended recipient it retransmits it to the next element.

The star network

With this network, all information has to pass through a central system controller. The controller examines each message to find out its destination and then forwards it in the right direction. The whole system dies, however, if the controller is out of action for any reason.

The ring network

This system has a number of interfaces built in to it. Each interface intercepts and examines messages to see if they are intended for its particular computer. If not, the message is passed along to the next interface in the network, where it is again examined before being passed on.
With the top removed you can take a look inside a modern microcomputer and see what it actually contains. The model illustrated on the right – called the Sinclair ZX Spectrum – is typical of the new generation of small, cheap home computers. It offers color, graphics, sound, and a level of computing power that would have been unthinkable in such a small machine and at such a low price only a few years ago.

Once the keyboard is separated from the main case, you can see that all the components are mounted on a single printed circuit board. The overall effect is one of surprising simplicity; the computer seems to consist of little more than a handful of chips, a UHF modulator (which sends pictures to the television screen or monitor), a tiny loudspeaker, and a collection of capacitors, resistors, and diodes which are situated among a complex pattern of circuit board wiring.

The chips are the key components in the working of the computer. They are the rectangular blocks of black plastic connected to the board via the metal pins that run along their edges. The most important of these chips is the one known as the Central Processing Unit or CPU. It is the microprocessor that carries out the actual computing functions. By itself, however, the CPU can do nothing. It needs memory in order to function. This is because the computer needs instructions telling it what to do, and it has to be able to store these instructions – together with the information or data on which it is to work. It also needs to store the results of its work somewhere before transmitting them to the outside world in one way or another. Memory is therefore the next most important component of a computer system.

The memory chips
You will see later exactly how memory works, and what different types of memory are found in a typical system. At present, it is enough to say that there are two basic types – first, the type used to store instructions and information while the computer is switched off (Read Only Memory or ROM) and, second, the faster memory used to hold the information and programs on which the computer is currently working (Random Access Memory or RAM).
Cassette recorder connector

Gives access to tape-based storage (see Module 21).

Uncommitted Logic Array (ULA)

A complex chip that acts as a sort of "administrator".

Logic chips (see Module 14)

Central Processing Unit (CPU)

The most important chip - the computer's "brain" (see Module 17).

Read Only Memory (ROM)

A single chip that holds all permanent data (in the form of machine code) that cannot be overwritten - for example, the operating system or language interpreter. This data will be called on by the CPU whenever necessary (see Module 17).

Keyboard

Normally linked to the main circuit board by two ribbon connectors, the keyboard features the standard typewriter layout plus special keys (see Module 20).

Expansion connector

This facility allows a printer (see Module 23) and other peripherals to be connected.

Power connection

This allows the computer to be connected to the main power supply.

Voltage regulator

Loudspeaker

Small speaker mainly used to provide sound effects for computer games (see Module 29).
RAM). In the Spectrum, the ROM is all contained on a single large chip. Here are held many of the instructions that the CPU needs in order to do its job. In this case, these include the machine's operating system and the computer language Basic. Thus the ROM chip can "interpret" a program written in Basic by translating it into a form that the CPU can "understand" and carry out.

The computer's RAM is made up of a collection of smaller chips. This model, which has 48 kbytes of RAM, has 16 separate RAM chips. They store the information that the computer needs only temporarily – the programs that are being run or data for calculations being made, for example. When the machine is turned off, all the RAM chips are wiped clean. The computer "forgets" what they contained.

A look inside your computer can be interesting, but on most machines this will invalidate any guarantee offered by the maker. Also, some machines simply are not designed to come apart easily and you

APPLE IIe

The Apple II is one of the most successful ever microcomputer and dates from the very earliest days of the microcomputer industry. The IIe, pictured here, is the latest of several versions of the machine developed since its introduction. It has extra facilities, an improved keyboard, and simplified internal construction. Because the Apple II has been in production for so long, and because several hundred thousand have been sold over the years there is now a vast range of software and hardware add-ons available. Just about anything you can do on a microcomputer you can do on one of the range of Apple machines - although not necessarily more easily or more economically than on other microcomputers in the same class.

Keyboard The original Apple had a limited keyboard, which has been much improved in the IIe version. It is still, however, rather an awkward board for some business applications when compared to many more modern micros.

Printed circuit boards Like many other sophisticated microcomputers, the Apple has a number of internal connectors, to which extra boards can be added. These provide a range of facilities like disk drives (see Module 22), better displays (see Module 20), and analog-to-digital and digital-to-analog and other interfaces (see Module 24). A massive range of boards is available and the two shown here are a disk controller card (1) and a color video card (2).
Introduction · Module 4

could easily cause damage in the process (this applies particularly to the Spectrum, in which it is easy to tear the connections between the keyboard and the computer).

**Are all computers the same inside?**
Some microcomputers, like the Apple II pictured here, the IBM PC, and the Epson QX-10, are designed to allow the user easy access to the internals. All of these machines—and quite a few more besides—have a series of sockets inside. These are designed to allow the user to plug in various printed circuit boards to increase machine versatility—more memory (see Module 19), extra input/output ports (see Module 20), or color display facilities for games (see Module 29), for example. You can see these sockets in the Apple below, close to the back of the machine, and they make it an ideal machine for a wide range of business applications as well as home use.

**Inside the Apple** All the main components can be seen easily. The entire machine is constructed on a single printed circuit board. The “core” of the Apple is its CPU chip (1), a 6502 microprocessor. Controlling software is held in ROMs (2) and an area of RAM (3) is also provided. Special ROMs (4 and 5) hold software to generate characters on the screen and to operate the keyboard. Two large chips (6 and 7)—almost processors in their own right—control the memory and I/O. Groups of logic gates (8) perform miscellaneous functions in the system, and a series of connector sockets (9) allow users to plug cards containing disk controllers, extra I/O interfaces, and more RAM, for example.

**Speaker** A small loudspeaker gives the Apple limited sound capabilities. Several sophisticated sound synthesizers, however, are made to interface to the Apple.

**Removable panels for extra I/O connectors**

**On/off switch**

**Power supply connector**
It is quite false to think that computers can do just about anything. Not only is a computer incapable of doing anything unless it has been given the appropriate instructions but it is, in fact, severely limited in its capabilities. The computer is a very versatile device, but cannot do anything in the physical world unless it is provided with special connections. It will quite happily, though, work out a complex mathematical puzzle, then tell you the time, and then play a game of chess, before helping you write a letter. And this is the reason for the computer's success - it is a blank machine, which, within limits, can perform a wide range of jobs.

One of the computer's main assets is the speed at which it can perform tasks. Ask a computer to add up a column of a thousand figures and it will do so in a fraction of the time a person would need. Likewise, present it with an entire book in a form that it can "read" - on disk, say (see Module 22) - and ask it to search for a particular phrase. As you will see
The power of the computer depends very heavily on the ingenuity of its software (see Modules 25-36) and that, for the most part, is produced by humans. Meanwhile, here a few examples of the sorts of things for which computers are often used.

**COMPUTERS AT WORK**

Game-playing (1) is one of the widest uses for home computers. The Videotex market (2) is now gaining widespread acceptance in business as a method of obtaining updated information via the telephone system. Graphics (3) are not only used in the games world but have their place in business, too. Database management systems (4) allow you to arrange information in a structured way so that the computer can manipulate it as you require. A spreadsheet (5) allows you to enter figures and to define the relations between them. Then, by altering one figure, you can see instantly the effect on any other figures linked to it. With the introduction of word processing (6) the days of the typewriter are numbered. Using a computer to enter, store, and manipulate text boosts productivity because of its flexibility.
The History of Computers

Computers have evolved through four generations in only 40 years. This short period has seen the bulky and slow computers of World War II develop into the compact and ultrafast machines we know today. In the wings waits the fifth generation, a new breed of computer with even more power.

Computers seem to be carrying us into the future at such a pace that it is perhaps hard to believe that they are as much as 40 years old. Microchip technology that enables modern machines to be both small and fast — as well as cheap and reliable — dates back little more than ten years. But the 1940s is the most appropriate date of birth we can assign to computers, for it was then that they emerged in the form in which we know them today. These computers were electronic digital machines. Computers can work at great speed and undertake tasks of massive complexity only because they handle data in the form of digits or numbers and because they are powered by electronic circuitry.

The necessities imposed by war brought the computer into the world hastily but well formed, for its period of gestation had been long. The origin of computers goes back many centuries as a computer is a machine that handles numbers. It may work with words as well and perform many types of logical operations, but within the computer these are all rendered into strings of digits.

Early aids to calculation

Apart from the very simplest of mental arithmetic, our first means of calculation must have been by counting on our fingers. Finger counting with a total of ten fingers would seem naturally to lead to a decimal number system, in which numbers are represented by groups of ten symbols placed in order of value. We use the numbers 0 to 9 in this way with ease, but the advantages of such a system were by no means obvious to ancient peoples. Instead, they developed unwieldy systems, like Roman numerals with their arbitrary choice of letters to represent numbers. Imagine trying to divide DCCL by LXXV; instead of complex calculations we can solve it in our heads, for it is, in fact, 750 divided by 75.

Fortunately, ancient civilizations had the benefit of a calculating device developed in Babylonia about 5000 years ago — the abacus. So good was the abacus that it can still be seen throughout the world as both a toy and teaching aid. The abacus marks the first step on the road to the computer. Like the computer, the abacus handles numbers in the form of codes made up of the presence or absence of material things — beads strung on wires or rods in a frame. In a computer they are pulses of electricity or regions of magnetism. The patterns of beads in an abacus represent a number in a decimal place-value number system — each bead stands for a one, or a ten, or a hundred, and so on, its value depending on the position of its rod within the frame. As the beads are shifted to make a calculation, zero is given by moving all the beads on a rod back and one may be carried over to the next rod. In computers,
The abacus. This was the earliest calculating device, dating from about 3000 BC. The mid-nineteenth-century example above has been set up to represent the number 7230189.

electronic pulses move in and out of circuits to represent zero and to carry one over in the same way.

Although the Egyptian, Greek, and Roman civilizations used the abacus, they did not develop a number system with a symbol for zero. This essential feature of calculation seems to have originated in India or passed there from Babylonia, and it was eventually rediscovered by the Arabs in the AD 800s. The man responsible was the scholar Al-Khwarizmi, whose name is immortalized in the computer term "algorithm," meaning a method of yielding a solution to a problem. Zero came to Europe four centuries later.

Although people now had a proper number system that allowed complex problems to be worked out, many found it difficult to master, especially for multiplication and division. To their aid came the mathematician John Napier, who devised logarithms in 1614. Every number has a logarithm, which is a figure that can be found by consulting tables. To multiply numbers, the logarithms of the numbers are added together, and the answer found by looking up the number that has this total for a logarithm. Division is carried out by subtracting the logarithms. A simpler method of using logarithms came with the invention of the slide rule by the English mathematician William Oughtred in 1621.

John Napier also invented an effective aid to the multiplication of larger numbers in 1617. Called Napier's bones or Napier's rods, it consisted of a set of strips marked with numbers. To make a calculation, the strips were put together in a particular order and a series of numbers that appeared on the strips were added together for the result.

**The first calculating machines**

The next real step toward the computer would be a proper calculating machine, and such a machine was not long in coming. The first one was built by Wilhelm Schickard in Tübingen, Germany, in about 1623. The model was similar to the first calculator to be developed into a full working machine, which was invented without any knowledge of Schickard's work by the French scientist Blaise Pascal in 1652. Pascal determined that routine calculation was worthy only of a machine and so set out to build one. He built more than 50 models before he was satisfied. Unfortunately, financial success did not follow. Employers were not keen to liberate their poorly paid clerks in favor of an expensive machine, and nor were the clerks too happy with the threat posed to their livelihoods.
The history of the computer from now on is studded with great intellects like Pascal, and the next to enter the story made advances on several fronts. He was the German philosopher and scientist Gottfried Leibniz, who foresaw that a general-purpose calculating machine able to multiply and divide would attract mathematicians and bookkeepers alike. Leibniz studied Pascal’s calculator and similar machines built later on the same lines by the English inventor Sir Samuel Morland. He then brilliantly devised a way of making such a calculator carry out repeated additions and subtractions to enable it to multiply and divide. To do this Leibniz invented the stepped reckoner—a long cylindrical gear wheel containing nine teeth of different lengths. It turned a smaller ordinary wheel, which, in turn, operated the remainder of the machinery to carry out the calculations. The small wheel was set by a pointer on a dial that indicated the number by which it was required to multiply or divide. It then took up a position along the multiplier wheel so that it was turned only by the same number of teeth on the multiplier wheel. Leibniz invented his calculator in 1694, but it did not work perfectly because of the lack of precision of the machined parts.

Leibniz made two other important contributions to the computer. The first was in 1666 when he put forward the idea that reasoning can be analyzed into separate elements placed in order. This is fundamental to the operation of the computer. The second contribution was made in 1679 when Leibniz perfected the theory of binary numbers—a number system using only the two symbols 0 and 1.

Practical calculators
Leibniz had predicted that calculating machines would be taken up and used generally, but it took a long time for this to become a reality. The first calculating machine to achieve commercial success was the Arithmometer introduced by the French inventor Charles X. Thomas De Colmar in 1820. It was essentially an improved version of Leibniz’s calculator and contained his stepped wheel. However, it was not until about 1860 that the Arithmometer began to sell in large quantities. Advances thereafter were made principally in the United States. The engineer George Barnard Grant made a compact Rack and Pinion Calculator. Then in 1875, Frank Baldwin invented a multiplier wheel that consisted of a wheel containing nine pins that could move in or out to vary the number of teeth. Dorr E. Felt developed a calculator in which the action of pressing keys provided all the power needed to operate the calculator. His first model was constructed in 1885 and was called the Comptometer. It was successful, as too was William S. Burroughs’ Adding and Listing Machine in 1888. This machine not only added figures but printed them all and the total on a strip of paper.

The father of the computer
These calculating machines were not really computers. They had ways of feeding in numbers and obtaining results, and contained calculating mechanisms that performed the necessary arithmetic. These are equivalent to the input, output, and processing units of a computer. Other essential features were missing, however. The machines had no memory and could not store temporary results. Furthermore, the mechanisms could only perform arithmetic and could not be programmed to undertake another task.

The man who conceived of the computer was the British inventor Charles Babbage. Babbage was a mathematician as well as inventor, and made great use of logarithms for the calculations necessary for his work. The tables then available contained errors, and Babbage became increasingly irritated by the time spent finding and correcting them. In 1812 he conceived the idea of a calculating machine that could compile accurate tables of logarithms. He designed his machine to work on the principle known as the method of differences. By 1821 he had produced a pilot model that demonstrated the
THE METHOD OF DIFFERENCES

The logarithm tables that Babbage proposed to compile with his Difference Engine (see inset right and enlarged detail) are given by complex algebraic equations. Calculations of this type are simplified by using the method of differences. Consider a table of cubes of integers. The first five numbers (the cubes of 1, 2, 3, 4, and 5) are 1, 8, 27, 64, and 125. If you calculate the differences between these numbers you get 7 (8 - 1), 19 (27 - 8), 37 (64 - 27), and 61 (125 - 64). These are the first differences. The second differences are 12, 18, and 24 (see below). The third differences all come to 6, and this is true of the entire series of cubes no matter how far it extends. Knowing only the value of the first term in each series of differences enables the whole series of cubes to be calculated solely by addition. Each term is simply the sum of the previous term and the next difference. In logarithm tables, the difference approaches a constant as the number of differences increases. Calculating logarithms to a high number of differences produces very accurate results.

<table>
<thead>
<tr>
<th>No.</th>
<th>Cube</th>
<th>1st difference</th>
<th>2nd difference</th>
<th>3rd difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>19</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>37</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>61</td>
<td>30</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>91</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>343</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
validity of his ideas. In 1823 the British government gave Babbage the first of many grants, and the construction of the Difference Engine began.

Basically, the Difference Engine was a vast collection of intermeshing gears, which had to be machined to extremely fine tolerances, finer than contemporary engineering could produce. The poor precision in the gears accumulated so that the Engine suffered from vicious backlash and vibration, even after many redesigns. The government continued to support Babbage for ten years, spending a vast sum of money, and then withdrew. The Engine came to a standstill. There was nothing, however, wrong with its basic design, and in other hands with better engineering, it did succeed.

In 1834, a year after the suspension of the Engine, a Swedish engineer, George Scheutz, came across an account of it and determined to build one. Scheutz redesigned parts of it and succeeded in making his own engine by 1840. This was improved and later given aid by the Swedish government.

Babbage may have been bloodied at this outcome but he remained unbowed. As he wrestled with the complexities of the Difference Engine, an even more ambitious scheme entered his head. If a machine could be constructed to solve the equations necessary to compile logarithms, then it should be possible to build an all-purpose computing machine that could be applied to many problems. Its internal mechanism could be designed so that it would function in different ways depending on the type of problem that it was required to solve. At this point, in about 1833, the idea of the computer was born. As a practical proposition, it was even further ahead of its time than the Difference Engine and would take a century to achieve.

Babbage called his new machine the Analytical Engine. To feed data into the Engine and give it the instructions needed to direct it to undertake a particular kind of calculation – in effect, a program – he
proposed to use punched cards of the type devised by French inventor Joseph Jacquard in 1801. The cards were originally designed to control the operation of looms to weave patterns.

The calculating operations were to be carried out by an arithmetic mechanism Babbage termed the “mill”, and which contained a control unit to make it follow the instructions. Temporary results were to be transferred to the “store”, which was to consist of a thousand columns, each with fifty wheels. In the Analytical Engine we see the basic features of the computer. It had input and output units to enter data and program instructions and to print results; it had a processing unit in the mill complete with a control unit to direct it to follow a program; and in the store it had a memory unit.

Only parts of this prototype computer were ever built, however. The full machine lay far beyond the engineering techniques of the day. Nobody took up its challenge as others had with the Difference Engine, and in 1871 Babbage died, having fathered a magnificent concept.

Into the twentieth century

Although no progress was made toward the computer for just over a century, advances were made in other directions. Babbage’s was to be a digital computer; it would have printed out its results as columns of numbers. In electronic form, this is the kind of computer used today. But there is another type – the analog computer. It does not produce results as numbers, but as some quantity that represents a number. The kinds of geared mechanisms that Babbage proposed to use were ideal for the development of the analog computer because they could be used to display results in the form of a graph. The first analog machine of this type was built by the British scientist Lord Kelvin and his brother James Thomas in 1872. It was used solely to predict the rise and fall of tides and draw a graph representing their heights.

Kelvin foresaw that an analog computer could be built that would solve any type of problem, but because of technical problems he never attempted to design or construct one. But he did give it a name – the differential analyzer – so-called because it would be able to solve any differential equation. The first machine of this type was built in America by Vannevar Bush in 1930.

Meanwhile, progress in another field important to computing had been taking place. This was the handling of information to obtain meaningful results from it – what we call data processing. It was brought about by the demands of analyzing the massive amounts of information generated by the census counts in America in the late 1800s. A better method than hand compilation of recording and counting was needed, and a competition was held to test proposals for new systems. It was won by Herman Hollerith and his tabulating machine. It was selected to undertake the 1890 census and Hollerith was able to announce that the population of the United States stood at just over 62 million people only six weeks after census day. The previous census of 1880 had taken seven years to count. On the basis of his success, Hollerith founded a company that in 1924 became the International Business Machine Corporation or IBM, now the biggest computer company in the world.

The first electrical computers

The United States was now in a real position to make a breakthrough. IBM had the technological resources and Bush had inspired important theoretical developments. His student Claude Shannon worked out the electrical circuits required to carry out binary arithmetic in 1937, and Shannon’s colleague George Stibitz constructed the first working

Felvin’s Tide-Predicting Machine

This machine was built by J. White in 1872 to the specifications of Lord Kelvin and his brother. Kelvin’s machine predicted the ebb and flow of the tides, producing graphs to show these fluctuations. Quantification in a form that represents numerical values, as here, is the prime characteristic of analog calculation.

The Hollerith tabulator

The results of the 1890 US census were computed by the American inventor Herman Hollerith, using his punched-card tabulator.
models using electromagnetic relays. Stibitz realized that these circuits could form the basis of a programmable computer, and drew up a design for one. This was never built, but in 1940 he built a test unit called the Complex Number Computer that could work in binary.

Under the direction of Howard Aiken, IBM was pressing ahead with a full-scale computer. It was to be a digital machine working in decimal, and relays were to perform the computing work. Work began in 1939 and was completed in 1944 at Harvard University. The machine was called the Harvard Mark I. The computing of results was slow, but the Mark I can be described as the world's first full-scale digital computer and was, as Aiken rightly claimed, Babbage's dream finally come true.

Back in 1938 a German engineer called Konrad Zuse had produced at his home a test unit with the features of a digital binary computer, called the Z1. It had a keyboard for input, a set of switches acted as a memory, and a row of lights gave results in binary. Zuse then proceeded to the Z2, which contained relays and used punched film for input. The Z3 and Z4 followed during World War II and they were electro-mechanical machines used in aircraft design. All of these machines were destroyed during the hostilities.

The first electronic computers
Electronic tubes (valves) had been invented at the turn of the century. Tubes have the ability to switch an electric current on and off quickly and could, therefore, be used in computers to handle data coded into electrical pulses. Aiken was aware of this when he designed the Harvard Mark I, but had opted for relays because they were more reliable, if slow. Tubes consumed a lot of power and tended to overheat and break down. Also aware of these factors was the British government. Scientist Alan Turing and others set about building a computer that could crack enemy code messages. Their machines, built from 1941 onward, had relays.
These were replaced in 1943 by the Colossus series of tube computers. In total secrecy these machines rapidly broke the German codes.

Colossus was the first electronic digital computer. It could perform only one task — code cracking — and was unable to solve other problems. While Colossus was at work in Britain, another electronic computer began to take shape in America. It was the brainchild of John W. Mauchly and J. Presper Eckert. Mauchly and Eckert set out to build a tube computer for possible use in ballistic calculations and it was not completed until 1946. The computer was called ENIAC, short for Electronic Numerical Integrator And Computer. ENIAC was astoundingly fast and could add a five-digit number 5000 times a second using decimal and not binary arithmetic. This was slightly faster than Colossus, and though it had little memory, ENIAC could be given a new program to do a different kind of calculation. The process was laborious, involving replugging and rewiring, but ENIAC can be called the first general-purpose electronic digital computer and was in service for ten years.

**Computer development to the present**

The British and the Americans were now leading the world in computer development and both countries kept pace in their achievements over the next few years. However, two vital steps in the evolution of the computer took place in the USA.

The first was made by mathematician John von Neumann, who saw immediately that the difficulty in reprogramming ENIAC greatly reduced its powers. Von Neumann came up with the solution. He suggested that the memory of a computer could be developed so that a program could be stored in it in the form of coded instructions. In this way, the computer could have instant access to the program and interact with it if necessary. The distinction, though, of being the first computer to run a stored program was gained by the Manchester Mark I, a British computer that began to operate in 1948. It was followed the next year by EDSAC (Electronic Delay Storage Automatic Calculator), built at Cambridge, England by a team led by Maurice Wilkes.

These early computers were developed for scientific and defence purposes, but the commercial field beckoned to many in addition to Mauchly and Eckert. Their new computer, BINAC, made in 1946, used magnetic tape input, and it was cheaper and faster than ENIAC. The two men then developed the first commercial computer called UNIVAC I for Sperry Rand in 1951. It had a hundred times the capacity of ENIAC, was ten times as fast, and almost a tenth of its size.

At this point the United States began to pull ahead in computer development. IBM began to dominate the computer business with the 700 series of tube computers that first appeared in 1952. These machines were expensive and complex to

---

**COMPUTER THEORY**

The theory behind the operation of computers has been important to their design. Following Leibniz's investigation of binary numbers and of the nature of reasoning, the process of logic was analyzed by the British mathematician George Boole in the mid-1800s. Self-taught, yet a professor of mathematics, Boole showed that logic could be treated mathematically, using symbols like algebraic notation to denote operations such as "if" and "or". Using his method, which is known as Boolean algebra, the logical outcome of a series of propositions could be proved. The British philosophers Alfred North Whitehead and Bertrand Russell developed the mathematics of logic, leading to the invention of electrical circuits that could carry out logical operations by comparing numbers as well as performing arithmetical operations.
Japanese keyboard While the use of word processors has spread rapidly in the West, countries which do not use our Roman alphabet face severe problems in introducing office automation. Japan and China, in particular, find it difficult to represent their characters electronically. An even bigger problem is typing these characters. One Japanese company’s answer is shown right: a keyboard with 12 characters per key. Mainland China, meanwhile, is encouraging the use of a Romanized alphabet.

operate, limiting the potential market for computers to the government departments and large companies that could afford them. If the computer was to fulfill its potential it somehow had to become much cheaper and simpler.

This barrier to the future development of computers was overcome by the second vital step mentioned earlier. This was the invention of the transistor, which occurred in the United States in 1948. It was the result of long research by John Bardeen, Walter Brattain, and William Shockley, who all shared the Nobel prize for their achievement. Transistors could do all that tubes could, but in a much smaller space and with very much less consumption of electrical energy.

Transistors did not enter computers right away. First they had to be developed so that their performance rivalled that of tubes. This happened in 1956 with the first transistorized computer – the TX-O (Transistor Experimental computer) at the Massachusetts Institute of Technology. This machine heralded a breakthrough in computer design and in four years all new computers were using transistors. At a stroke, their size and expense were reduced and their power and simplicity increased. In addition, memory improved in reliability and capacity with the development of magnetic tape and disk systems, making computers accessible to industries and businesses of all kinds.

The quantum jump in computer design brought about by the transistor was only the beginning of an ever-increasing process of miniaturization of computer parts and a consequent fall in price and a rise in power. If one component such as a circuit doing the work of a tube could be reduced to the small piece of semiconductor making up a transistor, so too could a whole group of components. This concept of the integrated circuit was first aired in 1952 but a working design was not achieved until 1958, when it was developed by Jack Kilby of Texas Instruments. Development subsequently took several years and the first integrated circuit computers began to appear in the United States in 1964. By this time, 30 components could be packed on to a “chip” only 5 millimeters square; ten years later the figure was a thousand times greater and it has continued to rise. Among the results of this microchip technology has been the development of the microcomputer. With this, the computer reached its fourth generation, the first being the tube computer, the second the transistor computer, and the third the integrated-circuit computer.

All these developments were made in the United States, as was Project MAC, the first system to interlink computers into a network. This occurred in the early 1960s, and was very important in extending the capabilities of computers. So too has been the development of computer languages to facilitate programming, again mainly in America. However as the fifth generation of computers grows ever nearer, the lead that America has enjoyed for so long may slip. This turnaround will be entirely attributable to the fact that one nation has made the achievement of the fifth generation computer virtually a national goal. That nation is Japan – and the target date it has set for itself is 1990.
Having taken the plunge and bought your first computer - now what? The presentation box your machine comes in, once opened, reveals a confusing collection of black boxes and cables that seem to snake out in all directions without any obvious purpose. The computer itself, which, after all, is what prompted the purchase, seems lost among it all.

Unpacking your computer
But take heart. Everything in the box is there for a purpose, albeit a dim one at the moment. Take all the various bits and pieces and spread them about on a table in front of you. Ideally, this table will be your workstation once you have connected everything up.

Very carefully check the accompanying documentation to make sure that everything that should be there is, in fact, accounted for. A typical checklist of equipment will include the following items:

1 The computer console itself. This, obviously, is the most important piece of equipment and may well have an integral keyboard.
2 Some kind of power supply. You cannot connect most types of computer directly to the wall socket. Instead, you have to go through an additional power supply unit. This unit will probably become quite warm after it has been on and working for some time, but unless it actually becomes hot to the touch, this is nothing to worry about.
3 Diagrams that come with practically all makes of computer will illustrate how you should connect your machine to the display - probably, in your case, your ordinary television set.
4 Cables. Assuming that everything is wired up safely, you can now connect your power supply unit to the power supply socket on the back panel of your computer, and plug the other end into the regular wall socket. Do not, however, switch on yet, as there are still a few things to cover.
5 An RF modulator. This device converts the video signal produced by the computer into one that can by displayed on a television set. Not all computers, however, will require one of these. Some machines produce a signal that can link directly to your television. One end of the RF modulator (if required) plugs into the back of the computer. This socket could be labeled either “RF-in” or “antenna”. The other end plugs into the antenna socket on the back of your television. Without this modulator, some machines would simply be incapable of producing a display at all. If instead of using a television set you have a monitor, sometimes called a Cathode Ray Tube (CRT) or Visual Display Unit (VDU), you will definitely not need an RF modulator. You should bear in mind, though, that many monitors (especially the cheaper ones) cannot produce a color image or reproduce sound. These are two of the most desirable features of most home computers, and being deprived of them could severely limit both the scope of the software usable and, more importantly, your enjoyment of using the computer.

If you can go to the expense of buying a sophisticated color and sound monitor, the actual picture quality produced by it will be far superior to that produced by a television. This is because the conversion process carried out by the RF modulator does result in a slightly degraded image.

The equipment above is usually all you will need to connect up your new computer, switch it on, and start to see the results of your efforts appearing on the screen in front of you. However, if you just do that, you are not going to get very far. Even if you are simply working through the examples of programs supplied with your user’s manual, every time you turn the machine off everything on the screen is going to be lost - wiped completely clean from the computer’s memory (see Module 19). In other words, you have as yet no way of “saving” anything you have typed in. To overcome this problem, all computers come complete with the ability to connect up (interface) with some sort of device for storing information that you have put into it. There are two different types of storage media. The most readily available is an ordinary cassette player, and for the vast majority of home applications this is perfectly satisfactory. The other common storage device is the computer magnetic disk, but for the present we will be concentrating only on the cassette.

Equipment compatibility
At this point you come across the first major division in makes of computers, and it is worth mentioning
ASSEMBLING YOUR COMPUTER

After removing the machine from its box the chances are that you will find yourself with a collection of components and a bundle of cables. Usually, the manual will tell you which cables go in which sockets, but it is not always easy to decipher some of the documentation. In fact, things are not nearly as complicated as they look. First, every item must have power supplied to it in some way. Many home computers have an external power pack that plugs into the computer and into the main power supply. But do not worry if your machine is not one of this particular type.

To allow the computer to display a picture on the television screen, you have to connect the video lead to the television antenna socket and the appropriate socket on the computer. The cassette recorder is also linked to the computer. But take care to connect the sockets marked EAR on both the tape and computer — and, likewise, the sockets marked MIC must be connected together.

The Apple Ile: The latest Apple, the Ile, has been designed to make setting up particularly easy. All the connectors have been grouped together at the back of the machine (see above right), out of harm’s way. One feature of the Apple is its expandability — a number of sockets are available inside the machine which allow you to add a wide range of plug-in boards, providing extra memory and input/output facilities. Across the back of the machine you will find a number of blank push-out panels (visible on the photograph above), which can be replaced by sockets for these extra ports. Previous models had a series of unprotected holes through which wires passed to plug directly into the internally located expansion boards.

The Spectrum: The Sinclair ZX Spectrum has been designed for easy setting up as well as ease of use. Like the Apple, all the connectors are grouped together at the back, but the machine does not have an internal expansion facility. Instead, a large connector at the rear allows extra devices, such as a printer or joysticks, to be added to the system. In fact, a very large number of signals generated by the computer are available at this connector.

Data output socket: Provides data output to the cassette and connects to the MIC socket.

Video connector: This provides an output to a monitor. An optional interface to a domestic television can be fitted.

Data input socket: This socket allows data from the cassette to be transferred to the computer, and it should be connected to the recorder’s EAR socket.

Power supply: There is an external power supply to reduce the main voltage to the 9 volts required by the computer.

MIC socket: This is the output from the computer to the cassette dock.

Input socket: Data is received by the computer through this socket. It must be linked to a similarly marked socket on the cassette.

Video modulator: This machine has an internal video modulator allowing it to be used with a television set.

Typical set-up: Here you can see a typical home computer system (with a few options) connected to the power supply and peripheral devices. You will not need a power pack if your machine has an internal power supply. Nor will you require an RF modulator if you use a monitor instead of an ordinary domestic television.
this fundamental difference. As far as tape-based storage is concerned, you can divide computers into two types - Commodore computers, and the rest.

For reasons of their own, the Commodore company made a decision to make computers that cannot interface to ordinary domestic cassette players (see Module 21). Instead, you have to buy a special player made only by Commodore. There is more information on dealing with this complication in Module 9 dealing with saving and loading programs and other sorts of data, but there is one point that concerns you now if you have bought one of these machines. If you have a Commodore computer, when you unpack it for the first time, the special tape deck has its own built-in cable that connects to the back of the computer console. With other makes, the cabling is loose so that you can use virtually any cassette player.

So that the tape recorder can “talk” to the computer, and the computer to the tape recorder, there are usually two cables (see previous page). One cable runs from the computer to the tape recorder, and one the other way round. All you need now do is plug them in (making sure that all connections are tight) and you are almost ready to start.

To recap then, so far you have connected the computer up to your television (or, possibly to a monitor), perhaps with the aid of an RF modulator, plugged the computer into the wall socket (via a power supply unit), and, for non-Commodore owners, connected up a tape recorder as well to act as your storage medium. Commodore owners can plug in their own, special tape recorder.

**Tuning in**

For those of you not using a monitor, tuning in your television set to obtain a good, crisp image may take a little time. Switch on your computer and television and then tune in your television to the channel recommended in your user’s manual. If you have a set with several spare channels, each with its own tuning knob, then it is obviously better to use one of these than a channel you would normally watch. Most home computers display some sort of copyright notice or logo when they are first switched on, and this is what you are looking for as you adjust the tuning knob.

Obtaining the initial copyright message on the screen is usually quite easy, but many computers only display a black and white picture at power-on, even if they have some color facilities. In order to make sure you have tuned in completely accurately, you should change color. On the now very popular Sinclair Spectrum, for example, hold down the CAPS SHIFT key (bottom left-hand corner) and then press the BORDER key (it is in the middle of the bottom row of keys). The word BORDER should appear on the screen. Now press the BLUE key (the number “1”) at the left-hand end of the top row. Finally, press the ENTER key, which you will find over on the right-hand side of the keyboard. If you have your television correctly tuned in, the border on the screen will turn blue. If, however, the border just goes black, you must keep readjusting the tuning knob until you can see a good deep blue color on the screen.

On other types of computer you will find different color commands to the ones described here – if you cannot immediately understand how these work, do not worry. You can either “load” (transfer from a storage medium to the computer’s memory) the demonstration tape that may have come with the computer, and if this does not produce a color image you can then tune in while the demonstration tape is running, or you can simply wait until you have found out a little more about your particular machine and have become more familiar with its color commands.

Incidentally, if your television has AFC (automatic frequency control) then you should turn this off while you are tuning in. And, it goes almost without saying that you can only receive a color image from your computer if you are connected up to a color television set. It is surprising how many people try to obtain a color image from a black and white television set.

If, after all this, you are still receiving a poor image, or you cannot see anything at all, then try tuning in to another channel until you find one that suits your computer. And if this still fails, there is probably something wrong with your machine and you will have to consult the store that sold it to you. This happening is, however, a very rare occurrence, so look for other possible causes first. Go over the computer’s back panel and make sure all leads are securely inserted – also the connection to the television may be loose.

As you survey your newly assembled computer, you probably see in front of you a maze of wires on the table top, and cables strewn about the floor just waiting for somebody to trip over them. And it is not only the person who falls over them that will be hurt. You will probably lose the program that you have just spent the last two hours typing in, and it will mean starting all over again. So it is well worth taking a little time out just to tidy everything away. Put cables where they will not get in the way, and clear all the wires so that you have a table, empty of all but the absolutely essential equipment, to work on.
Now that you have everything wired up correctly, and the computer is switched on and working, what do you do next? You are probably looking at the screen of your television that contains the computer's copyright message (and very little else) and wondering how to use this piece of equipment you have just bought.

**Getting to know your keyboard**

By now you have no doubt gathered that everything is controlled by the keyboard on the computer. Keyboards, like computers, can take many forms, but there are two main ones (with slight variations) that seem to dominate the home computer market at present. One of the most common ones, if only for the fact that it belongs to one of the biggest-selling home computers of all time, is that on the Sinclair ZX Spectrum.

The Spectrum has an extremely small keyboard, and although it is supposed to have the keys the correct distance apart, there are not too many typists who feel completely comfortable when using it. This type of keyboard is not "touch sensitive" - in other words you actually have to depress the keys before you see anything on the screen respond to the keyboard. Touch-sensitive boards are completely flat with the key names illustrated on the surface. Electrical contacts underneath transfer signals to the rest of the system. These keyboards are cheaper to manufacture but not as convenient to use.

A common feature of all small keyboards is what is known as the "single keystroke" method of entering data. "Data" is a word you are going to have to get used to - all computer people refer to words and information as data. The single keystroke entry in some ways compensates for the speed at which you can type on this sort of keyboard. By pressing just one or, more usually, two keys you can produce a whole word (or command) on the screen. These words are not ordinary English words, but are instead words that the computer can "understand" and translate for you.

Take the word PRINT, for example. It means and does exactly what you would expect it to - it prints something. Using the small Spectrum keyboard you can enter this word by pressing one key. On most other computers you would have to press five keys (P-R-I-N-T) to produce the same effect. However, for this particular word, because it is used so often, other makes of computer also have a shorthand method of entry.

So, on the small keyboard then, entering words that the computer can understand is usually very easy, but actual typing (when using, for example, your computer as a word processor - see Module 36) is rather slower.

On single keystroke entry boards you will notice that each key can contain as many as five or even six letters, symbols, or full words - rather like some calculators do. This is simply a way of reducing the physical size of the keyboard. This concentration of effects on to a single key can be irritating. Trying to type one thing, you discover that you are in the wrong "mode" for entering that type of information, and you have to play around with the other keys before you can proceed or you may have to go back and start all over again. After a while, though, you do become used to this system.

With most other machines, however, you are presented with a fairly orthodox typewriter-like keyboard, and any competent typist should feel immediately at home. Obviously, some of the names of the keys are going to be unique to your particular machine. Try the keyboard on your own computer. It does not matter if what you are entering is complete nonsense - you cannot in any way hurt the machine by doing this - just try to familiarize yourself with its layout and what the keys actually do.

**Using the individual keys**

Two keys that assume paramount importance when you start using your machine are the INSERT and DELETE keys. As you might expect, these are used to insert and delete spaces. Some computers have special-function keys, which, when you are more experienced, you can program to produce complicated or often-used commands at a single keystroke.

It is only by using the keyboard that you will become familiar with it, so here is a small program you can enter and watch the results. The language that you use to communicate to your computer with is, more than likely, called Basic (which stands for Beginner's All-purpose Symbolic Instruction Code). There are, unfortunately, slightly different
versions of Basic used on different machines, but the chances are very good that the following program will be compatible with your particular “dialect” of Basic. So, type in the following lines exactly as shown:

```
10 PRINT "WHAT IS YOUR NAME"
20 INPUT A$
30 PRINT "HELLO"; A$
40 FOR I = 1 TO 100
50 PRINT I$
60 NEXT I
```

What, then, are the numbers at the beginning of the lines? Computers are essentially “blank” machines, and can only follow your instructions if they are presented to it in the right order. The line numbers tell the computer this order. They go up in steps of ten in case you want to go back at some time and add extra lines between existing ones. This could be because you have made a mistake or you want to alter the program’s end result.

Line by line, this program asks you for your name, and then waits for you to type it in. When you have entered your name you must then press the ENTER (or RETURN) key before the computer can proceed – it cannot do anything without you telling it what to do. It will then print HELLO, followed by your name, and then proceeds to print your name on the screen one hundred times.

In order to enter this program you probably had to use keys that you had not touched before, and this next one will give you a chance to try a few more keys:

```
10 A$="I AM A COMPUTER"
20 PRINT A$
30 B$="AND YOU ARE ONLY A HUMAN"
40 PRINT B$
```

You can try altering both these programs yourself by simply changing some of the words or phrases. You will soon learn that programming at this level is not all that difficult.

Little programs like these, although instructive, do not really use the computer to its full potential. Later on in the book there are guidelines for producing more complicated programs (see Module 27). As was mentioned earlier, Basic comes in a variety of different forms, so guidelines are all that are possible. Although this standard Basic will be compatible with about 99 per cent of the home computer market, there is one noticeable exception – the Jupiter Ace. This computer uses an entirely different language called Forth (see Module 32), and typing in the above programs would have no effect at all – the computer would simply not understand a word of it.

**Using your computer’s color facilities**

As you are now a little familiar with programming and with the layout of the keyboard, you can start to look at some of the other things your computer can do. As you saw in Module 7, in order to obtain the best possible image on your television set it was necessary to tune the receiver. But most modern home computers also have the ability to display a wide range of colors, too. Computers usually display text or numbers on the screen in what is known as “reverse field” – white characters on a black screen. With the Spectrum, however, you start off with black characters on a white screen.

Find out from your user’s manual how to put your computer into reverse field (this differs from machine to machine) and press the SPACE key (sometimes this is a SPACE bar as on a normal typewriter), which is usually situated underneath all the other keys. You will then see a broad line appearing on the screen. To change the color of the characters again you must consult your user’s manual. On the Commodore 64, for example, you have to press the CONTROL key and one of the number keys. Pressing number 3 on the keyboard will give you a red character. (Pressing other numbers will produce different colors.) Continue pressing the SPACE bar and you will see a broad red line on the screen. Repeat the process for all possible colors and you should end up with a rainbow pattern on the screen. Continue experimenting until you feel completely familiar with the color commands governing this facility of your particular machine.

With all the colors on display, you can adjust your television set so that they all show to the best advantage. Try different settings for the contrast, brightness, and color density controls of your television until you find a combination you like. Once this is set up, you should not have to adjust it again, unless your television is one of those that seems incapable of holding a steady or consistent image for any length of time.

**Using your computer’s sound facilities**

Another popular feature of most modern makes of home computer is sound. This is one of the features that varies most from computer to computer, both in the range of sound offered and the way in which it is generated.

There are two methods of generating sound – one is by producing the sound from a speaker unit built
into the computer, and the other is by playing it through the speaker built into your television. As you might expect, the sound coming from your television's speaker is far superior because it is (usually) that much bigger and more sophisticated.

Many of the "arcade" type games available for home machines make extensive use of sound, and it can become irritating after you have played the game a few times. If the sound is coming from your television's speaker, you can always turn it off. The manner in which the sound is generated to some extent reflects on the way in which you can control it. To look at just two popular computers, "Beep" is the one and only command on the Spectrum for making a noise, while the Commodore 64 virtually has a synthesizer built into it.

But no matter what computer you have, try making a few experimental noises to become familiar with both the range of sounds possible and the commands needed to generate them.

Using your computer's graphics facilities

Another feature common to all microcomputers is the ability to display graphical characters of one sort or another. Most computers have their own set of characters built into the machine itself, allowing you to produce the entire graphics character set by pressing a few keys. Again, you will find that some machines have better facilities than others, but they are all easily accessible from the keyboard.

To write a program that would display all the characters for every computer would be impossible, but there is a way you can try it on your own machine. There is a "keyword" (a word that tells the computer what to do) on most machines called CHR$, which tells the computer that it is to display a character (as opposed to performing an arithmetic function of some sort). Every character, like the letters A to Z or the numbers 0 to 9, and any special graphics characters on your keyboard, can be printed with a CHR$ statement. To take one example, the letter upper case A is usually represented by the code 65. In other words, whenever you press the upper case A key, the machine reads this as code 65, which represents A, and so displays that character (see Module 13). So, if you type in PRINT CHR$(65) your computer will display the letter A. In this example it would have been easier simply to type the letter in the first place, but by using the keyword, followed by other numbers in brackets, you can start to discover the full graphics potential of your computer. Try them all and see what effects you can produce.

As with color and sound, experimenting with these graphics facilities is the only way to learn, and just to stress again, you cannot in any way damage your machine by doing this. The next step is to try to write a simple program using these extra facilities. Try to build sound, color, and graphics into all your programs until you are completely familiar with them, and your programs will be better for them being included.

By now you should be feeling that what was initially a miscellaneous collection of keys and wiring is, in fact, an extremely versatile tool and that you can, at least to a limited degree, exercise control over what appears on the screen. As time goes on, this control will become more refined and, therefore, require less effort on your part.

If all you could do with a computer is basically as described above, there would not really be much point in owning one. After all, every time you turned the computer off, everything you had entered would be lost. Consequently, there are a number of options open to you for making sure that what you type in can be saved and then recalled again and again whenever needed. In the next module you will see the ways available to you for saving programs, and how to connect up the various devices that enable you to load them back into the computer.
By far the most common method of saving what you have typed into the computer is the ordinary domestic cassette player. This is most often used because it is the cheapest of all methods and also because most people already own one.

The only real exception to this is if you are a Commodore owner. In this case, you will have to buy a special cassette deck made by Commodore. Commodore claim good reasons for this arrangement, chief among them being reliability. Being designed specifically for use with a particular computer, users should have very few problems. You cannot, however, use the Commodore cassette player with any other type of computer.

Cassette deck interfacing
Owners of most makes of computer do not have to go to the expense of buying a special cassette player or, at least, only need buy a cheap model for use with their computer. You might encounter some problems with very cheap decks, though, especially if your particular model has automatic tone and pitch control. Some computers can be fussy about these sound qualities and will refuse to accept a program unless the tone and pitch are just right. Another important consideration if you are buying a cassette specifically for use with a computer is compatibility of plugs and sockets. Make sure that the cassette player has the right number and right type of sockets to accept the cables supplied with your computer.

Using the cables supplied, connect the cassette to the computer as described in Module 7, and place a cassette tape in position. Any type of tape will be adequate, although you will find fewer problems if you use a good quality make. Special short-length cassette tapes are also available. These are ideal for storing a series of short programs and can save you much rewinding and searching time. Something to bear in mind is that you can only load a program at the normal playing speed for music. Any long program, such as some games programs, for example, can take a frustratingly long time to load.

Testing the interface
With the cassette deck connected up to the computer, and the tape in place and fully rewound, you will have to enter a short program to test everything is working the way it should. You can either write one yourself or use this example:

```
10 PRINT "THIS IS A TEST"
20 PRINT "FOR MY CASSETTE DECK"
30 GOTO 10
```

If, after entering this, you instruct your computer to RUN the program, all that will happen is that the two sentences THIS IS A TEST and FOR MY CASSETTE DECK will be printed over and over again because you have created what is known as a program "loop" (see Module 27). To prevent the message endlessly appearing on the screen you must stop the program executing by pressing the BREAK key. Some computers have a key marked STOP just for this purpose, and yet others have an ESCAPE key. They all, however, perform exactly the same function.

To save this program on tape, first think of a name for it – TEST PRG, for example. Then enter at the keyboard SAVE "TEST PRG" and press the ENTER or RETURN key. The computer will then tell you what to do, which is press the cassette's play and record buttons, just as you would if you were recording an ordinary record.

If your tape deck has a tape counter, make sure you take note of the number on the counter every time you save a program. This will save you a lot of time when looking for a specific program – a problem that becomes more acute the more program tapes you have. Using a tape deck with a computer is just the same as using it with a record player. You cannot start the tape at the program (or record track) you want – you have to wind the tape on until you come to it – unless you know the number on the counter. If you have not made a note of the exact position of the programs, all is not lost. Because you give each program a name, which must be in quotation marks, the computer can search through all your tapes until it finds that program name. This is a lengthy procedure, though, because it cannot use the fast forward control.

Going back to our program above, after a few seconds the tape will stop, and the computer will tell you that everything is OK. But do not be
The only way you can test whether your program has been successfully saved is to remove the program from the computer's memory. On most machines you do this by typing the command NEW. Do this, and then you will be able to see if the program was really recorded by typing in LOAD "TEST PRG" - the name you assigned to that program when you were saving it. On some machines, this load command may take another form. Do not forget at this point to wind the tape back to a point before the start of the program you want to load.

Follow the computer's instructions again, which will be to press the play button on the cassette deck, and then to wait until the computer tells you that it has finished loading. When it does you can type in the command LIST. This instructs the computer to display your program on the screen line by line, so that you can see if it is all there. If everything appears to be in order, then you can be sure that your cassette deck interface is working properly and you can go ahead and start to save longer programs with no worries.

If, on the other hand, the computer does not appear to do anything after you press the play button, stop the tape and rewind it, and then try again. This time, however, turn the volume up a little and experiment with different tone settings. After a few attempts you should find the right combination, and you can safely load and save programs.

As many manufacturers and independent stores supply prerecorded tapes for computers, you will want to have your tape deck working properly with your computer. The time spent making fine adjustments is well worth it. There are, however, more efficient ways of storing and loading programs than using a cassette recorder, and these are discussed in the following module.
USING DISKS AND CARTRIDGES

Disk drives are not yet as common as cassettes for home use because they are more expensive, but they do overcome all the limitations of tape-based information storage. First, disks operate at a much faster speed and have a greater storage capacity than tapes. There is also the added bonus that you do not have to remember the position of all the programs—the computer does it for you. This increased speed and efficiency is due to the fact that the disk spins at approximately 300 revolutions per minute and the information is not stored serially, but randomly. The use of the word “random” can give the wrong impression. All it means in this context is that the “read/write” head attached to each disk drive (see Module 22) can move over the disk’s surface, under computer control, and access information at any point. These features have established the disk as an essential part of any computer system used for business applications, and the trend is spreading to the home computer market as well.

Despite all the benefits offered by disk storage, they are simple to operate and use the same commands as cassette decks (see Module 9) to load and save programs, but with one important addition. Everything connected to the computer is assigned a device number or name so that the computer can differentiate between them all. So, in order to tell the computer that you want to save a program on disk rather than tape, you simply type a command such as SAVE “B:TEST PRG”. In this example B: is the device name for one of the disk drives attached to the computer, and TEST PRG is the name you have assigned to that particular program. The actual format and wording of these “save” commands will vary quite considerably between different machines, but they are all simple to use.

Using cartridges
An alternative to using disks as your storage medium is the cartridge (see Module 21). Cartridges slot into the back of suitable computers and, once they are in place, effectively “take over” the machine. Cartridges are nearly always used for loading programs into the computer rather than saving programs that you have written yourself, although some computers do allow you to save programs on cartridges. The types of programs written for cartridges are usually more sophisticated than the average home hobbyist could write, and the cost of manufacturing and packaging the cartridge all combine to make the cartridge relatively expensive. The system is, however, favored by the software manufacturers as, once a program is inside the cartridge, it cannot be copied.

Using the computer press
A very important source of program material that you should be aware of is the computer press. It does mean, however, you doing all the work, rather
than leaving it to the computer. There are now so many specialist computer magazines available that it is an easy task, once each week or month, to select the ones that contain the best program listings for your particular machine. Indeed, such has been the growth in this type of magazine that there is very probably an entire publication devoted to your own computer — especially if your is one of the more popular models. Once you have gathered them together, you will find there are more than enough listings for you to type in.

This is basically an easy, but time-consuming, job. With the listing to one side of you, you have to sit there and type in all the ones you want. Any complicated program is going to be long, containing hundreds of different program lines. Typing in a program of this length means you are inevitably going to make some mistakes along the way that you will have to correct at a later stage (a process known as “debugging”).

After you have been typing listings for a few hours, and before somebody pulls the plug on you and wastes all your effort, you must transfer the program either on to a tape or disk (see Modules 21 and 22). Always save the program before you try it out, in case a mistake you have made (or one already contained in the program) causes it to “crash” when you RUN it. Crashing is when a mistake somewhere in the program causes irretrievable loss of information.

Once you have finished typing the program in and have a copy of it saved, and you are convinced there are no mistakes that are going to cause any problems, then the fun begins.

The program that you are now looking at might be for a game or have a more serious application, such as your home accounts. Both magazines and books, however, also cater for the business users of microcomputers, so you will sometimes find applications programs covering things like databases (see Module 34) and spreadsheets (see Module 35). But no matter what the program is, once it is sitting in the machine (with a copy safely saved) you can start to modify it in any way you wish. Most are published in the full knowledge that you — the user — will want to personalize them. And once you are finally happy with it, it can be saved along with the original, effectively giving you two programs for every one published.

Try to modify as many programs as you can, learning from the original as you go. This will teach you more about the intricacies of programming than anything else ever can. And there is no reason why you should just restrict yourself to programs written specifically for your machine. Most home microcomputers use the same or similar dialects of Basic, so look at programs for other makes, and with a little time and patience you should be able to adapt them to run on your machine.
Once you have gained the initial confidence and feel thoroughly familiar with your computer, keyboard, and the commands necessary to control its various functions, writing programs is largely a question of adopting the right attitude.

Planning a program
When you finally set about writing programs for your own use, you must first of all decide what area of program writing you are going to concentrate on during any one particular writing session. You cannot, for example, start out with the idea of writing some sort of games program (see Module 29) and then, half-way through, suddenly decide that your time could be better spent by writing some sort of integrated accounts package (see Module 35). So, settle on one area and decide that for the next few hours, or however long you can spare at the keyboard, that this is what you are going to write.

Having done that, there are a number of other points that you must settle before actually starting to type in the code. Most importantly, you must have a good idea, preferably written down on paper, of precisely what you are going to do and how you are going to structure the program.

Using algorithms
Some people possibly can sit down at the keyboard and just type, modifying a program as they go along, with no preconceived idea of how the program is going to develop. People like that probably do produce reasonable programs that work but, inevitably, they could be better structured, could work faster, and could achieve more.

In order to produce this more efficient program, you must map out exactly what you want the program to do, and not be deterred when a section of it stubbornly refuses to work in quite the way you want it to. No doubt, you are doing something wrong, tackling the problem in the wrong way or from the wrong angle.

By mapping out beforehand everything that you want to do, the risk of this sort of problem occurring is lessened. The usual way of planning programs is by using a flowchart of its various aspects and how they interrelate — this is called an algorithm (see Module 26). If you were thinking of writing a games program you would have to put down more than “there will be forty aliens and one spaceship”. This tells you nothing about the program or how it will ultimately be put together.

Each section of the program should be written out thoroughly before sitting down at the computer. Using this same games example, you would need a routine to move the ship at the bottom of the screen from left to right, which takes into account the fact that you do not want the ship to move off either the left- or right-hand side of the screen. You will need to think how to write this routine — how to make the ship move and how to check that you are near the edge of the screen. If you just jump straight in without this preplanning, your program more than likely will not work, and even if by some chance it does work it will teach you absolutely nothing about programming.

Organizing a program
When you are in the habit of writing everything down beforehand you will soon realize that there is a lot more involved in the whole process than is at first apparent. As you gain more experience, you should begin to see that various parts of your current program are similar to other programs you have already written. So, rather than doing all the ground work time after time, you can substitute tried and tested routines, perhaps with just a little modification. This should teach you another lesson about program writing — always make sure that when you come back to a program after a length of time you know exactly what part of the program does what.

To facilitate this, home computers have a statement called REMARK (usually shortened to REM), which allows you to insert comments in your program. For example, at the start of your “move the ship to the left” routine, you could insert program lines like these:

```
100 REM THIS PART OF THE PROGRAM
1010 REM MOVES THE SHIP TO THE LEFT
```

In this way you will never be in any doubt as to what that section of the program does. Inserting these comments throughout listings, you will soon
be able to find your way through them again and spot any useful routines that could be adapted to a similar set of circumstances in another program.

This habit of putting comments in programs will not only help you, but also anybody else looking through your programs. You might, for example, submit a program to a magazine for possible publication, and it will certainly help your chances if the editor can read your program listing to see how it is constructed.

Using subroutines
One helpful program writing "trick" is to save all the common routines you find coming up again and again. Use a separate cassette tape or disk (depending on your storage system - see Modules 9 and 10), and before long you will find that you have a collection of useful routines, or, as they are more correctly called, subroutines. A subroutine is, essentially, a small program in its own right, which performs a specific function. For example, it might be a routine to format numbers to two decimal points. You have a numeric value called, say, A, equal to 12.7684239, which you want to look a little neater when you print it out on the screen. Your subroutine, then, will shorten A so that it equals 12.77.

In a lengthy program you may want to use this sort of routine often and it would become tedious typing it out again every time. This is the purpose of the subroutine, a short program that performs a useful function, and which performs that function many times within a major program. The Basic statement used to call up one of these subroutines is the command GOSUB. Using this transfers control of the program to the short subset in order that it performs whatever function is required. To return to the main flow of the program, you use the command RETURN.

By building up a set of routines such as this, your program writing time can be cut down quite considerably, and by planning everything out in advance you can also make considerable savings on development time. But where do all these routines come from? Obviously, most of them will be written by yourself, but there are also many publications available that detail programs of many different types. Magazines and books are there to help you in

your program writing, as is software made available through some kind of user group - a collection of individuals dedicated to the use of one type of computer. By their own efforts, they write, collate, and gather as much programming material as possible and, for a small fee, make it available to anybody who cares to join their user group.

As time goes by, and your programming efforts produce better and better results, you will ultimately build up routines for use in many areas of software development. A whole collection of financial routines, screen movement routines, and so on, will be there to be accommodated in programs as and when you require them.

Filing routine
As was mentioned before, all of these routines should, ideally, be stored on tape or disk for later use. Something you will learn almost as soon as you start developing (or simply start coming across) these programs is the need for an organized filing system that makes it easier to retrieve the right program at the right time.

An ideal piece of equipment for this purpose is a printer (see Module 23). Printers can be expensive, costing more than the computer itself, but they can
The variable ADD has been used to add two numbers together. As well as telling the computer what to do, it also indicates to you what is happening.

Here, however, you come across a problem – most computers cannot recognize more than the first two letters of a variable name. Although the computer will work happily with something called ADD, in memory (see Module 19) it will only be stored as AD. This is nothing to worry about unless you have other variables called, for example, ADDA.

Another potential danger area to watch out for is trying to incorporate Basic statements within your program names. You cannot use, for example, IFFY, PRINTER, or CATALOG because the first one contains the Basic statement IF, the second PRINT, and the third LOG. By using legitimate variable names in your subroutines, it becomes much easier to merge them together to form a coherent program that is simple to understand.

What sort of subroutines should you be using? There are many programming areas and many programming problems to be solved, but you will find the same kind of problem cropping up repeatedly. Inputting information into the computer is an area that has seen many routines published and it is certainly one of the first routines you should write.

Sorting routines is another common area, and there are many different types of sorting routines as there are computers. Each one works best in a different set of circumstances, so make sure you are using the right one for the job. If it takes a long time to sort something, you are probably using the wrong program, or you can increase the execution speed by perhaps altering it a little.

Never be worried about altering a program that you have come across in a magazine or perhaps one that has been given to you. Whoever wrote it first did so for themselves to work under a given set of circumstances. Just because it was correct for them does not make it so for you. And if you see the ideal routine for a program you are working on, but it has been written for a different machine, it is worth the effort trying to adapt it. Many dialects of Basic differ only slightly, so if you see a command like CLS, and look around your keyboard in vain for a key with the same letters, a little thought and a look at the program listing in full, will soon tell you that CLS stands for CLEAR SCREEN. All you need do then is substitute your machine’s code for this command. So it is with any program you come across. A little thought will usually tell you the way to correct everything. And that is how to go about program writing – define your application area, say to yourself that you are going to write that and nothing else, and make sure you follow a few simple rules.
Considering the electronic complexity of modern home computers, faults leading to a total system shut-down are surprisingly rare. If, however, you are unfortunate and something is the matter, the guide below should help you to solve the problem quickly. Although some of this advice may seem obvious, it is usually the obvious that does go wrong. This advice covers the majority of faults that might occur when initially setting up the computer for a programming session. If, at the end of all this, there is still something wrong, the next thing is to make sure your television or monitor is plugged in, or that the video cable is still in place, or that all fuses are working properly. If the system still refuses to work, turn your attention to any peripherals in use – the tape deck or disk drive, for example. Are they wired up correctly and connected to the right sockets? If the computer still fails to respond, you probably need professional assistance, so shut the whole system down.

**System crash**

Assuming that you have had everything running happily for some time, and then the system "crashes" (a term meaning that something serious has gone wrong with the computer, or some part of it), then you must adopt another set of procedures. The first thing to decide is whether the fault is yours, or the computer's, or in some peripheral attached to the computer. If you have perhaps been

**COMMON FAULTS**

If your computer will not respond in the way you think it should when starting a programming session, the chances are that something pretty minor has gone wrong (see some examples below). Modern micros are extremely reliable, so look for obvious causes of the problem like loose wires, disconnected cables, or blown fuses. Any problems with color or sound usually stem from your television or monitor, and not from the computer. Only after you have checked all external causes, and made sure that your software is not to blame, should you consult your dealer.

**Symptom** No picture on screen.
**Cause** A number of possibilities. The computer is not switched on, it is not plugged in to the power supply, or there is something wrong with the computer.
**Cure** Make sure the computer's power switch is on, that it is connected to the socket, and that you have switched it on. If it still does not work, look at the computer's fuse. This is usually located at the back of the machine, and can be reached with the aid of an ordinary screwdriver. If the fuse has blown, and you have to fit a replacement, make sure it is the right type. If in doubt, consult your dealer.

**Symptom** No picture of any sort on the screen although you know that the computer is turned on and that it is working properly.
**Cause** There are a number of possibilities. The television might not be tuned properly, the wires are not in the correct sockets, the modulator (if required) has not been plugged in, or the video cable connecting the computer to television is not in place. All of these possible causes are easily remedied, so check all connections very carefully.
**Cure** First, try turning the computer off for a few seconds, and then back on again. This sometimes clears any internal problems, although there is no logical reason for it working. Otherwise, check that the television is correctly tuned, that all the wires are in the right place, and the video cable is connected as it should be. Unless you tidy away wiring after connecting up, you are likely to be plagued by this type of minor, irritating problem.

**Symptom** Excessive background noise from the television or no sound at all.
**Cause** Poor adjustment of the television's volume control.
**Cure** This is an easy one – readjust the volume control on your television.

**Symptom** Picture without color or picture with poor color.
**Cause** The television set is not tuned in properly, or the color controls need adjusting.
**Cure** If your computer can only display black and white, there is not too much you can do. However, if it is capable of giving a color image, it is a simple matter of retuning the television set and checking all the colors with the test recommended in Module 7 (printing a lot of spaces in all the different colors available on your particular computer).
experimenting with machine code, or randomly POKEing values (see Module 27) into memory locations, the likelihood is that the fault is yours. Certain areas of the computer's memory are sensitive to random values being put into them. If you have been programming in machine code, then the possible causes of the crash are many. With all machine code programs there are a number of rules to follow. The principal one of these is always SAVE your program before attempting to run it. This is true of Basic programs as well.

One common cause of a machine code program crash is that you can become stuck in an endless loop somewhere, because you have forgotten to RETURN control to the main program.

**Problems with Basic**

If it is not machine code that has caused the crash, perhaps your Basic program is at fault. As usual, there are a number of rules you should follow in an attempt to track it down. You can usually get some idea from error messages produced by your computer. The most common of these, although the actual wording will vary from machine to machine, is SYNTAX ERROR IN XXX, where XXX is the line number containing the error. Other error messages are also helpful, and almost always give a clue to the likely cure. For example, if you see OUT OF DATA ERROR IN XXX, the computer is expecting to find some data in line XXX that is not there — so you must put it in. Or perhaps an error message might read NEXT WITHOUT FOR IN XXX. This will tell you that although you have told the computer to take the next step in a FOR . . . NEXT loop, you have not started the loop off.

If there is no error message at all, and the program just refuses to work, a useful tip is to include STOP (or BREAK) statements at different points in your program. If, immediately after a portion of program that refuses to work, you put in one of these statements, the program will come to a halt. You can then examine the variables being used, all the lines that have been executed, make a few changes, and try again. By the repeated use of this method you should, eventually, be able to solve most of your programming problems.

**Problems with the machine**

If you have not an obvious programming error in either Basic or machine code, then what else could have gone wrong? If you are convinced your program is working properly, or it is one that you have used successfully before, then it is reasonable to assume that the machine, or some part of it, is at fault rather than you. The most likely fault is that somewhere along the line a wire or cable has come loose and simply needs reconnecting. If this is the case then, unfortunately, you will probably have lost the program that you were working on or running at the time. But if you have taken the sensible precaution of making a back-up copy, nothing will be permanently lost.

If all the cables are in the right place, it is possible that one of the connecting parts of the cable has been slightly chipped. If may have been working loose for some time, and one last jolt caused it to snap off. This type of fault can be very difficult to spot, but if you do find a faulty connecting wire is causing the problem, the only solution is to buy another cable and remake the connection. If all the wires are in place and in good condition, the next thing to look for is fuses. These can blow at any time without any obvious sign, except everything grinds to a halt. As a precaution, it is always a good idea to keep spare fuses in the home. If you are a computer hobbyist, the chances are you spend most time with the computer in the evenings and weekends — and these are just the times when it is hardest to replace something simple like a fuse as most stores are shut.

If the problem is not the wires or the fuses, then are you trying to do the impossible? Are you, for example, trying to save a program or data on disk or tape, but using a device code that is not in use? This can happen very easily — all you need do is make a simple typing mistake. This type of error is more likely when you have been typing the same code so often that entering it has become automatic. If this happens, the computer will try to obey the command when, in fact, it cannot.

Having gone through all the likely and not-so-likely causes mentioned here, and still the system refuses to perform, look at everything one more time, and then call in professional help. If any of the internal components of the computer or peripherals are at fault, it is not recommended that you take the outer casing off to investigate. You are welcome to try, but if you make things worse then the fault is all yours, and most dealers will, quite rightly, refuse to look at a machine that has been tampered with in anyway at all.

To sum up, always check the obvious first. Wires do come loose, plugs can be pulled out of their sockets, and a line in your program that you are sure is right can contain just one small typing error, such as a missed bracket, that snowballs its way through the rest of the program causing chaos on the way. By taking sensible precautions, and always approaching any possible error in a calm, logical way, you should find that your computer gives you trouble-free service.
HOW COMPUTERS WORK
To the layman, a computer can be both fascinating and intimidating. It performs astounding feats at amazing speed, it provides answers to complex questions within fractions of a second, and, if the experts are to be believed, it never makes mistakes – any errors that do crop up are the fault of the machine minders.

We are all familiar with the science-fiction image of computers – they speak, hold conversations, and exhibit the human qualities of friendliness, cuteness, human sense, although terms like “artificial intelligence” (see Module 37) might give a contrary impression. It is possible to make a computer behave in a way that gives an imitation of limited thought, and technological advances are now being made that will push this impression much further. But a computer is only a machine – complex, ingenious, fast-working, and powerful, but still a machine.

Computer numbering systems
In order to solve problems, computers handle numbers and, being electronic devices, they need some method of representing them electrically. One way to do this would be to use different voltages to represent different numbers: 1 volt for “1”, 2 volts for “2”, and so on. There are, though, practical problems with this method, especially if you want to handle numbers like 100,000. Another problem would be providing accurate voltages to distinguish between, say, 10,001 and 10,002, and at the same time between 0.00002 and 0.00003.

An easier system would be to use the presence of an electric current to represent “1” and its absence to represent “0”. The obvious drawback is that we can only represent two quantities – “1” and “0” – and so a completely new system of number representation is necessary.

We are accustomed to using a numbering system based on 10, the decimal or denary system. Take for example the number 3057. Using 10 symbols – 0 to 9 – we arrange them in positions that give them added significance, the units, tens, hundreds, and thousands columns (see facing page). We describe the decimal system as being a numbering system to “base 10” as it is based around the quantity 10.

A numbering system using any base is possible. Suppose, for example, we wanted a numbering system based on 16. An immediate problem is that we need some extra symbols to represent decimal 10, 11, 12, 13, 14 and 15. We could use letters: A=10, B=11, C=12, D=13, E=14, and F=15. We would count just as we do in decimal: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, until we arrive at F in the units column. Adding “1” makes 16, so we put a “1” in the 16s column and reset the units column to zero. So “10” in this system represents the decimal number “16” and “2A” is decimal “42”. A base 16 numbering system is called “hexadecimal” and is, in fact, used in computer systems.

The binary system
To use only two symbols – 1 and 0 – we need a numbering system based on 2 – the “binary” system. As before, we can count by increasing the units until there is a carry, at which point we set the units column to zero and increase the 2s column by one. So, binary counting would be (with decimal in brackets) 0 (0), 1 (1), 10 (2), 11 (3), 100 (4), 101 (5), and so on (see facing page).

Binary, then, gives us a method of representing numbers with 1s and 0s, and binary notation is what is needed to carry counting into the world of electronics. To make binary a little clearer, it can be imagined as a series of lamps (see facing page) – a lit lamp representing “1” and an unlit lamp “0”.

There is little point, though, in devising a numbering system if it cannot be used for arithmetic. In fact binary arithmetic works in the same way as its denary equivalent, as you can see:

| 1001 | + | 9 |
| 101  |   | 5 |
| 1110 |   | 14 |

Bits and bytes
Each binary digit is referred to in computer jargon as a “bit”. One bit by itself does not represent very much, so they are arranged in groups of a useful size. In most microcomputers this unit is the “byte” – a group of eight bits. Microcomputer memory
DECIMAL AND HEXADECIMAL

Our normal counting system is based round the quantity 10, and is known as the decimal or denary system. We use symbols - 0 to 9 and to represent quantities larger than 9 we alter their positions. In the example right, the number 3057 is arranged in its four columns to show how the decimal system works. It is possible, however, to use other bases for numbering. The hexadecimal system, for example, uses 16 as its base. Quantities 10, 11, 12, 13, 14, and 15 are represented by the letters A, B, C, D, E, and F. Counting in hex (the usual abbreviation for hexadecimal) is the same as in decimal. The quantity in the units column is increased until it reaches F (decimal 15); adding 1 to this means that the amount in the next column (the 16s column) can be increased and units reset to 0.

BINARY

Binary notation is, quite simply, a numbering system to base 2, and you can represent any number using it. Because it is to base 2, it requires two symbols only - 1 and 0. These can be represented conveniently in the computer by the presence or absence of an electric current (see below). Counting in binary is carried out in exactly the same way as with decimal and hexadecimal above. The only apparent difference is that because you only have two symbols to manipulate you need more columns in order to represent most numbers. The columns with binary are: units, 2s, 4s, 8s, 16s, 32s, 64s, and so on (see right). So starting off with zero, and adding 1 each time, counting in binary goes like this (with decimal in brackets): 0 (1), 1 (2), 10 (3), 110 (4), 101 (5), 110 (6) (for a more extensive list see right).

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
</tr>
</tbody>
</table>

The use of an electric current to count in binary is illustrated here. A lamp switched off equals 0 and a lamp switched on equals 1.

By switching on the lamp in the units column, and keeping the others switched off, the quantity 1 is represented.

In this example, reading the lamps from right to left, you should be able to work out that the number represented is 11.
capacity is measured in the number of bytes of information it can store. It is done using binary groups of 1024 bytes, called kilobytes, or kbytes, or simply k. The common maximum memory size for many micros is 64 kbytes, an actual total of 65,536 bytes. Occasionally you will hear mention of “nibbles”. Simply, a nibble is a group of four bits, or half a byte, but it is not a widely used term. Until now we have been talking about computers dealing with numbers, as though this is all they did. We know, of course, that they have to handle characters as well, so that they can display, store, and manipulate text – a major use of computers today, in fact.

The ASCII system

How, then, are letters of the alphabet represented in binary? Simply, by giving each one a number that can be represented in binary. To enable computers to exchange information, a standard code has been adopted – the American Standard Code for Information Interchange, or ASCII (pronounced “ass-key”). As you can see from the table on the right, ASCII codes exist for upper and lower case letters, punctuation, and various symbols and numbers.

Two questions people ask after being introduced to ASCII are: how does the computer distinguish between printable ASCII characters and binary numbers, and why is the eighth bit of each ASCII byte not used to give double the number of possible characters? The answer to the first question is that the computer cannot distinguish – it is up to the programmer to make sure that the computer does not try to add “234” to “Z”. As to the second question, the eighth bit is used in some micros to provide another range of characters, usually graphics characters. However, there is no recognized standard for these, so a program written for one machine will probably not work on another.

**Binary to Hex**

Binary is difficult to read when looking for a discrepancy in the middle of a mass of numbers (as in the left-hand columns below). By dividing each byte into two nibbles, the first byte in the list would be 1101 and 0001. A nibble can represent quantities from 0 to 15, and as hex can show quantities up to 15 in one digit, a byte could be written in two digits (right-hand columns). Any discrepancy is then obvious.

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Character</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010100</td>
<td>07</td>
<td>BEL</td>
<td>Sending this code to the terminal sounds the “bell” (usually an electronic beeper these days).</td>
</tr>
<tr>
<td>00100000</td>
<td>08</td>
<td>BS</td>
<td>The backspace code moves the cursor back one character position on the screen.</td>
</tr>
<tr>
<td>00010001</td>
<td>09</td>
<td>HT</td>
<td>The horizontal tab signal works like a typewriter’s tab – the cursor jumps forward to the next tab position. Usually, tab positions are preset every eight columns but are alterable on many machines.</td>
</tr>
<tr>
<td>00010100</td>
<td>0A</td>
<td>FF</td>
<td>The line feed signal moves the paper up one line. On a monitor it moves the display up one line.</td>
</tr>
<tr>
<td>00100000</td>
<td>0C</td>
<td>ESC</td>
<td>Form feed on a printer advances the paper to the start of a new sheet. On many monitors it clears the screen and puts the cursor at the top left-hand corner – the electronic equivalent of a fresh sheet of paper.</td>
</tr>
<tr>
<td>00010110</td>
<td>0D</td>
<td>CR</td>
<td>The carriage return moves the print head to the start of a new line when sent to a printer. On a screen it moves the cursor to the start of a line.</td>
</tr>
<tr>
<td>00110000</td>
<td>0F</td>
<td>LF</td>
<td>The delete code – like escape – is usually generated by a special key, and it deletes the last character typed.</td>
</tr>
</tbody>
</table>

THE ASCII TABLE

The first 32 ASCII codes are, mostly, not printable characters and are given code names. The majority are used for communications work, but a few (see smaller table) have more general applications. CR, for example, means carriage return and sends the “cursor” on the screen back to the start of the line. HT tabs it along to the right, LF moves the cursor down a line, and FF blanks the screen. Many of the characters can be generated from the keyboard by pressing the “control” key and the appropriate letter simultaneously, thereby producing the “control code” shown in the table by ^A or ^B, for example. Being an American invention, ASCII has no code for currency symbols other than $; the ¢ symbol is often used instead, and non-American computers/printers “translate” this into the correct symbol.
<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Hex Character</th>
<th>ASCII Code</th>
<th>Hex Character</th>
<th>ASCII Code</th>
<th>Hex Character</th>
<th>ASCII Code</th>
<th>Hex Character</th>
<th>ASCII Code</th>
<th>Hex Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00 NUL ^@</td>
<td>032</td>
<td>20 (space)</td>
<td>064</td>
<td>40 @</td>
<td>096</td>
<td>60 '</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>01 SOH ^A</td>
<td>033</td>
<td>21 !</td>
<td>065</td>
<td>41 A</td>
<td>097</td>
<td>61 a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>02 STX ^B</td>
<td>034</td>
<td>22 &quot;</td>
<td>066</td>
<td>42 B</td>
<td>098</td>
<td>62 b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>03 ETX ^C</td>
<td>035</td>
<td>23 #</td>
<td>067</td>
<td>43 C</td>
<td>099</td>
<td>63 c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004</td>
<td>04 EOT ^D</td>
<td>036</td>
<td>24 $</td>
<td>068</td>
<td>44 D</td>
<td>100</td>
<td>64 d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>05 ENQ ^E</td>
<td>037</td>
<td>25 %</td>
<td>069</td>
<td>45 E</td>
<td>101</td>
<td>65 e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td>06 ACK ^F</td>
<td>038</td>
<td>26 &amp;</td>
<td>070</td>
<td>46 F</td>
<td>102</td>
<td>66 f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td>07 BEL ^G</td>
<td>039</td>
<td>27 '</td>
<td>071</td>
<td>47 G</td>
<td>103</td>
<td>67 g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>08 BS ^H</td>
<td>040</td>
<td>28 (</td>
<td>072</td>
<td>48 H</td>
<td>104</td>
<td>68 h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td>09 HT ^I</td>
<td>041</td>
<td>29 )</td>
<td>073</td>
<td>49 I</td>
<td>105</td>
<td>69 i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>0A LF ^J</td>
<td>042</td>
<td>2A *</td>
<td>074</td>
<td>4A J</td>
<td>106</td>
<td>6A j</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>0B VT ^K</td>
<td>043</td>
<td>2B +</td>
<td>075</td>
<td>4B K</td>
<td>107</td>
<td>6B k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>012</td>
<td>0C FF ^L</td>
<td>044</td>
<td>2C ,</td>
<td>076</td>
<td>4C L</td>
<td>108</td>
<td>6C l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>013</td>
<td>0D CR ^M</td>
<td>045</td>
<td>2D -</td>
<td>077</td>
<td>4D M</td>
<td>109</td>
<td>6D m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td>0E SO ^N</td>
<td>046</td>
<td>2E .</td>
<td>078</td>
<td>4E N</td>
<td>110</td>
<td>6E n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>0F SI ^O</td>
<td>047</td>
<td>2F /</td>
<td>079</td>
<td>4F O</td>
<td>111</td>
<td>6F o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>10 DLE ^P</td>
<td>048</td>
<td>30 0</td>
<td>080</td>
<td>50 P</td>
<td>112</td>
<td>70 p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017</td>
<td>11 DC1 ^Q</td>
<td>049</td>
<td>31 1</td>
<td>081</td>
<td>51 Q</td>
<td>113</td>
<td>71 q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>018</td>
<td>12 DC2 ^R</td>
<td>050</td>
<td>32 2</td>
<td>082</td>
<td>52 R</td>
<td>114</td>
<td>72 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>019</td>
<td>13 DC3 ^S</td>
<td>051</td>
<td>33 3</td>
<td>083</td>
<td>53 S</td>
<td>115</td>
<td>73 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>14 DC4 ^T</td>
<td>052</td>
<td>34 4</td>
<td>084</td>
<td>54 T</td>
<td>116</td>
<td>74 t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>021</td>
<td>15 NAK ^U</td>
<td>053</td>
<td>35 5</td>
<td>085</td>
<td>55 U</td>
<td>117</td>
<td>75 u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>022</td>
<td>16 SYN ^V</td>
<td>054</td>
<td>36 6</td>
<td>086</td>
<td>56 V</td>
<td>118</td>
<td>76 v</td>
<td></td>
<td></td>
</tr>
<tr>
<td>023</td>
<td>17 ETB ^W</td>
<td>055</td>
<td>37 7</td>
<td>087</td>
<td>57 W</td>
<td>119</td>
<td>77 w</td>
<td></td>
<td></td>
</tr>
<tr>
<td>024</td>
<td>18 CAN ^X</td>
<td>056</td>
<td>38 8</td>
<td>088</td>
<td>58 X</td>
<td>120</td>
<td>78 x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>025</td>
<td>19 EM ^Y</td>
<td>057</td>
<td>39 9</td>
<td>089</td>
<td>59 Y</td>
<td>121</td>
<td>79 y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>1A SUB ^Z</td>
<td>058</td>
<td>3A :</td>
<td>090</td>
<td>5A Z</td>
<td>122</td>
<td>7A z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>027</td>
<td>1B ESC ^[</td>
<td>059</td>
<td>3B ;</td>
<td>091</td>
<td>5B ]</td>
<td>123</td>
<td>7B {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>028</td>
<td>1C FS ^/</td>
<td>060</td>
<td>3C &lt;</td>
<td>092</td>
<td>5C /</td>
<td>124</td>
<td>7C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>029</td>
<td>1D GS ^]</td>
<td>061</td>
<td>3D =</td>
<td>093</td>
<td>5D ]</td>
<td>125</td>
<td>7D }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>030</td>
<td>1E RS ^^</td>
<td>062</td>
<td>3E &gt;</td>
<td>094</td>
<td>5E ^</td>
<td>126</td>
<td>7E ~</td>
<td></td>
<td></td>
</tr>
<tr>
<td>031</td>
<td>1F US ^~</td>
<td>063</td>
<td>3F ?</td>
<td>095</td>
<td>5F —</td>
<td>127</td>
<td>7F (delete)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As you saw in Module 13, for a computer to handle letters and numbers, a special system – the binary system – is needed. The binary system uses a series of 1s and 0s to represent this information electronically, but up to this point there has been no explanation of exactly what “handles” means, or how the computer manages it.

The ALU
When you look at a computer program (see Module 27) you will begin to see that many of the instructions involve the computer comparing the binary strings of information making up the characters typed in via the keyboard. For example, part of a computer program might involve checking a series of dates being fed in by system users. In order to validate this simple piece of information the computer might be instructed to compare the length of the entry to make sure it has exactly the right number of bytes (as determined by the programmer). Next it will have to ensure that any punctuation within that date is in the correct position by comparing sections of the date to a standard already in the program. Then, again according to the particular program, if the last two characters representing the year fall within a certain range then that whole unit of information might be shunted to a particular memory location and added to or subtracted from information already contained there.

The part of the processor that performs this comparing function is the Arithmetic and Logic Unit (ALU). When the program is running, data is moved from memory to the ALU where it is manipulated before being put back into a memory location or moved to the output screen or printer.

Truth tables
To understand how the ALU performs these logic tasks, it is necessary to look at the mechanics of the operation. Suppose you have two bits, A and B, and you want to know whether they are both “1”. It would be convenient if you could perform some sort of operation on them, which, without altering them at all, would result in a third bit – a “1” if both are “1”, and a “0” if they are not. In computer logic terminology this is called a truth table (see chart above right). This particular operation is called AND – if A AND B are both 1, then the result is 1, otherwise it is 0. Suppose, however, that you want a 0 as the result of this operation rather than 1. The truth table will be different (see chart above) and you can see that it is exactly opposite to AND – it is NOT AND in fact, a name that is abbreviated to NAND. Other types of logic functions are perfectly possible as well. For example, if you need to know whether A or B or both are 1 you can draw up another truth table (see facing page). This function is called OR – if A OR B OR both are 1, the result is 1, otherwise it is 0. Like AND, OR has its opposite – NOR. Finally it would be useful to know whether A or B is 1, and this function is called EXCLUSIVE OR, abbreviated to either EOR or XOR.

Applying logic functions
Through the application of logic functions the computer is capable of creating new bits of information; rather than being a passive accumulator of facts and figures typed in through the keyboard, it can compare, manipulate, and come to conclusions regarding its input. As an example, if you take the binary number 01100001 – 61 hex or ASCII “a”,...
How computers work · Module 14

OR The OR function tells us whether one or the other or both inputs are 1 by giving 1 as a result. If both inputs are 0, then the result will be 0.

NOR Just as NAND is the opposite of AND, so NOR is the opposite of OR. It outputs a 0 if either or both inputs are 1, and a 1 if both are 0.

XOR The EXCLUSIVE OR (XOR) operation produces a 1 if only one of the inputs is 1, and a 0 if both inputs are the same.

and then AND it with the number 11011111, the result is:

\[
\begin{align*}
\text{AND} & \quad 01100001 \\
\text{Result} & \quad 01000001
\end{align*}
\]

If you check the result in the ASCII table (see Module 13), you will find that 01000001 is 41 hex or ASCII “A”. The result of this operation is to change a lower case ASCII character into its upper case equivalent. As another example you can use the XOR function:

\[
\begin{align*}
\text{XOR} & \quad 01000001 \\
\text{Result} & \quad 01100001
\end{align*}
\]

The result of this particular function is to change an upper case “A” into its lower case equivalent.

To carry out the logic operations in the two examples above, take them a bit at a time. Working right to left, regard the top bit as A and the one beneath as B, and then consult the relevant truth table to see what happens to each particular combination of bits. Comparing groups of binary codes—two strings of letters, for example—is something that computers spend a lot of time doing.

In order to spot discrepancies between two groups of numbers, the computer can perform logic operations on each result byte to determine whether or not it is zero:

\[
\begin{align*}
11010001 & \quad \text{XOR} & \quad 11010001 & \quad \text{gives} & \quad 00000000 \\
11010011 & \quad \text{XOR} & \quad 11010011 & \quad \text{gives} & \quad 00000000 \\
01110110 & \quad \text{XOR} & \quad 01110110 & \quad \text{gives} & \quad 00000000 \\
10001010 & \quad \text{XOR} & \quad 10001010 & \quad \text{gives} & \quad 00000000 \\
11000101 & \quad \text{XOR} & \quad 11000101 & \quad \text{gives} & \quad 00000000
\end{align*}
\]

XORing each pair of numbers makes the discrepancy immediately obvious in line 4.

Logical shift

One operation that could be used to determine the discrepancy in the example above is the “logical shift” operation. Imagine that after completing this operation, you have your result byte held in a special register with room for an extra bit at one end. If you can manage to “insert” a bit at the opposite end (a 0, in fact), and “push” the other eight bits along one space, you can create an overflow into the extra location, which you can then test to determine whether this “carry” bit is 0 (see below). By repeating this operation eight times you can test every bit in the byte to discover whether the result is 0.

Although these types of logic functions seem very complicated, they really involve carrying out a number of simple operations very quickly. And this is part of the “magic” of computers—the amazing speed with which apparently complicated tasks are being executed. But all that is happening is that the computer is performing a few very simple operations over and over again at high speed.

The result of this particular function is to change an upper case “A” into its lower case equivalent.
How computers work • Module 15

ELECTRONIC
LOGIC

As you now know, computers store numbers and text using binary notation, taking a byte per character and splitting numbers into a series of bytes. Using certain logic functions, such as AND, OR, and XOR (see Module 14), the computer is capable of manipulating, or changing the location of, these bytes. But in order to perform these functions, special circuits are needed—one for each operation.

From relays to chips

When you were looking at binary notation (see Module 13), it became obvious that simple switches and lights could be used to represent binary numbers—with “1” being indicated by a lit bulb, and “0” by an unlit bulb. Counting in binary numbers using this manual arrangement would be very tedious and would necessitate many switch movements. It would be much easier to have an arrangement that involved just one manually operated switch—a pushbutton, for example—with all other bulbs turned on or off automatically. All that is needed is some sort of electronic switch—one that is operated by an electric current instead of a mechanical lever.

One or two very early computers used electromechanical relays to achieve this sort of effect. The solution was far from ideal, though, as relays tend to be temperamental, use a lot of electricity, and operate quite slowly. Electronic switches were next used, first tubes and later transistors. Certain types of transistors can be made to operate in this way and these are now produced as integrated circuits or chips, consisting of large numbers of transistors in complex networks that perform the equivalent of the logic functions you saw in the previous module.

Logic gates

System designers do not have to worry about how all these transistors are wired up—that task is taken care of by the chip manufacturers. The designer’s concern is with the resulting individual logic functions provided by each chip. And these functions, or “logic gates”, to give them their proper name, are represented by a series of symbols, as shown in the box below left.

The NAND gate is actually one of the most useful of all the functions and can be used as a building block to form all other logic functions. For example, if the same signal is applied to both inputs of a NAND gate you obtain the circuit below.

The NAND gate has now become an “inverter”, and can be used to create an AND gate out of two NAND gates.
An OR gate can be created by wiring up three NAND gates, but the first pair of gates must have one of their inputs wired to a positive voltage supply in order to force them to the logic “1” state.

If you can follow this through, using the NAND truth table and comparing the results with the OR truth table in the previous module, you will be able to see that to create a NOR gate all you need do is invert the output of the OR gate.

The circuitry for EXCLUSIVE OR, or XOR, is probably the most complicated.

You do not, however, have to draw collections of NAND gates every time in order to represent one of these other logic functions. Special chips exist to provide OR and XOR, and special symbols are used to represent them.
their chips using a schematic diagram. The diagram shows which pins connect to which gates, and which ones are inputs and which are outputs.

**Using logic gates**

It is beyond the scope of this book to go into all the practical aspects of constructing these circuits, first because there is a number of logic tutor kits available, and also because it is the subject for an entire book in itself. But experimenting on paper is possible, and much cheaper, too. The circuit right uses two NAND gates and an XOR gate, and is known as a “half adder”.

As you can see, the circuit has two inputs – A and B. But now there are two outputs as well. The XOR gate performs normally, outputting a 1 if either A or B is 1, otherwise the output is 0. But the first NAND gate will output a 0 if both A and B are 1 (and a 1 otherwise), which is inverted by the second
NAND gate to give a 1. The resulting truth table looks like this:

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Output</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Carry</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In effect the circuit adds A and B together. If you regard the output signal as the units column of a binary number and the carry signal as the 2s column, you are performing a simple sum:

\[
\begin{array}{c}
A + 1 \\
B + 1 \\
\text{Result 10}
\end{array}
\]

This circuit type is called a “half” adder because although it can generate a carry, it cannot take as its input the carry of a preceding half adder. In order to perform proper addition, taking carries into account, you need a slightly more complex circuit—the “full adder”.

Follow this circuit through very carefully, referring to the truth tables for NAND and XOR gates, and you will see it provides a truth table like this:

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C-in</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C-out</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Full adders can be “cascaded”, with the carry-out (C-out) of one being fed into the carry-in (C-in) of the next, allowing you to add entire binary numbers. The diagram below shows four full adders. As each adder can only handle one bit, this arrangement of adders can only deal with nibbles (four bits) of information at a time. To add two bytes together you would have to double the cascade. As they are full adders, there is always a carry-out signal at the end of the cascade. This can be passed through additional circuitry to tell you that the two numbers you have added produce a result that is too big to fit into a single byte. Numbers of this size would have to be split into two or even more bytes.
PUTTING THE BITS TOGETHER

By now you should be beginning to see how logic gates can be arranged in different combinations to perform a variety of operations on data and to control the content of data output (see Modules 14 and 15). The next step is to apply this to the actual computer in practice.

Flip-flops
The basic building block of all logic circuits is the NAND gate. It would be possible to construct an entire computer using NAND gates but, for convenience and economy, there are many, more specialized chips that can be used as an alternative. An example of one of these specialized circuits is the flip-flop (see figure 1 below). This is particularly useful because it allows you to store a bit that you can come back to later. Put eight of these together in parallel and you can store an entire byte. In fact, the flip-flop is a type of memory and a computer's RAM is basically just a massive collection of flip-flops.

Figure 2 shows how a one-byte memory cell can be built. By placing data on the input lines 1 to 8 and then switching on the clock line by sending a "1"

Data

Clock

Figure 1 In a flip-flop, a pulse on the "clock" input changes the output to the condition of the "data in" signal; with the clock turned off, the output will be held at the level of the input.

Figure 2 Eight flip-flops in parallel are able to store one byte. The byte is applied at the "input" lines and the clock signal is pulsed, transferring the byte into the flip-flops.

Figure 3 The flip-flop's output forms one input to the AND gate (see above), while the other input is the control signal. Turning it on will make the AND's output reflect that coming from the flip-flop.

Figure 4 AND gates and inverters can be combined to decode signals. If both control lines are 0 no device is selected; a 0 and 1 will select device 1; 1 and 0 selects 2; and 1 and 1 selects 3 (see below).
By combining the principles outlined in the preceding four diagrams, you can construct a simple way of storing data and retrieving it on demand. Here there are three memory cells. By sending a binary address — 01, 10, or 11 — along the address lines, you can select any of the three cells. Sending a 1 on the write line clocks the selected group of flip-flops so that they copy whatever data is on the data bus at the time. By selecting a group of flip-flops and then sending a 1 on the read line, you allow the contents of the selected cell to move on to the data bus. This, therefore, forms the basis of a computer's memory. The CPU can write data to and read data from any selected memory location. Most microcomputers have at least 16 address lines, allowing them to access thousands of memory cells. A typical microcomputer system involves a number of basic components, interconnected by three "buses", which carry various signals and information around the system. Because signals from the CPU are not powerful enough to drive all the system components, they must first be amplified by buffers. Each unit in the system then decodes the address bus and control lines with its own circuitry (not shown) and is only activated when the appropriate combination of signals is issued by the CPU.

Figure 5

<table>
<thead>
<tr>
<th>Address</th>
<th>Read</th>
<th>Cell 1</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>From databus</td>
<td>To data bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From databus</td>
<td>To data bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From databus</td>
<td>To data bus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cell 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cell 3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
<tr>
<td>From databus</td>
<td>To data bus</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From data bus</th>
<th>To data bus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
along it for a brief period, you can load the byte into the memory cell. There is a problem with this arrangement, however, apart from the fact that the circuit shown can hold only one byte. When you load the value into the memory it immediately appears at the outputs – inconvenient if you do not want to use it immediately. What is needed, therefore, is a way to isolate the outputs from the rest of the system until you actually want to read the contents of the memory cell.

This problem can be solved by using an AND gate as a switch. By placing an AND gate in the flip-flop’s output, all you then need is a “read” signal to the AND gate’s other input and you have a way of switching off the output until it is needed. Figure 3 shows how this is done. You can control the memory cell with two signals – one to allow data to be written to it (the write control), and one to read data from it. This demonstrates an important aspect of memory operation: the contents of a memory cell are only altered by a write operation. Each time the read signal allows the cell’s contents out on to the data bus, you receive a copy only of the cell’s contents – the contents are not altered in any way by the read operation.

A memory system that can store one byte only is not of much practical use. Modern computers require thousands – sometimes millions – of memory cells. There is a need, therefore, for some way of switching individual cells on and off for both reading and writing operations. One way of doing this is with AND gates, and a simple three-way switch is shown in figure 4. The basis of the system is the address bus (in this case, two wires), along which a binary number can be sent corresponding to the number of the device to be switched. Two lines allow four binary combinations – 00, 01, 10, and 11. As you can use the 00 signal to indicate that no device is required, this arrangement would allow you to control up to three devices. Inverters are placed across the inputs of two of the AND gates to ensure that only one gate at a time receives the 11 signal that switches it on.

Having devised a way of addressing each individual memory cell and of controlling both its input and output, it is possible to construct a working memory system. Figure 5 describes a three-byte memory system, and it shows how the address, read, and write signals act to allow data to be put into a particular memory location or to be read from it. In a computer system, the read, write, and address signals are all supplied by the CPU.

The CPU, although vastly more complicated, uses the same principles. For example, the CPU’s registers are simply memory cells, either 8 or 16 bits wide, within the CPU chip itself where their contents can be accessed more quickly than by referring to cells in the main memory. Data in these internal registers is moved around within the CPU by exactly the same system of gates as illustrated in figure 5. Also, the various parts of the CPU’s arithmetic and logic unit are switched in and out of action by such gates. The action is complex and is carried out under the supervision of the CPU’s
control unit, which interprets the incoming binary instructions into the pattern of signals required to switch on the various sections of the CPU required by that operation. It also sends switching signals to the AND gates, allowing data to flow through these circuits to and from the registers and/or data bus, as required.

It should now be apparent that although computers are complicated devices, there is actually nothing difficult in the way they work or in the way they are put together — their seeming complexity masks the underlying simplicity of the way in which they work.

**Disk-based computer system**
The trend in modern microcomputers is to cram more functions into each chip and to develop even more complex chips. The industry has now reached
the point where “intelligent” disk controller chips are available. These are capable of performing operations that only a few years ago required whole boards full of components. Performing the more complex mathematical operations, such as division, sines, and logarithms, is very time-consuming, so special math processor chips have been developed. The CPU sends two numbers to the math chip, together with a code signifying what is to be done with them, and the chip does the actual computing. This leaves the CPU free to continue with other things until the math chip signals that it has completed this particular task.

Once considerations such as what CPU is to be used, how much memory is required, and whether graphics, color, input and output ports, and disk drives are necessary, designing a new system today means selecting as many “intelligent” controller chips as are necessary and almost literally wiring them together.

To give you some idea of how the different components in a microcomputer are linked together, figure 6 shows a block diagram of a typical disk-based business system. The diagram has been simplified – not all the circuitry for the RAM and I/O ports has been included, for example. Neither have all the power supplies been included – each chip must be supplied with power (usually standardized at 5 volts in a microcomputer) and connected to ground, but showing all these extra lines would make the diagram too difficult to read.

At the center of the system is, of course, the CPU. The system clock generates a continuous stream of pulses that cause operations within the CPU to be carried out at a regular pace, in the right sequence, and at the right times. Some other components, especially the video circuitry and UART (see Module 20), might also be connected to this clock or be equipped with their own clocks.

You have seen that the CPU generates the address of the memory locations or I/O ports it wants to work with and sends out and receives data from these components. These addresses are distributed through the system by the address bus, which is a series of connections (usually 16 for an 8-bit system) to which the address decoding circuitry for each component is linked. Likewise, there is a data bus (8 bits wide for an 8-bit system) that carries data between the CPU and the rest of the system. The CPU also generates and receives various control signals, which are carried around by the control bus. In figure 6 only the basic ones – read, write, and two to show whether or not the read or write operation is concerned with memory or I/O – have been illustrated to keep things as simple as possible.

The signals that emerge from the CPU are not particularly strong – they might be capable of powering a small number of logic gates only, but in any but the crudest of systems there will be far more logic gates attached to the address, data, and control buses. So the first thing to be done to the signals when they emerge from the CPU is to give them enough power to “drive” all the necessary logic gates. This is the function of the buffers you can see next to the CPU – they simply boost the signals to avoid overloading the CPU’s internal circuitry.

This system incorporates a small amount of ROM that contains a piece of software called a “bootstrap loader”. When the computer is first switched on, this circuit arranges for the CPU to fetch its first instructions from this ROM. On a typical system, these instructions will set up a few basic operating procedures and then arrange for the operating system to be loaded into RAM from one of the disk drives. Once this has been done, the loader instructs the CPU to execute the operating system software in RAM, and from then on ROM is ignored.

Basically, the rest of the system consists of a number of I/O ports to link the computer to a printer or monitor, for example. The most complex component here is the disk controller chip, which usually appears to the system as several I/O ports – some for control and one or two for data. Also, the display interface might appear as one or several I/O ports, depending on its type.

The keyboard is controlled by its own chip that works out which key has been pressed and translates this into the correct ASCII code. The CPU monitors the keyboard’s I/O port whenever its program tells it to expect keyboard input and waits until a character has been typed. The ASCII codes are then usually transferred, one at a time, to a reserved area of memory until the user types in a carriage return to indicate the end of the input.

All business machines and many home models have a facility that allows you to connect the computer to a printer. There are two common methods of connection – the Centronics parallel and the RS232 serial interfaces. Usually, both interfaces are provided to allow the user to connect to whichever port happens to match that of the printer. In the system illustrated in figure 6, the RS232 interface is provided by the UART and its associated circuitry, which converts the UART’s output to the RS232 voltage levels. Most recent UART’s provide two independent ports, so the system has a second RS232 interface, which can be used to connect to, say, a modem (see Module 24) to transfer data over the telephone system either from or to another similarly equipped computer system.
Nobody is absolutely certain where the terms "hardware" and "software" originated but, with the spreading of computer awareness and knowledge, they seem to be gaining in popularity outside the computing sphere.

**What is hardware?**

Hardware is used to describe the computer itself and all the extra electronic and mechanical devices that may be linked to it to make up the complete system – screen, keyboard, disk drives or cassette player, and printer, for example. When most people think of a computer they think of the hardware. Yet, this is only part of the story, for without software the hardware is useless. If you are buying a computer, especially if it is for business use, the most important aspect is the software (see Module 25). If you want a particular job done, then you must ensure that there is software available (or somebody who can write the software). Only when you have satisfied yourself on this point should you start looking at computer hardware on which to run the software. And the type of software that will run on any particular computer, and the way in which it is treated, depends on the CPU.

**The CPU**

At the heart of every computer system is the CPU, or Central Processing Unit. This is the “brain” of the computer; the part where all the hard work is done. In large computers, the CPU is a closet-sized cabinet crammed with electronic components; in a personal micro it is a single component – known as a microprocessor.

In the previous section you saw how logic gates can be linked together to form arithmetic functions. Inside the microprocessor, this is extended to the point where a number of interconnected gates can be “rearranged” electronically to make them perform different tasks.

The photograph (right) shows what the microprocessor looks like before it is enclosed in its protective plastic packaging. Initially, it appears to be a random jumble of lines, but a closer look reveals some definite features. First, notice the small squares all round the edge of the chip. These are connection pads and before the chip is enclosed in

---

**THE ZILOG Z80 AND Z8000**

The 8-bit Zilog Z80 (right) is one of the most widely used microprocessor chips, and is a direct descendant of the Intel 8080, for which the popular CP/M operating system was written (see Module 33). The Z80 has a large number of registers – or RAM memory cells – which allow the programmer to perform many tasks more quickly by operating on the data held in the registers instead of shifting to and from main memory locations. The Z8000 (far right) is Zilog’s first 16-bit processor. It differs from other CPUs in that it does not use machine code to convert incoming binary instructions into CPU understandable signals. Instead it is “hard wired” – instructions are translated directly by logic circuitry.

**Z80 registers**

Register layout of the Z80 is shown below. The registers themselves are given arbitrary labels (B, C, D, E, H, L). These are duplicated for extra speed and power in the alternative register set.

The flags indicate the results of various operations taking place in the ALU. The final block is a set of special-purpose registers and these are essential for assembler programming (see Module 31).

**Main register set**

<table>
<thead>
<tr>
<th>Accumulator</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

**Alternative register set**

<table>
<thead>
<tr>
<th>Accumulator</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

**Special-purpose registers**

<table>
<thead>
<tr>
<th>Index register X</th>
<th>Index register Y</th>
<th>Program counter</th>
<th>Stack pointer</th>
<th>Interrupt vector</th>
<th>Memory refresh</th>
</tr>
</thead>
</table>
Internal "architecture" Features of the Z8000 below are (1) flags, (2) the ALU, (3) the registers, (4) logic circuitry, (5) and (6) bus controls. These areas handle the movement of instructions and data between the CPU and the rest of the system. The registers diagram shows the layout of the Z8002. The 16-bit registers (RO-R14) can be linked in pairs (RRO-RR14) to handle 32-bit numbers. These, in turn, can be paired (RQ0-RQ12) to form large 64-bit registers.
plastic, fine gold wires are welded to these to connect to the pins on the package. Through these, data, memory addresses, and control signals can be passed to and from the rest of the system.

Various areas of the chip perform different functions. There is an area composed of fairly regular, geometric patterns. This is, in fact, the area of ROM and forms part of the control unit—instructions coming in along the “data bus” are translated by the control unit into the signals that re-configure all the logic gates to enable the instructions to be carried out. At the heart of the CPU’s operations is the ALU (see Module 14), which performs the actual computing work. “Registers”, each made up of a few bytes of RAM, are used to store data that the ALU needs to carry out its instructions.

How the CPU works

For a fuller understanding of how the CPU performs its functions, it is possible to follow an instruction through the system and watch the sequence of events. Suppose we want to extract two numbers from memory, add them together, and store the result in a third memory location. The diagram (below) shows schematically how the various parts of the CPU are connected. The connections are called “buses”, and carry instructions and data round the chip. Although the explanation of what goes on inside the CPU may seem complicated, it is, in fact, only a simplified description. For example, the CPU cannot carry out these operations by itself—there must be a program stored somewhere in memory that instructs it to do all these things. At each stage, the CPU must receive the next instruction in the program, so a special register is used, called the program counter. This keeps track of where the computer must look in memory for its next instruction. The program counter is set up with a value corresponding to the memory location holding the first step in the program. Each time a new instruction is required, the counter is increased and its contents sent out along the address bus to tell the memory chips to send the instruction at that particular memory address to the CPU (via the data bus). It is possible to change the contents of the program counter by including the appropriate instruction in the program, and this is one of the key features of computers—you can divert or alter the computer’s progress through the program under program control. Generally, this is done in response to some condition arising from a previous operation. If the result of your addition, for example, was greater than a certain quantity, you could jump to another part of memory and continue to execute the program from that point. If the result failed the test you could continue with the next instruction.

CPU SCHEMATIC

The three units on the left obtain instructions from the computer’s memory (via the data bus) and turn them into signals used to activate parts of the CPU. The registers (B, C, D, E, H, L) are small areas of RAM used to hold data required by the CPU.

The program counter (PC) sends out (via the address bus) the location in main memory of the next instruction needed by the CPU. The flags register holds a number of 1-bit values, the condition of which is affected by operations taking place in the ALU. The accumulator is where data is stored before and after being processed by the ALU.

Data bus—carries instructions and information to and from memory.

Address bus—signals which memory location is needed.

Control bus—carries signals describing the current operation (memory read or write).
MOS TECH 6502

The 6502, manufactured by MOS Technology, is a popular 8-bit processor chip used in Commodore, Apple, and Atari computers. Although not as powerful in some respects as the Z80, it still incorporates some very useful programming facilities. Various versions of the 6502 have been produced, described by some as refinements and by others as complications. Its main disadvantage is that it is short on registers (see below). This tends to make the chip very memory-intensive and, hence, relatively slow in operation. Variations of the Z80, for example, can run up to three times faster than the fastest 6502. The instruction set of the 6502 is completely different to that of the Z80 and cannot, therefore, use Z80 – or CP/M – software. MOS Technology has, however, introduced special components that result in systems with overall fewer chips, leading to less expensive computers.

INTEL 8088 AND 8086

Internally, these 16-bit processors are virtually identical, with the same registers and instruction sets. The 8088 (below far right) has an 8-bit data bus connecting it to the outside world. This means that it can only shift data to and from memory a byte at a time, although internally it operates on 16-bit units of data. The 8086, with its 16-bit data bus, avoids any bottlenecks shunting information around the system, but, as a result, costs more. All the registers have a 16-bit capacity.
“Chips” are often referred to without really explaining what they are and how they are made. At first sight, these ingenious devices may appear very disappointing—a slab of black plastic with a row of connector pins along each of its larger sides. The actual chip itself is buried inside this package and cannot be seen, except in the case of EPROMs (see Module 19), which are fitted with a small window to allow you to erase their contents using an ultraviolet light. Even with an EPROM all that you can see is a tiny silver-colored square, which, if you look closely, has minute patterns engraved on its surface.

Making the chip
Chip manufacture begins with a logic design. This design is in the form of a diagrammatic representation of all the “gates” needed and how they should be connected to allow the chip to perform its assigned tasks. Very simply, a gate is a digital circuit that only functions when it receives a certain type of input. With complex chips, the internal design is worked out with the aid of computers. The gates themselves are constructed from the junctions of differently treated layers of silicon. In the next step of manufacture, the logic diagram has to be translated into a series of “masks”, which you can think of as being similar to photographic negatives, each one representing one of these layers. Again, extensive use of computer-aided design techniques is made, both for speed and accuracy. The masks are drawn as large, table-sized designs that are then photographically reduced to the size of the finished chip.

In fact, chip manufacture involves preparing a series of each of the layers of masks, and progressively building up a large number of chips at a time on a single disk of silicon. The process is complex and involves treating the silicon with photosensitive chemicals, exposing it to light through each layer of masks in turn, with intermediate stages to wash away unexposed parts of the disk’s surface, and repeating the process until all the chips on the disk are complete (see right).

There is a constant battle to cram more and more of these gates on to a chip, and manufacturers are now forced to use techniques involving electron beams instead of light, because the fineness of the
designs results in lines that are thinner than the wavelength of light.

One seemingly obvious solution to this space problem is to use larger sized pieces of silicon for the more complex chip designs. In fact this turns out not to be the answer. There is an optimum chip size and exceeding it results in fewer chips produced at a time, with consequent price increases. A smaller than optimum chip size gives less space in which to place logic gates.

The large disks of silicon, on which dozens of chips are made simultaneously, are then cut up into individual chips and each one tested. Each chip is mounted in a special computerized machine to test its electronic functions. In the early stages of a new chip design's life a lot will fail these tests, and even a well-established design will continue to produce failures, perhaps caused by specks of dust or other impurities creeping into the process. The yield rate is an important factor in chip production and is one reason why a new chip, especially a complex CPU, starts off with a very high price and becomes cheaper as a higher percentage of working chips emerge from the factory.

Chips that pass the tests are then wired to a series of connector pins with fine gold wire and embedded in plastic or ceramic "packages" to protect them. The finished, packaged chips are given a final testing before being shipped to customers. It is interesting that despite the high level of automated test procedures, it is impossible to test every aspect of a complex CPU chip. Running through the chip's entire instruction set (with every possible data combination) would take a computer several years.

Chip design This process starts with a logic diagram (see far right). This is then translated into patterns making up the masks from which the chip is made. The masks are made many times larger than the final chip to ensure the greatest possible accuracy. After careful checking (see right), they are photographically copied and reduced to the size of the final chip.
CHIP MANUFACTURE

Making a chip begins with a disk of very pure, highly polished silicon. Because the smallest speck of dust can ruin a chip, the entire manufacturing process is carried on in an atmosphere many times cleaner than that of a hospital operating theater. All staff wear hats, gowns, and masks to avoid accidentally contaminating the environment.

First of all the disk of silicon is heated in an atmosphere of pure oxygen to allow a layer of silicon dioxide to form on its surface.

The surface is then coated with photoresist. This substance hardens on exposure to ultraviolet light making it resistant to certain solvents.

The disk is exposed to ultraviolet light, shining through a mask of photographically produced film containing a pattern of conductors. Washing removes any resist that has not been exposed, leaving a copy of the mask's pattern on the disk.

An acid bath eats into all silicon dioxide left exposed by the removal of the resist. Another process removes the hardened resist.

Other layers are "doped" to alter their electrical properties. Washing away unexposed areas of chemicals forms other connections.

Finally, a layer of aluminum is deposited on the silicon's surface to form a connector between parts of each chip. The chips are cut up and tested.
As you will see in Module 27, the computer can do nothing unless it is given a list of instructions called a “program”, which tells it, step by step, exactly what to do. These instructions must be in binary code so that they can be represented within the computer as a series of 1s and 0s.

To be read and used by the computer a program must be stored in such a way that the computer can get at it. In other words, all those 1s and 0s need to be sent via electrical connections to the CPU (see Module 17). It is therefore necessary to keep the program in a memory of some kind. Most programs deal with information of one sort or another, either typed in by the user or generated by the program itself or both. This information must also be kept in a memory location.

The simplest way to think of memory is to envisage a large wall covered with pigeon holes. Each pigeon hole can hold one “unit” of information (in most microcomputers this is a byte) and each hole is numbered, starting at zero. Imagine that each pigeon hole is connected, via a series of logic gates (see Module 14), to the system’s address and data buses. To find out what is in a particular hole, the CPU sends out the number or address of that hole on the address bus. The logic gates connected to each hole are so arranged that when, and only when, the address of that hole appears on the address bus,

MEMORY AND THE CPU

In the computer below you can see the CPU (1), which performs all the computing functions. This machine has its operating system and a Basic interpreter housed in a single ROM chip (2). When the machine is turned on the CPU automatically takes instructions from ROM. Any program is held in RAM (3) and, when the RUN command is given, the interpreter translates the program into machine code which the CPU “understands”.

Memory in operation. These illustrations show in simplified form how the CPU interacts with RAM and ROM. The CPU reads data in a RAM location (top) by sending its address on the address bus; special circuits in the RAM area use this to activate the individual RAM cell that the CPU wants to read. The CPU also sends a control signal telling the RAM cell that it wants to read, rather than write data into memory. The write operation (center) is similar, except that the CPU sends a write control signal. In the bottom illustration the ROM cell is being read. Because you can only read what is in ROM and cannot alter its contents, there is no need for a write control signal. But the ROM still needs the read signal as well as the address to activate it.
they cause a copy of the hole’s contents to move on to the data bus and travel to the CPU. Note that only a copy goes to the CPU, and the hole’s original contents remain unaffected by this “read” operation.

When the CPU wants to “write” data to a memory location a similar process takes place – it puts the memory address on the address bus and the data on the data bus. The appropriate logic gates are activated by the address, and the data is placed in the memory location concerned, replacing anything that already exists there.

RAM and ROM

Because the CPU can read from or write to any location directly, this sort of memory is called Random Access Memory, or RAM. There is not however anything random about the way the CPU uses RAM – the term is used solely to differentiate this from the older styles of memory in which access was made serially, with the CPU starting at location zero and working through all locations until it found the one it wanted. There is a limit to the amount of RAM that a CPU can address and this limit is imposed by the number of wires in the address bus. Most 8-bit CPUs have a 16-wire address bus. This means that the largest number of memory locations that can be addressed is \(2^{16}\) (because the address is sent as a 16-bit binary number) or 65,536. This memory capacity is generally referred to as 64 kbytes, as each location can hold a single byte. Newer 16-bit processors can address much larger memories, typically several megabytes because there are more address lines emerging from the CPU.

The chips from which RAM is constructed suffer from one disadvantage – as soon as you turn off the power their contents are completely lost. There is an obvious advantage in having some way of giving the CPU instructions when it is first turned on, even if they simply tell it to load further software from disk into RAM. The usual way of achieving this is with a special type of memory called Read Only Memory, or ROM. You can think of ROM as a special type of pigeon hole with a glass front. You can look inside at the contents – which were put there when the pigeon holes were built – but you cannot alter them. Reading them is carried out in the same way as with RAM.
The most commonly used device for communicating with the computer is the keyboard. At the keyboard the user types in commands or responses to requests for information from the computer. Some basic home micros use touch-sensitive keyboards, but these are not really suitable for anything more than playing games or entering quite short programs. Attempting any business applications would be extremely tedious. More sophisticated home machines, and all business computers, use traditional typewriter-style keyboards, which often incorporate a few extra keys. These keys make certain computing operations simpler and perform other functions not found on typewriters. Many micros also have special keys called "programmable function keys", which can be set up to issue complex commands to the computer at a single keystroke. These are most useful with often-used commands.

Typical business keyboard This photograph shows a typical top-quality keyboard for business or home use. Categories and functions of keys are annotated in the diagram below.
Maltron keyboard

The QWERTY (first six keys on a typewriter) keyboard was originally designed to slow down typists, as early typewriters could not function as quickly as people could type. The QWERTY layout has, however, persisted, although several attempts have been made to introduce more efficient designs. One such attempt is the Maltron keyboard shown here. It allows potentially much faster typing speeds and is easier to learn, than the traditional QWERTY layout.

Every computer needs some way of communicating with the outside world. At its simplest, this could take the form of a switch and a light – you press the switch to start the computer and the computer turns the light on to tell you that it has finished its job. This would hardly be a useful interaction.

Keyboards and screens

About the bare minimum of input/output (I/O) facilities essential for a basic computer set-up are a keyboard and a display unit. The very crudest keyboard you will find on the cheapest of home computers is a touch-sensitive arrangement. Most business machines have typewriter-style keyboards, and some have extra keys such as “programmable function keys”, which can be made to produce complicated commands at a single touch.

The most common output medium by which the computer communicates with you, the user, is via a monitor’s screen. This looks like a domestic television set and is connected to a special piece of circuitry that allows it to display ASCII codes as printed characters. On a business computer the monitor is a specially made piece of equipment, while most home computers use a domestic television as the display.

You can think of the screen as a window into an area of the computer’s RAM set aside for this display purpose. Sometimes this area occupies part of the CPU’s main RAM and sometimes it is a separate, additional block of memory that the CPU accesses with the help of some additional circuitry.

Memory mapping

Generally, the memory is arranged so that a specific RAM location corresponds to a specific place on the screen. This is the screen RAM, and it allows the CPU to put characters anywhere it likes on the screen. This process is known as a “memory-mapped display”. Older systems use a terminal accessed through a serial port (an electronic circuit that allows data to be transferred one bit at a time) and many of these systems simply do not permit this degree of flexibility.

The computer’s video circuitry displays the contents of the screen RAM by using each ASCII code as the address with which to examine an area of
ROM containing a pattern of dots which make up that character, and then converting this pattern into a video signal. So, somewhere in ROM there is a pattern of dots making up each ASCII code.

It is possible to take memory mapping a stage further by making each bit in the screen RAM correspond to a dot on the screen; placing a binary 1 in any bit causes the corresponding dot to light up on the screen while a 0 turns it off. This is called "bit mapping" and can give the CPU much greater control over the screen, enabling high-resolution graphics to be displayed. This does require more sophisticated software, and some machines with color displays set aside three times the usual amount of memory to provide red, green, and blue components of a color image, with spectacular results.

Interfaces
Computers generally need to be able to communicate with other equipment as well as screens and Sinclair keyboards
In 1980 Clive Sinclair introduced in the UK the world's cheapest ever computer – the ZX80. He followed this a year later with the ZX81, launched in the United States as the TS1000, a much improved version both functionally and stylistically. Both these machines featured touch-sensitive keyboards, in which moving keys are replaced by a membrane with keys marked on it. Although cheaper to manufacture and impervious to dust, these keyboards are awkward to use. This problem was partly alleviated by the use of "single keystroke" Basic keyword entry. When you enter a Basic program at the keyboard, pressing a single key produces an entire Basic language word. Unfortunately there are more Basic words than keys on the board, so a complex sequence of key presses is sometimes necessary to produce the less frequently used words.

The Sinclair ZX Spectrum uses the same keystroke system but the keyboard has moving keys, molded in a soft rubbery substance. As the Spectrum has keys that are almost the same pitch as a small typewriter, it should be just about possible to carry out some semi-business functions on it, such as word processing, for example. The problem here is the limitations of its display.

Typical monitor
A sophisticated monitor from a typical business microcomputer is shown below. For a usual business application, where somebody will probably have to stare at the screen for hours at a time, the quality of the display is an important consideration. Good points to look out for are a non-reflective coating on the screen (to eliminate eye-fatiguing reflections) and the ability to tilt and swivel the screen to the most comfortable angle for the operator.

Monitors or Screens
The main method used by the computer to communicate with the user is the monitor or screen. It looks like a simplified television set and is capable at least of displaying letters and numbers (alphameric). Most home machines and many business microcomputers can also display graphics symbols. An increasing number of micros now have color displays. Color adds a new dimension to games and other software, but it is more than a gimmick in the business world – it provides another information channel as color graphics can represent information in a far more efficient manner than black and white.

Typical monitor A sophisticated monitor from a typical business microcomputer is shown below. For a usual business application, where somebody will probably have to stare at the screen for hours at a time, the quality of the display is an important consideration. Good points to look out for are a non-reflective coating on the screen (to eliminate eye-fatiguing reflections) and the ability to tilt and swivel the screen to the most comfortable angle for the operator.
Graphics image

The image on the screen is made up of small dots—the more dots the finer the image. These four images (left) are all displayed on a bit-mapped screen with a resolution of 600 horizontal dots by 400 vertical dots. Each dot on the screen corresponds to a bit in the computer's memory—placing a 1 in a memory location lights a corresponding dot on the screen (see enlargement below), while a 0 turns it off.

Displaying text on the screen

Text from a word processor is displayed using similar techniques to graphics, but the ASCII character codes must first be "translated" into the appropriate dot patterns (see enlargement below). Some computers do this within the video display circuitry, using hardware, while others use software techniques. The latter allow easier modification of the characters so that different type styles can be displayed.
serial interfacing

A serial interface takes a byte from the computer and sends it, one bit at a time, using a special chip called a Universal Asynchronous Receiver/Transmitter or UART. The UART must do several things: it must add some extra bits to tell the receiving device that a piece of data is about to be transmitted, then it must send the data, and finally it must tell the receiver that the data has all been sent and the transmission has finished.

First, both transmitter and receiver must agree on the speed at which data is sent. This is known as the "baud" rate. Baud means "units of information per second" and, roughly speaking, the baud rate divided by 10 equals the number of characters a second. Next it is necessary to agree on a "protocol" method for the computer to know when it can send data and for the receiver, or peripheral, to know when it is about to be sent something. The simplest method of achieving this is for the UART to add some extra "start bits" to warn the receiver that the next eight bits form a character, and it tags a "stop bit" on the end to signify that the data has ended. Before successful communication can take place both devices must agree on the number of extra bits. Finally, there must be an electrical standard comm-

**SERIAL INTERFACE**

The serial interface communicates between the computer and a peripheral device such as a printer, by transmitting and receiving data one byte at a time. At the center of the serial interface is the UART (Universal Asynchronous Receiver Transmitter). This device is used to convert the parallel data used within the computer system to serial form for output. It is a two-way device, because it also receives serial data from the peripheral and turns it into parallel format for use by the computer's internal circuitry.

The output sequence For the computer to output data through a serial port (right), the central processing unit fetches it one byte at a time from the memory, and sends it via the data bus, to the UART. Control signals tell the central processing unit that the UART is ready to receive data, and the CPU must also indicate to the UART that the data is about to be transmitted.

UART Inside the UART (below) parallel data is loaded into a shift register. A series of pulses from an electronic clock cause the contents of the register to move one bit per pulse. As each bit "falls out" of the register, it is sent out as serial data.
mon to both devices involved in the particular inter­
face. The computer and peripheral can therefore be
connected by only three wires — a common ground,
one for data coming from the computer, and one for
data coming from the peripheral. Under the RS232
standard the three-wire system works well up to a
certain speed, but for high-speed work it is usually
necessary to add extra wires so that if the peripheral
needs more time to process data (print it on to
paper, for example) it can tell the computer to wait
before sending the next character. This is called
"handshaking". This type of serial interface may
seem unnecessarily complicated, but for simple
things (like sending text to a printer) it does make
the cost of connections much cheaper, especially
over long distances using a device called a "modem"
(see Module 24).

**Parallel interfacing**

It is also possible to send data as eight bits simul­
taneously, using one wire for each bit. This is known as
"parallel I/O", and is often used to work printers
with a standard interface called a "Centronics"
interface. To make the system work, ten wires are,
in fact, necessary — one for earth, eight for data, and
one for handshaking. Sending data by a parallel
interface is obviously much quicker as far as the
actual data transmission is concerned, but the
printer must still keep the computer waiting while it
actually prints out the characters. Many printers
incorporate a "buffer" — a small amount of memory
in which, usually, an entire line can be stored. The
printer waits until an entire line has been trans­
mitted, placing each character in the buffer, and
then tells the computer to wait while it prints that
line. Some printers have much larger buffers, and
add-on buffers capable of holding up to 64 kbytes
are available. This memory capacity speeds things
up considerably as the computer can dump a mas­
sive amount of text in the buffer and get on with
other things as the printer turns out the hard copy.

There is another standard for parallel interfaces
called the "IEEE-488" or Hewlett-Packard General
Purpose Interface Bus. While some machines use
this for communicating with peripherals such as
printers, it is more usually associated with labora­
tory equipment and instruments that need to be
connected to a computer for control or data collec­
tion purposes. It is considerably more complicated
(or versatile) than the Centronics interface as it
allows several items of equipment to be connected
to the same I/O port. Each device contains elec­
tronics that effectively give it an address, rather like
a memory address in some ways, and the computer
must first send out the address of the device to which
it is sending data (or receiving data from) to activate
that particular device and, at the same time, deacti­
vate any other devices connected to that port. Thus
the number of peripheral devices that can be
handled by a computer is increased considerably,
but at the expense of extra electronics and more
complex software.

<table>
<thead>
<tr>
<th>RS232 interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a fairly standard interface for serial communica­</td>
</tr>
<tr>
<td>tion between computers and peripherals. As serial data emerges from the UART it is usually in the form of a +5v pulse to signify binary 1 and a 0v pulse to signify binary 0. Before being sent to a peripheral, these are converted by the RS232 interface cir­cuity into either a +12v pulse to rep­resent the binary 1s or a -12v pulse to represent the binary 0s.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PARALLEL INTERFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 wires are needed to send data in parallel format — one for each bit, plus two for ground and handshaking (see below). One allows the peripheral to indicate that it is ready to receive data, and the other carries the computer's signal that it is about to trans­mit a byte to the peripheral.</td>
</tr>
</tbody>
</table>

---

Handshaking signal

2 start bits
8 data bits
1 stop bit

Data format produced by UART

+12v
0v
-12v

Peripheral

RS232 data format

UART

Computer

Ground connection

Peripheral
Because a computer's RAM is wiped clean every time the power is switched off, and because it would be tedious to have to type in a long program whenever you needed to use it, various methods have been devised for storing programs and data in a form that can be read directly into the computer. Disks are one method (see Module 22) and they are becoming increasingly popular for the home user, but most home machines can also save programs and data on ordinary audio cassette tapes.

**Cassettes for data storage**

If the hardware designer has done a good job (not always the case with home micros), you can use almost any domestic tape recorder to store programs and data. You can also use ordinary audio tape, although it is safer to use specially prepared digital tape that has been manufactured to a considerably higher standard.

There are several disadvantages to cassettes that make them unsuitable for business use. For a start, they are very slow—cassette baud rates (see Module 20) vary from 300 to 2400 baud, but even the fastest are many times slower than disks. Loading programs and large amounts of data at cassette speeds is acceptable for most hobbyists but too slow for business users.

The biggest disadvantage to tapes, though, is that they allow only serial access to data. So if you have built up a file on cassette and want to, say, read the 50th item, you must wait while the cassette plays its way slowly past the first 49 items. Then, if you want to refer to the 20th item you must rewind the tape to the start and work through the first 19 items.

Various attempts have been made to produce cassette systems that overcome these problems. One, called the stringy floppy, used a continuous tape loop to allow access of individual data items. With this system you read item 50 and the system waits automatically for number 20 to come round again, thereby eliminating the time spent rewinding the tape. It worked quite well, was much faster than...
ordinary cassette systems (but was still slower than disk), and had a good reputation for reliability, but never caught on with users.

Other systems have been devised in which the entire cassette deck mechanism, including fast forward and rewind, was under computer control, with arrangements of varying ingenuity and efficiency for telling the computer exactly where it was in the tape at any given moment. But these systems have not been a success either, partly due to the fact that they are rather expensive. Their main application seems to be for use in "hostile" environments such as dusty factories, where disk drives cannot operate with any degree of reliability.

For software producers, too, cassettes pose a particular problem — software theft (sometimes euphemistically called "piracy"). It is often very easy to copy a commercially produced program from one cassette to another, and although this is often done without any actual intent to steal (innocent trading between friends is the usual case), it does deprive the people who put in the hard work developing the program of their money.

**ROM cartridges**

Both the problems associated with cassettes — lack of speed and software theft — can be solved by fixing the software permanently in one or more ROM chips. The ROMs are mounted on a small printed circuit board and all their connections linked up to a special connector on the edge of the board. The board is then enclosed in a plastic case and the result is known, generically, as a ROM cartridge.

Many home computers now have a slot in the back or side into which these ROM cartridges can be plugged. Each make of machine only accepts the cartridges produced by its own manufacturer or those designed specifically for it. To use the cartridge you switch off the machine, plug it into its slot, switch on again, and either the machine will begin to run the program in the cartridge immediately or you type in a special command to activate it.

---

**CARTRIDGES**

Cassettes suffer from two disadvantages — they are slow to use and it is easy to copy a cassette, making software theft a real problem for manufacturers. To overcome these problems, many micros have a socket into which you can plug software cartridges. Cartridges are small plastic containers housing a printed circuit board on which is mounted one or more ROM chips containing the actual program. The program is usually a game but occasionally it can have a more "serious" application, such as simple word processing, for example.

With the cartridge in place, the program is usually activated by either turning on the computer or by typing in a simple command.
The contents of RAM disappear when the power is switched off, therefore it is necessary to have a way of storing programs and data in a more permanent way so that the computer can read them whenever necessary. And although you have seen that you can use cassettes to do this (see Module 21), they have severe limitations, particularly for business use.

Magnetic disks overcome the problems of tape-based storage systems and are universally used on business microcomputers as well as being available as optional extras on an increasing number of home machines. In principle, disks are a cross between a tape recorder and a record player. The disk itself is coated with a magnetic oxide compound similar to that used on recording tape. The disk drive (see facing page) contains a central hub which spins the disk, typically at 300 rpm for a floppy disk. An arm carrying a read/write head is moved under computer control back and forth across the disk’s surface.

**Formatting the disk**

To allow the computer to find information on the disk easily, its surface is divided up into concentric tracks, each of which is in turn divided into a number of sectors. There is nothing visible on the disk’s surface – they are created by the computer itself in a process called formatting or initializing. This has to be carried out on every new disk before it can be used for data storage and retrieval.

**Disk directory**

The computer keeps a record of everything recorded on the disk by reserving a couple of tracks to store a "directory" of the disk’s contents. The directory contains the name of each program or data file and a "map" of which tracks and sectors each occupies. On an ordinary CP/M machine, the operator can see what is on a disk by typing DIR (short for DIRECTOY) and the computer will display all the program and file names. Computers using other operating systems have a similar facility.

When, on a CP/M machine, you want to use a program, you first type the program name. The computer then looks in the disk directory for a program with that name and, assuming it finds it, uses the map to locate which tracks and sectors on the disk actually contain that program. As it finds each

---

**Floppy Disks**

The floppy disk is the most widely used medium for storing programs and data with business micros, and is becoming increasingly popular with home micro users. Although disks and disk drives come in a variety of shapes and sizes, they all work on the same principle—a cross between a record player and a tape recorder. The disk itself is made of a plastic material coated with magnetic oxide and then sealed in protective sleeve, from which it is never removed. A slot in the sleeve allows the disk drive’s read/write head to access the disk under computer control.

8 in

5½ in

Alignment notches

The write-protect notch

Sizes and formats

Disks come in the original 8 in version, 5½ in, and micro-floppy (3 or 3½ in) sizes. The disk’s surface is divided into tracks, each of which is divided into sectors. These divisions are defined by signals recorded by the computer during a process called formatting (done with each disk). To allow the computer to detect the start of the first sector of each track, a hole is punched near the hub of the disk. This is known as “soft sectoring”, because the recording and detection of sectors is carried out in software. “Hard sectored” disks have a hole punched near the hub for each sector.

84
Inside the disk drive: More often than not, disk drives are buried inside the main box of the computer. The only part normally visible is the slot through which the disks are inserted. Inside, though, you can see a combination of sophisticated control electronics and precision mechanical components that are required to record data on and retrieve data from the floppy disk. Deep inside the unit is located the all-important read/write head (see detail right).

Control electronics: These electronics convert the commands sent by the computer into pulses that move the head back and forth.

Read/write head: This moves back and forth over the surface of the spinning disk. It is rather like a tape recorder's head in that it turns electrical signals into magnetic fields, which are then recorded on the disk. The process is reversed during the reading operation. In a single-sided drive a pressure pad presses against the disk to keep it in contact with the head. On a double-sided drive, there is a second read/write head located here.

Head motor: This controls the movement of the head across the surface of the disk.

Disk motor: This motor rotates the disk, usually at 300 revolutions per minute, although some machines have a variable-speed mechanism.

Disk door: The floppy disk is inserted here and is then held in place by a locking mechanism.
HARD DISKS

The amount of information you can record on a disk depends on, among other things, the speed at which the disk spins and the distance between the disk's surface and the read/write head. Hard disks spin quicker than floppies, and this creates a moving layer of air in which the read/write head is designed to "fly." A special area of the disk is reserved for the head to "land" and "take off" whenever the power is turned on or off. If the head touches an area of disk used for data storage, both the head and disk will be damaged. To prevent air-borne particles sticking to the disk, the unit is kept in a sealed enclosure. With the disks sealed, however, you cannot take a disk out and slot in another, and it is expensive to install two of them just to keep a safety back-up.

At present, hard disk units provide the greatest amount of storage - 5, 10, or even more megabytes, compared to a maximum of less than 1 megabyte for the densest floppy disk. Hard disks use the same basic technology as floppies, but the disks are made of rigid backing material (typically aluminum) with a much higher-quality coating. This extra storage capability does make a tape back-up unit more important.

Tape back-up Hard disks provide huge storage capacities - up to 20 or more megabytes. One problem with the hard disk, though, is that it is hermetically sealed. This means that data cannot easily be copied on to a spare disk. To prevent a total loss of its contents (should anything go wrong) you can make copies on lots of floppy disks, or use a special hard disk back-up unit. This unit is necessary both to save time and because the size of the file being copied will probably exceed the capacity of the floppy disks. With the back-up unit, it is possible to store the entire output of a hard disk because of the physical length of the tape used.

Disks The disks themselves are made out of one or more accurately machined aluminum "platters" coated with magnetic oxide.

Chamber A hermetically sealed chamber covers the disks to prevent dust and dirt entering and causing the read/write head to "crash" on the disk's surface.

Circuit boards Control signals from the computer are "translated" by these boards of electronic components into instructions that move the head back and forth across the disk's surface.

Head actuator This mechanism moves the read/write head across the disk's surface, positioning it accurately on the tracks.

Read/write head This is aerodynamically shaped to "fly" in the moving layer of air that builds up on the disk's surface when it spins. There is one read/write head for each disk.
one it reads it into memory (see Module 19), and when the last one has been read in, the computer begins to execute the program.

Writing information to a disk—that is saving data or a program on it—is carried out in a similar way. The computer uses the information in the directory to work out which tracks and sectors are unoccupied, stores the data on these and builds up a map of the new file, which it then puts into the directory.

This system allows the computer to reach any part of any file stored on the disk directly. Once it has worked out which track and sector contains the information it wants, it can go directly to it instead of having to read through everything else in the file, as it would with cassette storage.

The way in which disks store information is very complex, especially when you remember that the computer, having decided which part of the disk it wants to access, then has to control the disk drive mechanism. It does this by sending a series of signals that cause the read/write head to step back and forth across the disk’s surface until the right place is reached.

**Disk types**

Provided that you take care not to damage a floppy disk in any way, it is a reasonably reliable method of storing data. Disks can wear out, however, and no matter how careful you are, accidents do sometimes happen, so you must always keep a back-up copy of your information on another disk. How often you copy disks depends on how much data you can afford to lose—doing it at the end of every day means that, at worst, you will lose a day’s worth of information should your working disk fail before you have backed it up. Floppy disks have other disadvantages, too. They do not hold very large amounts of information—capacities range from 100 kbytes to over 1 megabyte—and in computer terms they are rather slow (even though you might consider them quick when you see them working). To overcome these problems, some machines are now equipped with “hard disks”, either as optional extras or built into the basic machine as standard.

The most common type of hard disk presently available for microcomputers is the Winchester disk, so called because the first one to be made, by IBM, bore the same model number as the Winchester rifle—3030. These hard disks comprise one or more rigid aluminum disks sealed inside an airtight plastic enclosure. Hard disks allow you to pack information far more densely on the disk’s surface. This is achieved partly by spinning the disk more quickly and partly by designing the read/write head so that it “flies” across the surface. The read/write head is aerodynamically designed to use the moving layer of air produced by the spinning disks as a cushion, keeping a minute gap between it and the disk surface. Were the head to touch the surface it would gouge a groove straight through any information stored there, destroying not only the head and the data but also the delicate disk coating.

A hard disk set-up can contain up to 20 or more megabytes in a unit no larger than a floppy disk drive, and they work many times faster than a floppy. They do, however, suffer from two disadvantages. First, you cannot take out one disk and replace it with another, as you can with floppy disks. Second, although hard disks are far more reliable than floppies, they can still go wrong. Because of this you require some method of backing up the information on them, either by using many floppy disks (and creating a storage problem) or by using a special tape back-up unit (see facing page).
All business computers need to produce material on paper for one reason or another. This hard copy might be for invoice purposes, text from a word processor, or self-adhesive labels for addressing envelopes. Many hobbyists also buy printers so that they can produce listings of programs they are working on—it is much easier to find out what is wrong with a program if you can see the whole thing listed out on paper rather than by working through it a screenful at a time.

Printer types
There are two basic types of printers presently in use associated with microcomputers—dot matrix (see facing page) and daisywheel (see below). The one you choose really depends on what you want to use it for (the daisywheel types produce a better quality image) and the amount of money you have to spend (dot matrix types are much cheaper).

As its name implies, the dot matrix printer forms characters on paper by building them up from a series of dots. In the most common variety, a vertical line of seven or nine needles (each one operated by a solenoid) passes across the paper in a series of minute steps. At each step whichever needles are required to produce dots are "fired" at the paper. In front of the needles is an inked ribbon, just as in an ordinary typewriter, and the character is built up dot by dot.

The major problem with dot matrix printers is that they produce characters with a broken appearance, which is fine for rough drafts or for accounts.
work, but is not really suitable for letters going out to clients, for example. The printer manufacturers have invested a lot of time and effort to try to improve the quality of output from these machines, using a larger number of needles, or moving along the paper in smaller steps to allow dots to overlap. Some very good results are possible but at the expense of both speed and print head life — the needles become very hot and wear out quickly.

In order to produce typewriter-quality output you need a daisywheel printer. You are probably familiar by now with the modern type of office typewriter, which uses a spoked wheel instead of type bars or golf balls. The wheel spins very quickly until the desired character is uppermost, and then an electromagnetic hammer is activated and knocks

---

**Hardware · Module 23**

---

**Paper for printers**

The sort of paper you need depends on your printer. Electrostatic and thermal types require special paper, but neither are suitable for business use. Dot matrix and daisywheels allow you a wider choice. Continuous stationery, with pages linked together but perforated for repetitive work such as invoices is often used.

---

**DOT MATRIX**

This system uses a matrix of dots to form a printed character. It prints a column at a time by means of a moving print head holding a vertical line of pins. As the head moves across the paper the appropriate pins are fired against a ribbon to build up each character. Dot matrix printers are usually faster and cheaper than daisywheels, and they also tend to be quieter. Print quality is not as good, but they can print graphic characters.

---

**Print head enlargement**

This shows the "business end" of a print head (left) with the needles just visible. Usually 5 columns of 7 needles are used (above) although some use as many as 24 needles.
it against an inked ribbon to print the character. Daisywheel printers are more expensive, noisier, and heavier than dot matrix printers, but they remain the only way of obtaining a top-quality print-out from a computer at present. Also, even the fastest daisywheels are still slower than medium-speed dot matrix printers.

**Thermal and electrostatic printers**

Other types of dot matrix printer use the same basic principle described above but have heat-producing or spark-producing electrodes instead of needles and an inked ribbon. These are called thermal and electrostatic printers, respectively, and they require specially treated (and expensive) paper to work. They are, however, usually faster and quieter than the needle variety. Some home computer manufacturers have started to produce very low-cost printers – the Timex/Sinclair ZX Printer, for example, is an electrostatic unit that builds up characters one line at a time using a thin electrode on an endless belt arrangement. It is still rather crude but works well enough for hobby use.

**New developments**

Attempts to produce good-quality output at high speed have resulted in new types of printers which are about to come on to the market. One strong contender for further development at the moment is the ink jet printer, which fires ink dots at the paper to build up the characters from a large number of tiny dots. Laser printers are another possibility, but are still very large and expensive, and are only just catching on for larger computer systems. Some manufacturers have announced "desk-top" models (so called, perhaps, because they seem to be about the same size as a desk). Laser printers are incredibly fast and immensely versatile, but it is unlikely that you will be seeing them on the personal computer market in the immediate future.

**ELECTROSTATIC AND INK JET**

The Timex/Sinclair printer (below), the first very low-cost home computer printer, works on the electrostatic principle. Instead of using a column of electrodes to build up characters, it uses two – only one of which is in contact with the paper at any time, so characters are built up dot by dot. For extra speed, the electrodes are mounted on a continuous belt so that when one has scanned the paper, the other is just starting.

Ink jet printers fire minute electrostatically directed blobs of ink at the paper from a nozzle in the print head. But this technology poses its own problems as the ink has to be liquid when it leaves the nozzle but dry the instant after it hits the paper. Olivetti has overcome this problem with the spark jet (below). This uses a stick of solid ink and an electric spark to melt a small blob at the tip, which is then fired at the paper.
There are four popular types of printers available for micros today. Daisywheel units produce typewriter quality output and are used mostly for word processing where type quality is important. Low-cost dot matrix types are cheaper, quieter, and faster but type quality is not as high. Thermal units cannot be used for high-quality work, but they are nearly silent and ideal for program listings or rough drafts. They do require special paper. Electrostatic printers also require special paper and print quality is fairly poor. But they are adequate for home use.

1 Diablo 630 This daisywheel printer can print out between 32 and 40 characters a second. It weighs 60 pounds and uses either single-sheet or continuous paper.

2 Epson MX-80 The Epson is a dot matrix printer with a character rate of 80 per second. It uses continuous paper.

3 Apple Silentype This can print up to 40 characters per second. It is a thermal type printer and, therefore, uses special paper that only comes in rolls.

4 Sinclair ZX The ZX is an electrostatic printer and is only really of use for the most basic of applications. It uses its own special paper in roll form.
There is a wide range of specialized peripherals that you can use with microcomputers. An increasing application of computers involves transferring information between machines using the telephone system. To send data along a phone line it must first be converted into audio tones and sent via a serial interface (see Module 20) to a device called a modem. This performs the actual conversion and connects directly to the telephone system. But because of the direct connection, you need the telephone company’s permission before installation.

You can overcome this problem by using an acoustic coupler. This device has two rubber cups into which you fit the telephone handset after first dialing and being connected to the remote computer also equipped with an acoustic coupler.

An increasing number of computers have graphics capabilities, but producing paper copies of graphs and diagrams requires specialized equipment. While some dot matrix printers can produce graphics, the more usual method is to use a plotter. This is a computer-controlled pen that draws the diagram on ordinary paper or transparent film for overhead projection. Sophisticated plotters come with eight or ten pens, allowing full-color artwork.

Also available are speech recognizers and synthesizers. Speech recognition is the subject of massive research budgets all over the world. But speech synthesizers are somewhat easier and rely, usually, on a ROM chip containing the digitized versions of a range of phrases, words, or parts of words. An important application of microelectronics concerns the conversion of analog signals to digital form, and vice versa. An analog-to-digital converter measures the analog signal at regular intervals, and each measurement produces a binary number corresponding to the signal’s voltage. To play back the sound, the numbers are fed at the same rate into a digital-to-analog converter, the output of which is an analog signal corresponding to the original. When linked to a computer, A/D and D/A conversion is particularly useful in scientific and engineering work. It enables the computer to monitor and analyze many laboratory processes and to produce analog signals for control purposes.

MECHANICAL MOUSE

The idea of this "mouse" is to reduce the necessity of using the keyboard to the minimum. The mouse is a small box on wheels, connected by a cable to the computer. As the mouse is moved around on the desk top, an arrow on the screen moves accordingly. Once the arrow is positioned over the picture representing the chosen task, you press a button on the mouse to tell the computer.

Inside the mouse

As the mouse moves around on a steel ball (1) its movements are transferred to two small drums (2). The ball turns the drums in proportion to the length and direction of travel. Small wheels (3) covered with strips of conductive and non-conductive material are connected to the ends of the drums.

As the wheels turn, they send electrical signals for each rotation of the drums. Wires (4) connected to the wheels decode the electrical signals and send them in digital form to the computer. The three buttons (5) on top of the mouse are used to choose the application from the screen symbols.
Computers can talk to each other using the telephone system provided they are equipped with the necessary peripherals to convert binary signals into tones that can be transmitted along the telephone lines. One use of this facility is to send data between a micro and mainframe. A large company, for example, might have a number of branch offices, which, every day, need to send details of transactions to the main office's computer.

Home hobbyists, too, are big users of telephone facilities, to access "bulletin boards" (remote computers containing various types of information and/or offering electronic mail facilities).

To connect a computer to the phone system you need a "modem" or "acoustic coupler". These accept serial data from the computer and convert it to audio tones, one tone signifying a 1 and another for 0. These tones can then be sent along the lines and decoded at the other end by similar equipment.

**Acoustic coupler** This device (right) allows you to connect your computer to the phone system without any special wiring. The coupler plugs into the computer, usually through an RS232 interface, and all you need do is dial the computer you want to contact on your phone. Once connection is made, the handset is placed into rubber cups on the coupler. Although easy to use and cheaper than a modem, couplers cannot receive or transmit data as quickly as a modem and they are also susceptible to any background room noise.

**Modem** A modem (below) has to be wired directly into the telephone system and is then connected to the computer in the same way as a coupler. Because connection between computers is direct, modems are not affected by exterior noise and data transfer is quicker than with couplers.

**Loudspeaker** The coupler converts serial data from the computer into audio tones that are picked up by the microphone in the telephone handset.

**Microphone** Signals from the remote computer are "heard" by the coupler as audio tones; the coupler translates these to serial data and sends them to the computer.

**RS232 interface** It is through this interface that the coupler communicates with its computer.
GRAPHICS PERIPHERALS

Using a set-up like the one here, designers and artists can produce extremely complex designs with simple-to-use software and hardware. Typically, a joystick is used to "draw" interactively on the screen. The resulting picture can be saved on a floppy disk for later use, and can also be drawn out on a plotter, often in color. The system illustrated is the Robocom Bitstik, an amalgamation of special graphics software and add-on peripherals that turn a computer - in this case an Apple II - into a very powerful graphics tool. The specially engineered joystick (below) allows you not only to draw on the screen but to shrink and enlarge your drawings, rotate them, and to position them wherever you like on the screen. Thus, an extremely complex drawing can be built up a piece at a time, with each piece being shrunk once it is finished and positioned properly in relation to the other components of the picture. At any time, you can zoom in on a part of the image, with the amount of detail being limited only by the disk space available on the computer at that particular time.

Computer-aided design Like other design packages, this one, based on the Bitstik allows you to build up and save on disk whole libraries of pictures or picture elements (see above right). A typical system (below) for computer-aided design work with specially written software would require a computer with disks, preferably a color monitor, the joystick, and a graphics plotter to produce hard copies of the final design. On some systems you can also use a dot matrix printer with graphics facilities.

Hardware · Module 24

94
PLOTTERS AND TABLETS

While many computers allow you to display graphics on the screen, usually you need to put your design on paper at some stage. Dot matrix printers with graphics facilities are available. These are fast and inexpensive but image quality is not high.

For high-quality work a plotter is the best peripheral. This has a special pen mounted on an arm. Commands sent by the computer tell the plotter where it should move and any shape can be drawn out on paper. It is difficult, though, to draw, say, a car using the keyboard. And to overcome this you can use a tablet. You simply draw on the tablet using a special pen and the computer reads the pen's position as it moves across the tablet, recording the pen co-ordinates, and displays the resulting drawing on the screen.

Plotters A plotter like the one below holds the paper electrostatically. In response to commands from the computer, the pen is moved around very precisely. There are commands to lift the pen so that it can move elsewhere. Usually, lettering can also be produced, in different sizes and styles. Many plotters now offer a range of colored pens.

Pens Colored pens under computer control.

Arm The arm with pen attached moves in two dimensions under computer control.

Tablets A graphics tablet allows you to enter freehand designs into the computer. The computer tracks the pen across the tablet and the result is a digitized version of your drawing held in memory or saved on disk for later use with a plotter or printer.
COMPUTERS AND SYNTHETIC SPEECH

It is now possible to get your computer to speak to you—not just display words on the screen, but actually produce speech. Speech synthesis add-ons are increasing both in numbers and in power and versatility. Early versions tended to sound like a small astronaut trapped in a sewer, but it is only a question of time before most home computers will come with very realistic speech capabilities.

Two methods are used to produce speech. The most realistic comes from units like the Chatterbox (see below) which incorporate real speech in digitized form and simply convert it back to analog and then play it through an amplifier and speaker. This system, though, does use a lot of memory. The other method—true speech synthesis— Involves a speech-generating chip. This chip takes codes corresponding to “allophones” — the basic sound components of speech—and strings them together to form words. Although not yet as realistic, it is the system of the future.

Analog to digital

Sound waves travel through the air in the form of waves of higher or lower pressures. The same pressure waves can be converted into an electric current with a microphone; the resulting current varies in sympathy with the sound, as shown here—it becomes a current with a varying voltage. This can be turned into digital form by measuring the voltage at set intervals. Each time you measure the voltage you get a binary number corresponding to the voltage at that point, which can be stored in memory or even on tape or disk. As the voltage changes, so too does the value of the binary numbers coming from the analog-to-digital converter. By feeding these numbers to a digital-to-analog converter at the same rate as you recorded them, you can re-create the original sound as an analog electrical signal and this can be played through a loudspeaker.

MEMORY PROMPTS

Here are some short definitions of some of the words found in this section.

Bit mapping A system in which an area of memory is set aside to hold information as it is to be displayed on the screen. Each dot in the memory block corresponds to a dot on the screen.

Graphics The displaying of information in diagrammatic form.

Hardware The physical components that make up a computer system.

Plotter Device for reproducing computer graphics on paper.

Video Disk A device for storing information in digitized form, readable by a special unit.
SOFTWARE

10 REM RULE
20 REM
30 PRINT "PEN"
40 PRINT "===
50 PRINT "PIM
60 LET P = 100
70 LET C = 1000
80 PRINT "We each
90 PRINT "broke in
100 REM
110 REM First, let
120 REM he wants to
130 REM to gamble...
140 REM
150 PRINT : PRINT
160 INPUT "Type a
170 IF C$ = "Typ
180 IF C$ = "Typ
190 IF C$ = "Typ
200 IF C$ = "Typ
210 IF C$ = "Typ
220 IF C$ = "Typ
230 IF C$ = "Typ
240 IF C$ = "Typ
250 IF C$ = "Typ
260 IF C$ = "Typ
270 IF C$ = "Typ
280 IF C$ = "Typ
290 IF C$ = "Typ
300 IF C$ = "Typ
310 IF C$ = "Typ
320 IF C$ = "Typ
330 IF C$ = "Typ
340 IF C$ = "Typ
350 IF C$ = "Typ
360 IF C$ = "Typ
370 LET B = 10
380 PRINT "= GDSUR
390 REM

AN INTRODUCTION TO SOFTWARE

Up to this point in the book, the emphasis has been very much on computer systems and hardware, and only passing reference has been made to the other major aspect of computers – the software, or instructions that tell the computer what to do, and without which the hardware simply would not be able to function.

To many non-experts the software side of computers may appear fairly mystifying – after all, you cannot see it, touch it, or see a picture of it, yet it is vital to the system. Software can also be expensive; sometimes up to half of the cost of a fully operational business system can be taken up with software expenditure. And all you get is a disk and an instruction manual. You do not, however, have to buy your software prepackaged. It is perfectly possible to write your own programs (see Module 27) for entertainment purposes and also sometimes for business applications.

System hierarchy
In order to gain an understanding of how the various types of software are used and how they interact with one another, it is useful to think of a computer system (both hardware and software) as a sort of core surrounded by a shell, as shown in the diagram right. In the center of the shell is the computer hardware – the machine itself. Outside of the shell is you – the user. Between the two is situated the computer software, which, as you can see, is made up of several “layers”.

Closest to the machine itself is the operating system (see Module 33). This takes care of the basic “housekeeping” functions that the system has to carry out. These functions include looking after the disk drives, accepting characters typed in at the keyboard, and displaying characters on the screen.

On most home micros the operating system is quite invisible to the user because it is incorporated into the Basic “interpreter” – circuitry that takes the characters typed in and translates them into machine code that the CPU (see Module 17) can understand. With the majority of business micros, the operating system is the first thing the user comes across after switching on. Typically, the operating system for a business machine is kept on a disk – the machine is switched on, the disk inserted, and the operating system loaded automatically from the disk. At first sight this may seem a little cumbersome – why not, for example, build the system with the operating system in ROM (see Module 19)? The answer is that by keeping the operating system on disk it can be altered easily, either to correct faults in the original version or to add new facilities. All the software manufacturer need do is send out new disks and the user can change to a new operating system simply by loading the disk. The most widely used operating system for business microcomputers is called CP/M and there is a very large range of software available for this type of operating system (see Module 33).

Even an operating system – no matter how complex – is not of much use by itself. It merely allows you to perform a few chores such as formatting a new disk (see Module 22) and examining the “directory” (the list of program and text file names held on the disk currently in use) of a disk. Beyond that, there is nothing immediately useful that can be done without more software.

There are two other layers of software that can be seen in the diagram – programming languages and applications programs. A programming language allows you to write computer programs in a very simplified English that can be readily understood by systems users. The programming language software then converts this into the binary instructions that the computer can work with (see Module 13). The most popular programming language in the microcomputer world is called Basic, which stands for Beginners’ All-purpose Symbolic Instruction Code. This language was specifically designed to be easy and quick to learn, so that computer users without any formal computer training could very quickly come to terms with the machinery and carry out useful work with the minimum possible time spent learning a new skill.

Basic, however, is by no means the only programming language. In fact, there are many alternatives, although not all are available on microcomputers. While Basic is reasonably good in most respects as a general-purpose programming language, it does have some deficiencies that make it unsuitable for certain jobs. There is unfortunately no single language that could justifiably be described as the per-
SOFTWARE HIERARCHY

Between the user and the computer hardware there is a whole hierarchy of software, layers of different types that each play a role in enabling the user to communicate with the computer. Any useful piece of software needs to perform certain functions, such as accepting and displaying characters or driving a printer. And most computers come with these types of functions built into a piece of software called an operating system, which forms the innermost layer of the software hierarchy. The other categories of software can be of two types: compiled software (where the computer using a compiler turns the English-like commands from a text editor or word processor into binary instructions, which can be loaded into memory and used by itself) and interpreted software. Here, there must be a language interpreter in memory, which turns each line of the program into machine code (a line at a time) as it is being used, and thus forms an additional layer in the hierarchy. And outside of this system of language software is you – the user.

Systems relationships
This diagram shows the relationships between the various elements of the computer system and their human user. At the center of the shell is the hardware—the computer itself. Forming concentric circles around it are the operating system and the applications programs (see far right). Between these two elements in the hierarchy are the programming languages (see below). Outside of the shell is the user. It is these elements of software that permit the user and the computer to communicate.

PROGRAMMING LANGUAGES

It can be puzzling to discover how many programming languages there are (a few of which only are described here). Eventually it should be possible to talk to computers in normal language, but until then you will need to know how to instruct your computer. The reason for such diversity is that, so far, it has proved impracticable to devise a language that caters for every application.

Basic
This is the most widely used programming language in the world. Most implementations of Basic are interpreted versions. Nearly every computer manufacturer has tended to adapt it. C This is a compiled language and very versatile. It produces a compact, fast-running code suited to writing operating systems and applications software. Pascal Another compiled language highly regarded by programmers writing general business packages. Cobol A good language for handling files of information and for writing financial-type packages. Not strong for applications such as writing word processor or an operating system. Forth An interactive language that allows the user to define new words from a set of defined words that already exist. Logo Combining aspects of artificial intelligence and psychology, Logo aims to teach children how to adopt a general approach to problem solving.
fect programming language, despite the claims of certain computer language writers. Each language was either designed for a very specific type of job or happened to be suitable for one type of programming and unsuitable for others.

Even a programming language (with very few exceptions) is only a means to an end, rather than an end in itself. In order to be able to perform a real job of work you need an applications program — a program that will carry out the particular function for which you originally purchased the computer. The purpose of a programming language is to allow an applications program (see Modules 34 and 35) to be written as quickly and as easily as possible. If you are a business user, for example, you will probably not be particularly interested in programming languages — you will want a job done and the applications program that can do that will be your major concern. If, however, you are a home hobbyist, languages may be of much greater interest. For most computer hobbyists, applications software means games (see Module 29), often bought off-the-shelf from the growing number of specialist computer software stores. There are now some extremely sophisticated games available for computers, but there can be a tremendous amount of satisfaction in learning to program and then writing your own games. Basic is often too slow for games that involve movement or animation and the serious games programmer will inevitably want to investigate other ways of translating games concepts into silicon by using languages that allow more exciting things to happen (see Module 31).

Computer programming

Learning to program a computer is not nearly as difficult or technical as many non-experts think. You do not even have to know very much about computers (although, if a rule can be made, the more you know the better your programs will be), and you most certainly do not have to be any sort of expert mathematician. In fact, surveys carried out among professional programmers have shown that if there is a "best" type for programming, it is somebody who is good at languages and can communicate ideas effectively. For this is what programming involves — communicating ideas in a special language (which, like all languages, has its own rules), although the ability to analyze and break it down into its constituents is an important part of writing good programs, too.

The diagram we looked at earlier shows that some applications programs "interface" directly between the operating system and the user, while others require a programming language sandwiched between the user and the operating system. There are two ways of turning a program written in a programming language into the computer-understandable binary instructions. One method, called "compiling", involves the programmer writing the program with the aid of a word processor or text editor (see Module 36) and then running a "language compiler" to turn the program into binary instructions. With an operating system like CP/M, the program can then be used immediately, simply by the system user typing in its name — the operating system looks for the program on disk, loads it into RAM, and "hands over" control to it. When the program has finished running, the operating system takes over control once more.

Most home computers and many business machines use an "interpreter" rather than a compiler. The interpreter — normally for use with the Basic language — is often held permanently in ROM in home machines and on disk in business micros. To write a program with a Basic interpreter, you first load it into RAM (unless, of course, it is a ROM-based one) in exactly the same way as you would if running a program that had been compiled. The interpreter allows you to type in your program and to save it on cassette (see Module 21) or disk (see Module 22), but no attempt is made at this point to turn it into binary instructions. Some versions of Basic simply store what you type in the form of ASCII text (see Module 13) in RAM while others perform a sort of intermediate operation that reduces the amount of memory required to store the program by converting each word in the language into a single-byte token. At this stage a few interpreters check each line after you have typed it to make sure that you have not made any typing errors with the language's own words and that each word is used correctly — computer languages, like all other languages, have grammar and syntax.

With the program typed in (or loaded from disk or cassette) you need to type a command like RUN to start the program off. The interpreter then works its way through the program, converting each line in turn into the requisite binary codes, and executing them before continuing to the next line. This, as you can see, is much slower than running an already compiled program. Depending on the particular language and computer, an interpreted program can run anything between 5 and 50 times slower than its compiled equivalent, although for many applications this lack of speed is not very significant.

Software applications

A vast range of computer software is now available for almost any machine — both home and business —
to do everything from playing games to carrying out sophisticated "artificial intelligence" work (see Module 37).

The home computer games market is immense, fuelled by the boom in the last few years in home computer sales. It would appear that playing pre-packaged games is one of the major uses for home computers, despite the more-often voiced claim that they are bought for their educational value. It is not uncommon, though, for those who become involved in home computing to go on and write their own, original games, or write improved versions of those already on sale.

Computer literature
One problem faced by many home computer hobbyists is the frequently very poor standard of documentation accompanying microcomputers – especially those machines that have only recently appeared on the market. Trying to learn to program in Basic from some of the manuals currently available is almost impossible. Apart from poor spelling, misleading punctuation, and bad presentation, they are frequently jargon infested, badly worded and, sometimes, completely inaccurate. Machines imported from the Far East often have manuals that have been poorly translated into English and although these can sometimes make amusing reading, they are no substitute for the clear, concise explanations every programmer needs.

Even when manuals are well written, authors frequently assume that their readers are familiar with many basic computing terms, which are in no way obvious to newcomers to the subject. This section of the book, therefore, introduces some of these concepts and outlines the principles of programming in Basic. Unfortunately, the most complex aspects of programming a home computer are usually those concerned with the more attractive features of the machine, such as sound, color, and graphics. Because there is no standard in Basic for controlling these facilities, and because the facilities available vary considerably between machines, there is no practicable way of explaining in detail how these are used, although some of the broad principles behind their use is discussed.

One way of finding out more about your own computer is to buy a book that deals specifically with that machine. These days the appearance of a new micro is normally accompanied by a flood of books and even magazines. The best advice is to ignore the books that come out almost simultaneously with the computer, and to wait a while instead. The early books are likely to be as error prone as the manuals, and they have probably been written by somebody with access to a prototype machine only. Often, there are changes to the computer before it goes into production and aspects of the prototype on which the book was based may have altered in the production model. Generally, it is also not a good idea to buy a new machine as soon as it appears on the market. However eager you may be to get your hands on the latest micro available, wait a few months until the inevitable bugs and production/delivery problems have been sorted out and some worthwhile books and software are available.

Despite the more serious applications for business users, software for business microcomputers is in many ways a much simpler subject altogether. For a start, no business users buying a computer strictly for business purposes should ever consider writing their own software, unless there is some particularly obscure application for which there is no pre-written software available. Writing software for fun is one thing as there will be no serious consequences if things go wrong. But, trying to develop software for, say, stock control will take a lot of time – time that probably could be better spent on other aspects of business. It will also involve a lot of hard work and could, if things go very wrong with the program when you are using it, have a disastrous effect on your business. There are literally hundreds of programs already on the market covering things like stock control and virtually every other aspect of business a computer can be applied to. In the end you will find that it is cheaper to buy the software package that most closely does the job.

Bad documentation also plagues the business computing world but, fortunately, it is slowly improving. Increasingly, software developers are either enhancing the ease of use and "user-friendliness" of their products or organizing training courses for software end users. With a complex package, it is often better to take a training course rather than struggle through a large manual. On a course you will at least pick up the basics of working with the package and you may well be made aware of features that you could otherwise have missed. In the not too distant future, the business buyer will be able to walk into a computer or office equipment store with a business problem and walk out with a box of electronics that will solve the problem. Once back in the office, all that will be necessary is to unpack it, plug it into the existing system and start to use it immediately, much in the same way as today with a typewriter or photocopier machine. In five or ten years this should be commonplace – at the moment, though, this is not the case and as you will see later in the section, buying a business micro still involves a few potential pitfalls.
Computer programming involves a mixture of intellectual challenge, technical knowledge, and creative flair, although it is difficult to appreciate how much of this last quality is necessary until you are acquainted with programming. But is it really important to learn to program a computer when there is so much software available?

If you are a business user, then the answer to this question is almost certainly no – it is not important in any business connection. Trying to write your own package could just be a waste of time and money and the result, unless you have had lots of programming experience, would certainly not be as good as a professionally produced package.

The opposite is true of the computer hobbyist. Writing your own programs can be one of the main attractions of owning your own computer.

What are algorithms?
Some of the implications of writing a program – a list of instructions – may not be immediately obvious. You cannot, for example, expect the machine to guess what you are trying to tell it to do. Therefore, you must be clear in your own mind about what exactly you want of the machine before sitting down to write a program.

Having decided what you want the machine to do, the next step is to consider how it is going to do it – in other words, the method involved in carrying out a particular operation, and this method is called an “algorithm”. An algorithm is a formal description of the way one of the steps of a flowchart is carried out (although it is not the same thing as a flowchart), and there are many standard algorithms for carrying out all sorts of operations. Often these are expressed in mathematical notation, but there are also those that are described in plain language. Many programmers will have whole sets of algorithms, which they often use in their work.

Algorithms in practice
To describe how you can apply algorithms, we can take a familiar, everyday, non-computer example – making a cup of coffee. If you examine this operation carefully you can see that it can be broken down into several steps:

1. Fill the kettle with water
2. Put the kettle on the stove
3. Light the gas
4. Put the coffee powder in a cup
5. Wait for the water to boil
6. Pour boiling water into the cup
7. Add sugar
8. Add milk
9. Stir

You can see from this that making a cup of coffee actually involves several distinct steps, but each of these could be broken down further. The operation of filling the kettle, for example, breaks down into picking up the kettle, removing the lid, putting the lid down, carrying the kettle to the water, turning the water on, and so on. And several of these steps

**FLOWCHART SYMBOLS**

A flowchart provides a very useful visual representation of the method or "algorithm" used to solve a problem. Although it is not essential to use them, a series of standardized symbols has been devised to represent the various types of operations you will need to be familiar with. These symbols are shown below. A flowchart always begins and ends with a "terminator". In between, the various steps are connected by pathways (shown as straight lines).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Terminator" /></td>
<td>This symbol is used to represent both the start and finish of the flowchart.</td>
</tr>
<tr>
<td><img src="image" alt="Instruction" /></td>
<td>This symbol tells the computer what it should do next.</td>
</tr>
<tr>
<td><img src="image" alt="Manual input" /></td>
<td>The computer requires the operator to key in a piece of information.</td>
</tr>
<tr>
<td><img src="image" alt="Connector" /></td>
<td>Here is the point where pathways intersect.</td>
</tr>
<tr>
<td><img src="image" alt="Decision point" /></td>
<td>The computer will ask a question and will then follow a different pathway according to the reply it is given.</td>
</tr>
<tr>
<td><img src="image" alt="Input/output" /></td>
<td>The computer will either &quot;input&quot; (receive and make a note of) or &quot;output&quot; (print or display on a screen).</td>
</tr>
</tbody>
</table>
Most of us are perfectly capable of making a cup of coffee without analyzing each stage of the process. But trivial though this example may be, it does illustrate how real-world problems can be broken down into their component actions—a necessary first step toward arriving at a computerized solution to a problem. Also, what often seems to be the "basic steps" can themselves be divided.

As you can see from the diagram, the basic operation of making a cup of coffee is the same as the description above, but various symbols have been used at different places. To put things on a formal basis, there are special signs to show the start and end of the process. Each step is described in its own box apart from the place where you wait for the water to boil—this is enclosed in a diamond with an arrow coming out of one side. If you were to think about "waiting for the kettle to boil" you would see that it involves checking constantly to see if it is boiling and deciding whether or not it is. If it is not, you go on waiting, but if it is you can continue with the next step. The diamond shape, therefore, is used to denote a decision-making step in an operation, with "yes" and "no" paths to indicate where the flow of action goes. As you have seen earlier in the book, this type of decision making is something computers can do very well.

It is important to realize that you do not have to draw one of these flowcharts every time you write a program but, if you feel the need to do so, then proper stencils are available for all the symbols necessary. In the sphere of professional programming, algorithms are mostly used for mainframe computer programming, where a piece of commercial software may be in use for a decade or more and may need alterations long after the original program author has left the company. If there still remains a comprehensively drawn flowchart, the job of understanding what is going on is made easier for anybody altering the program.

Planning a program
If a formal flowchart is not always necessary, a planning stage, involving the careful analysis of what the program is to do and how it is to do it, is essential for all but the most trivial of programs. This is so because the planning stage clarifies your thinking and makes the job of actually writing the program code much quicker and less error prone.

A useful way of preparing your program and planning its course of action is to write out a verbal description of what it is to do, including details of
what you want to appear on the computer's screen - which, in some programs, particularly games, may be the most important of all considerations. Suppose you want to write a program that allows you to type in a series of numbers and then print out their average. You could begin by writing a description something like this:

1. Clear the screen
2. Ask the operator to type in a number
3. Add this to the previous numbers (if any)
4. Increase the count of the number or numbers typed in so far by one
5. Ask the operator if there are any more numbers to be typed in
6. If the answer is yes, go back to step 2
7. If the answer is no, divide the total by the number of individual numbers typed in
8. Display the result on the screen

You will notice immediately that you need a way of keeping two numbers in the computer's memory - the total of the numbers typed in and a count of the number of numbers typed in. In Module 27 you will see in detail how this is done. For simplicity's sake here, you can give these numbers a name and let the computer take care of the rest. As an example you can simply call the total TOTAL and the number count COUNT, and use these titles to produce another type of flowchart (see right).

By now you should be able to see the principle behind drawing up a flowchart, and you can practice it even without a computer, by analyzing everyday tasks that you normally carry out without thinking. Chances are that you will find a few surprises - some things will require much more detailed actions and decision making than you might imagine, while you may find that there are other things that you can do in an easier and simpler way.

**Advantages of algorithms**

One of the principal advantages of expressing program methods as algorithms is that you avoid having to use a particular programming language. As you will see in Module 32, there are quite a few programming languages currently in use for different functions, and it could be very inconvenient if you happened to have found a particularly neat and tidy way of doing something, but it was written in one language when you are writing in another. An algorithm is "language independent". In other words, once you have found out the method of performing the task, you can easily translate it into any programming language terms, and that is the topic of the next module in the book.
Almost without exception, every personal computer — home or business — can be programmed in one particular programming language — Basic (Beginner's All-purpose Symbolic Instruction Code). Many machines also allow you to use other languages, but because it is almost certain that your computer has a version of Basic, and because Basic is easy to learn, it is the language you will be using in this module.

There is one major problem with Basic: like many ordinary languages, it comes in a variety of "dialects" or versions but, unlike people, computers cannot understand different dialects. A computer comes with one version of Basic and cannot use programs written in a different Basic. The situation is particularly difficult with home computers as these all have different hardware facilities — screen formats, for example, or sound capabilities — and manufacturers produce their own versions of Basic designed specifically to use these facilities. At the end of this module there is a panel about converting programs written for one machine to run on a different make, but it is a difficult thing to do unless you are familiar with both machines.

It is because of these differences that it is impossible to include here details of how to use these more exciting features of your machine. Instead, the emphasis here is on an introduction to some elementary programming concepts so that you can start writing some simple, but usable, programs.

Fortunately, there is a "core" to the Basic language, comprising features that are found on every machine with little or no variation. And equally fortunately, it is possible to do quite a lot with this core without ever using the bells, whistles, color, and graphics, which represent the machine-specific parts of programming. Once you have mastered the basics, use the manual supplied with your machine to learn the more advanced features.

Basic in practice
To follow this through it is easier if you have a computer in front of you. But even if this is not possible, you can still come to terms with the principles and even write a few programs out on paper — although there is no substitute for trying things out in practice. Mistakes are inevitable, but there is no way that your programming mistakes can damage the computer and, often, you learn more from sorting out where you went wrong than from endlessly studying the theory.

Assuming that you have a computer, switch on and type in the word HELLO. Then press the RETURN key (on some machines it is labeled ENTER or NEW LINE). The computer should reply with a message like SYNTAX ERROR or SN ERROR; it certainly will not say HELLO. You will see quite a few syntax errors while you are learning. All it means is that the computer has encountered something it cannot understand. Like a human language, a computer language has a collection of words that convey specific meanings and which must be used correctly. People can understand slight word mis-uses or mis-spellings in human language because we are good at inferring meaning, even when the rules of language are slightly abused. This is not the case with computers, and you must communicate with them on their terms. This means using only words that are contained in the programming language used by your computer, and using them correctly.

Well, the reason you have just achieved your first syntax error is because the word HELLO is not part of the Basic programming language. Type in the following, exactly as it is printed here, remembering to press RETURN at the end of each line:

```
10 PRINT "WHAT IS YOUR NAME?"
20 INPUT A$
30 PRINT "HELLO, ";A$;" NICE TO MEET YOU!"
40 END
```

Once you have typed this in, type LIST and then RETURN. The computer will print out what you have just typed in. If you have made a mistake, re-type the affected line. Typing LIST again will display the program once more and you will see that your correction has been included. Now type RUN and RETURN — RUN is the way you tell your computer to start executing a program. (If you get a syntax error message you probably made a typing mistake — LIST the program again and check it.) The ? on the second line means that the computer is waiting for you to tell it something, so type in your
name and hit RETURN. The computer will then respond with the programmed greeting. But in order to understand what you have done to obtain this result, you can analyze the program to see what is going on.

Analyzing the program
The first thing to notice is that the lines of the program have numbers: 10, 20, 30, 40. All lines in a Basic program must be numbered because the computer uses them to decide the order in which the lines are executed. The numbers go up in steps of 10 because you may want to go back and add a line somewhere in the program you have already typed. If you wanted, for example, to add a line between 20 and 30 you could easily call it 25, still leaving you scope for further additions. The order in which you type the lines does not matter, as long as you number them in the order they are to be executed. When you type LIST the computer will rearrange the line order according to the numbers. One of the advantages of Basic is that you can alter and amend it. If you were to type:

```
25 REM THIS IS AN EXTRA LINE
```

when you LISTed it you would see that the computer had added the extra line between lines 20 and 30. If, however, you type 25 followed by RETURN and LIST the program again, you will see that line 25 has disappeared. Typing just the number of a line followed by RETURN deletes that line. Now you can go through the program line by line.

PRINT is the Basic word that tells the computer to display something on the screen. Here you want it to display the words WHAT IS YOUR NAME?, and you tell it this by enclosing the words in inverted

**BREAKING DOWN A PROGRAM**

The very simple program you can see annotated below shows how a computer can display information (text, in this example) on the screen, accept information typed in at the keyboard, and then use that information as instructed by the program. In this case, the machine asks you for your name and then uses it to greet you. Although this program is obviously trivial and cannot be used for anything practical, it does illustrate very clearly several fundamental computing concepts, which you can go on to use and develop in programs of your own.

```
10 PRINT "WHAT IS YOUR NAME?"
20 INPUT A$:   
30 PRINT "HELLO, " ; A$ ; " NICE TO MEET YOU!"
40 END
```

- **Basic keyword**
  - PRINT
    - This command tells the computer to display the following information (within the quotes) on the screen.

- **Quotes**
  - Text within these marks after the PRINT command is displayed exactly as it appears in the program listing.

- **String variable**
  - Information typed in after INPUT is assigned to an area memory referred to by a name. The $ tells the computer that it is a string alphanumeric in this case.

- **Line numbers**
  - These tell the computer the order in which to execute the lines.

- **Basic keyword**
  - INPUT
    - This tells the computer to wait for something to be typed in at the keyboard.

- **Basic keyword**
  - END
    - This command tells the computer the program is over.

- **Semi-colon**
  - This prevents the computer from moving on to a new line after PRINTing something.
commas. So, the keyword PRINT followed by some text in inverted commas displays that text on the screen.

```
20 INPUT A$
```

INPUT is the way you tell the computer that at this point you want to type in something at the keyboard, and that what you type is to be used by the program in some way. For the computer to be able to use information, it has to store it in memory. Later, when the program needs that information the computer must be able to find it, so you need to set aside an area of memory and give it some sort of name. This area of reserved memory is called a “variable” because its contents can be changed, and its name is referred to as a variable name. The variable in line 20 is called A and any information typed in, in response to the INPUT command, will be stored in an area of memory called A. The computer also needs to know the type of information to be stored there — either numerical or “string”. A numeric variable is one that contains numbers, while a string variable is one that contains text. You tell the computer what variable type is being input by adding a $ sign to the end of the variable’s name if you want it to be a string variable. Leaving the $ sign off indicates a numeric variable. If you were to retype line 20 omitting the $ after A and tried to type in your name, you would see an error message and the program would stop. The computer would assume that the variable called A is to be a number, so when you type in your name it looks to see if you have typed in a number and rejects the non-numerical information.

If you use the program as it was originally written and type in a number instead of a name, the program will still work. This is because, with a string variable, almost any printable character — including numbers — can be accepted and stored in the memory area referred to by that variable name. But if a number is typed in you will not be able to perform any arithmetic operations on it. This is a valuable feature when, for example, you want to put information like an address, including the street number.

```
30 PRINT "HELLO,";A;" NICE TO MEET YOU!"
```

In this line you have PRINT again, so you must be telling the computer to display something and there is something in inverted commas for it to print out. But this time that something is divided into two parts with an extra bit in the middle. Notice first the semi-colons on either side of $A$. When line 10 was executed, the words WHAT IS YOUR NAME? appeared on one line and then a ? appeared at the start of the next. This is because after the computer has PRINTed whatever follows the PRINT command, it automatically moves to the start of the next line unless you tell it not to. And you do this by adding a semi-colon at the end of the line, outside the inverted commas. If you put it inside the inverted commas, the computer will PRINT the semi-colon, too. So, in this line you have told the computer to display the word HELLO, and then stop on that line. By placing the $A$ in the middle, you tell the computer to PRINT the contents of the memory area referred to by the string variable name $A$, and, to keep the computer on that line, there is a semi-colon after $A$, too. Then there is some more text in inverted commas for the computer to PRINT out — NICE TO MEET YOU! If you were to put the $A$ in inverted commas, the computer would print simply A$ and not the contents of the memory location specified by the string variable name $A$.

```
40 END.
```

The END command is obvious. You are telling the computer that this is the end of the program and that it should stop. Some computers allow you to miss off the END statement, but it is best to include it anyway. Later you will see that you may not want the computer to work its way through every line of a program, so it is possible that the END statement might occur half-way through the actual program listing. To summarize, you can see an annotated listing of the program on the facing page, telling you what each part of it does.

**From algorithm to program**

The first program above illustrated a few points about programming and about the Basic language, but it really did not do very much. In Module 26 you saw how, using algorithms, you could get a computer to accept a series of numbers and work out their average. But how do you go about turning that flowchart into a program? Remember that what you want to do is type in a series of numbers, add them together, keep a count of how many numbers there are, and, when there are no more numbers, work out the average by dividing the total by the number of entries, and then display the result on the screen. It became apparent when discussing the algorithm that you would need a way of storing the count and the total in the program, and from the program above you now know that you can do that with
variables – one for each. You will also need a third variable to act as a temporary store for the number you type in before you add it to the total. And as you have to ask if there are any more numbers to be typed in, you also need a string variable for the answer. The variables you will need for this program are:

- T for the totals of the numbers
- C for the count of the number of entries
- S for the temporary store for each number before it is added to the total

A$ for the answer to the ANY MORE NUMBERS? question

(for the program listing, see facing page).

**What does the listing mean?**

From the listing you should notice several new things. First, lines 10 and 20 - LET T=0 and LET C=0. These mean exactly what they say - they set the value of the two variables (T and C) to zero right at the start - and you must type the figure 0 and not the letter O, just as you must type the number 1 instead of the lower-case letter l. Many Basics allow you to omit the LET part of this sort of program statement.

You know that a variable name is a reference to an area of memory in which the variable's value is stored. When you start the program you do not exactly know what is in those areas of memory - they may have been used by a different program earlier in the session - so you must make sure that they contain the value needed at the start (in this case, 0).

In line 30 there is a short cut. Instead of using a separate PRINT statement to display the "prompt" to the operator, you can use INPUT to perform the displaying by putting the message immediately afterwards in inverted commas, followed by a semicolon outside the inverted commas, and then the name of the variable into which the number is to be placed temporarily. It is not necessary to set the value of S to zero before you start because whatever value you type in as a response to the question will replace any random value that might already be in the memory area called S.

Line 40 adds the number you have just typed in to the total in T. The same applies in line 50, where you count the number of entries by adding 1 to the value of C.

In line 60, INPUT is used again to both display a question and to get its answer. This time you want a string answer, so your variable name has a $ at the end to tell the computer. Having obtained a reply to your question, you have to make a decision based on it – to go back and get another number or to work out and display the average. Line 70 does this with the IF . . . THEN statement: IF the answer is YES, GOTO line 30. GOTO redirects the flow of the program and allows you, in conjunction with IF . . . THEN, to make decisions and carry out alternative courses of action – this is one of the computer’s most powerful capabilities. (Notice, though, that IF . . . THEN is not just used for redirection; you can also perform actions such as IF A=B THEN C=D or IF X=Y THEN PRINT “X EQUALS Y”.)

If you type NO in reply to the question, the program continues to the next line instead of jumping back for another number, and in line 80 the computer prints out the average of the numbers. You use PRINT, followed by text in inverted commas, and then, rather than work out the average, assign its value to another variable, and then PRINT that variable's value, you can instead put in the formula that gives the average and the computer will print the answer. In this case, the average is calculated by dividing the total by the count, and to indicate this you use Basic’s symbol for division – / (an oblique stroke). This prints out the result of T (the total) divided by C (the count). All that is left is to END the program in line 90.

**Altering the program**

Provided that you typed it in exactly as it is printed here, this program will work perfectly on most machines. It demonstrates several important computing concepts, and you should practice making a few changes – different messages, perhaps – and observe their effects.

As a program, though, it leaves a lot to be desired. It is, for example, boring and time consuming to have to answer YES every time you type in a number and want to type in another. A way of avoiding this would be to use another criterion as the basis of deciding whether or not to go back to line 30. You could, for example, decide that the average should only be worked out when 10 numbers have been typed in. This is very easy if you remove line 60 completely and change line 70 to:

```
70 IF C<>10 THEN GOTO 30
```

The <> symbol means "not equal to". It is, in fact, made up of two symbols often used in mathematics – < and > – meaning, respectively, "less than" and "greater than". So what this new line of program is actually saying is "IF C is less than or greater than 10", which is the same as "IF C is not equal to 10". This modified program will now keep asking for numbers until you have typed in 10 of
them, then it will produce their average for you automatically.

There is a serious disadvantage to using this approach — you can only work out the average of exactly 10 numbers, and this considerably reduces the program's versatility. So why not use the numbers themselves as the test basis as to whether or not there are more numbers to be added? You can arrange for the program to keep on asking for numbers until you type in one special number, which it can recognize and use as a signal that you have finished your list of numbers and want their average printed out. About the only number you can use for this purpose is zero, although this would prevent you including zero as a number to be averaged (which could cause problems at some stage). The easiest way to do this is to add a line between lines 30 and 40:

```
35 IF S = 0 THEN GOTO 80
```

**AVERAGES PROGRAM**

This is a simple program that allows you to work out the average of a series of numbers typed by the user. In order to do this, the program has to keep a count of the number of entries typed in — held in the variable C — and also has to total up the numbers. It keeps the total in the variable T and works out the average by dividing T (the total) by C (the count) in line 80. Using this program, you can type in as many numbers as you like and after each number the machine will ask you if there are any more numbers. Any answer other than YES will cause the average to be calculated.

```
10 LET T = 0
20 LET C = 0
30 INPUT "ENTER A NUMBER: " ; S
40 LET T = T + S
50 LET C = C + 1
60 INPUT "ANY MORE NUMBERS? (YES OR NO): " ; A$
70 IF A$ = "YES" THEN GOTO 30
80 PRINT "THE AVERAGE IS: " ; T/C
90 END
```

10 and 20 — Initialize the variables T and C to value 0.

40 and 50 — Add the value in S to variable T and add 1 to variable C.

70 — The text in string variable A$ is tested. If the answer is YES, then the computer goes back to 30.

80 — This prints the text between the quotes on the screen and then prints the result of dividing variable T by variable C.

**MEMORY PROMPTS**

Here are some short definitions of some of the words found in this section.

- **Assembler** A piece of software that converts assembler language instructions (those that use a mnemonic code for the binary instructions understood by the processor) into binary.
- **Basic** The most widely used programming language, found on almost every microcomputer.
- **C** A programming language that allows assembler language control and is often used for writing operating systems.
- **Cobol** A programming language designed and used for writing operating systems.
- **Fortran** A programming language used mainly for scientific and mathematical work.
- **Interpreter** Software that executes a program line by line, rather than turning it all into binary code as a compiler does. Most Basics on microcomputers are interpreted versions.
- **Operating system** Software that takes care of all the "housekeeping" operations in the computer.
- **Pascal** A popular high-level programming language designed for all general-purpose computing work and available on many microcomputers. Unlike most dialects of Basic, Pascal is a compiled language.
and to change line 70 to:

70 GOTO 30

Notice now that you have to make your decision earlier in the program, immediately after the number has been typed in, in fact. Otherwise, the zero, when typed to conclude the program, would be added to the total (which would not matter) and also counted as an entry (which would matter very much to the average). Make these changes and RUN the program; you will find you can enter as many numbers as you like and terminate them by typing 0, whereupon the average will be displayed.

Expanding the program

There are some applications for which it would be useful to store all the numbers you type in and print them out again, or perform some other sort of operation on them as well as working out their average. As the program is written now, once a number has been added to the total you “lose” it because its value, held in the variable S, is replaced by the next number typed in. One way of overcoming this would be to include the numbers in the actual program:

10 A = 254
20 B = 12.065
30 C = 1

These types of entry could go on indefinitely, but it would be tedious and awkward. Another method is to include them in the program as DATA items, like this:

10 DATA 254, 12.065, 1, 99, 3465, 121

There is a special command – READ – that allows you to access any pieces of information held in DATA. Imagine that Basic contains a “pointer”, which points at the DATA items to be read by the READ statement. When you start the program: RUNning, the pointer automatically points at the first DATA item. If your next program line is:

20 READ X

the first item in the DATA list (the number 254 in this case) becomes the value of the variable X and the pointer is advanced to point at the next number in the list. A subsequent line:

30 READ Y

will set Y to the value of the next number in the list and the pointer is advanced again, and so on. If you want to READ these numbers again, there is a special command that resets the pointer, making it point at the first item on the list again – RESTORE. You can see a small program using RESTORE on the facing page.

The RESTORE in line 50 has set the pointer back to the start of the DATA list after lines 20, 30, and 40 had read the numbers into the variables A, B, and C. Lines 60, 70, and 80 therefore read the same values into the variables X, Y, and Z, so X is the same as A, Y is the same as B, and Z is the same as C. Notice also that there is another feature of the PRINT command – the variable names are separated by commas, and as you can see from your display, this has the effect of printing the numbers out with spaces between them (usually eight or ten).

This is still not particularly convenient for dealing with your list of numbers to be averaged. DATA statements are more useful for holding the values of variables that will remain more or less fixed, or which at least need to be “initialized” (or set) to specific values at the start of a program. Should you subsequently want to change some or all of these values, it is much easier to do so by altering the single line containing the DATA statement.

Arrays and loops

Another feature of Basic is that it allows you to create “arrays” of numbers or strings. An array can be thought of as a row of pigeon holes to which you have given a name, say, A. Before you can use an array in a program you must tell the computer how large it is, and you do this with a DIM statement (for DIMension):

10 DIM A(50)

This tells the computer that you are going to use an array called A that has 50 pigeon holes, numbered from 1 to 50. You can refer to each pigeon hole within the array by its number, enclosed in brackets after the name – A(1), A(2), A(3), and so on. But more importantly, you can use a variable to refer to a particular pigeon hole:

10 DIM A(50)
20 X = 2
30 A(X) = 123
This little program will place the value 123 in pigeon hole 2 of array A, because you have set the variable X to the value of 2 and used this to tell the computer which pigeon hole you are addressing. Arrays are extremely useful in programming because of this facility of using a variable to access the pigeon holes. And in Basic you often find arrays used in conjunction with a feature called the FOR...NEXT loop.

As its name implies, a loop is a piece of a program that the computer executes over and over again. Hopefully it will only execute it a specific number of times, otherwise you will have a never-ending program. Here is one example of a loop:

```
10 PRINT "HELLO - THIS IS A LOOP"
20 GOTO 10
30 END
```

RUNning this program will result in the words HELLO - THIS IS A LOOP being printed on the screen endlessly, and you will have to hit the ESCAPE (or BREAK) key to stop it. Line 30 will never be reached because, after printing the words in line 10 the computer progresses to line 20, which simply tells it to go back to line 10 again. Here is another loop:

```
10 X = 1
20 PRINT "THIS IS A DIFFERENT LOOP"
30 X = X + 1
40 IF X < 11 THEN GOTO 20
50 END
```

You will find that this prints the words THIS IS A DIFFERENT LOOP 10 times and then stops. Because of the test in line 40, it only goes back to line 20 if X is less than 11. If X is equal to 11, it continues to line 50 instead.

The FOR...NEXT loop gives you more versatility when you want to carry out an operation many times. Here is a program that will do the same as the one above:

```
10 FOR X = 1 TO 10
20 PRINT "THIS IS ANOTHER LOOP"
30 NEXT X
40 END
```

The FOR...NEXT loop consists of lines 10 and 30 - anything between these lines will be carried out a definite number of times. You determine the number with the FOR statement. Here, the variable

Data required by a particular program can either be typed in by the user or included as an integral part of the program itself in the form of DATA statements. These DATA statements can be positioned anywhere within the actual program (although putting them either at the beginning or end makes them easier to find and alter, if necessary), and they are ignored by the computer until a READ instruction is encountered.
CREATING AN ARRAY

10 DIM A(50) : REM Set up the array to hold 50 numbers
20 LET T = 0 : LET X = 0 : REM Set the variables to zero
30 FOR I = 1 TO 50 : REM This is the start of the loop
40 INPUT "ENTER A NUMBER: "; S : REM Get the number as before
50 IF S = 0 THEN GOTO 120 : REM Zero terminates the input stage
60 T = T + S : REM Add the new number to the total
70 C = C + 1 : REM Count the number
80 A(I) = S : REM Save the number in the array, using I
90 NEXT I
100 REM At this point we have an interesting situation. The loop
101 REM is set up for 50 executions because that's the size of
102 REM the array. If more than 50 numbers are typed in, there
103 REM is nowhere to put them in the array, so if no zero is
104 REM typed in before we have 50 numbers, the FOR...NEXT loop
105 REM must terminate and we have to tell the user what's going
106 REM on with the following message. Note that in line 50 we
107 REM skip over this message because the user has typed a 0
110 PRINT "I CAN'T ACCEPT ANY MORE NUMBERS!"
120 PRINT "THE AVERAGE IS: "; T/C
130 PRINT "AND HERE ARE THE NUMBERS..."
140 FOR I = 1 TO C : REM We can use the number count as the upper
141 REM limit in this loop; that way we won't try to print out
142 REM any unused elements in the array.
150 PRINT A(I)
160 NEXT I
170 END
X was set to the value of 1 at the start of the looping and the next lines executed until the NEXT X statement was met. The NEXT statement will check the value of X against the end value, specified in line 10 by the TO 10 part of the statement. IF X is less than 10, you go back to the top of the loop again and increase the value of X by 1; otherwise the program continues with the statement following the NEXT X statement.

The FOR...NEXT loop is, in this case, controlled by the variable X (although you can use any numerical variable at all, but not a string variable). You can also use variables for the other two parameters controlling the loop – the start and end values:

```
100 FOR A = B TO C
```

This line is quite permissible provided B and C have been given values earlier in the program. Even more useful, you can use the variable controlling the loop within the loop itself. Suppose you want to set all the elements in an array to a particular value:

```
10 DIM A(20)
20 FOR I = 1 TO 20
30 LET A(I) = 123
40 NEXT I
```

This would set every element of array A equal to 123. You can use the variable I to address each element in turn by starting the loop with I = 1 and using I instead of a number inside the brackets that describe which element of A you are referring to. On the second pass through the loop, the value of I will have increased to 2 and you will be addressing the element A(2).

By now you should be able to see that the averages problem – keeping the numbers and printing them out, as well as calculating their average – can be easily solved using an array. There is only one limitation – because you have to DIMension the array first, you do impose a limit on the number of entries you can handle. On the left you can see a program that will accept 50 numbers, work out their average, and print this out followed by the numbers themselves. A few comments have been added to take advantage of two other features of Basic – the REM statement and the multistatement line. REM allows you to add REMarks to your program. These are useful as you can insert little notes to yourself – anything on the line following a REM statement is ignored by the computer. Using the multistatement, which nearly every Basic allows, you can use a colon to the same effect as a line number, so saving space when program writing.

How to write a roulette program
Most versions of Basic allow you to generate random numbers. In fact, though, these are only pseudo-random numbers because the algorithm normally used for this purpose produces a stream of numbers which eventually starts to repeat itself – so it is not quite random. It is, however, random enough for the purposes of writing a roulette program.

As roulette is a game of chance, you need to generate quite a few random numbers. In this game, both you and the computer play against the bank. You enter your bet (both color and number, as well as the amount of your stake), but this is decided randomly for the computer. The result, of course, is also decided randomly by the computer.

You will need to repeat the basic process several times to take account of all the variables, obtaining a random number between 0 and an upper limit, which varies according to what you are doing. If, for example, you are deciding how much money the computer should bet (line 360), the upper limit is the amount of money it has, represented by the variable C. If you want to decide if the ball has stopped on red or black, you choose an upper limit of 100, and it is red if the resulting random number is less than 50 or black if it is more than 50 (line 460).

Rather than type out the random number generating function each time, you can use a "subroutine". You will find subroutines a very useful feature when programming for they allow you to perform the same action repeatedly from different parts of the program. Here, the subroutine is in line 870, and you use it with the command GOSUB 870. The effect of this is to transfer the program flow to line 870 and the computer carries on executing the program there until it comes across a RETURN command. It then leaps back into the program and continues at the point immediately following the GOSUB order. At any point in the program, you can GOSUB to a line number and RETURN to the point where you left off.

There is one trap in all this, though. For the sake of neatness, it is customary to put the sub-routines at the end of the program, as is the case with the program shown here. It is, therefore, necessary to make sure you put an END statement before the subroutine(s) starts, as in line 780. Otherwise the program will "fall into" the subroutine, continue executing whatever code is there, and eventually encounter the RETURN. But as it fell into the subroutine by accident, it will not have anywhere to RETURN to, so the program will crash, with an
ROULETTE PROGRAM

10 REM ROULETTE GAME
20 REM
30 PRINT "PETER'S CASINO"
40 PRINT "=================
50 PRINT : PRINT
60 LET P = 1000 : REM Player's starting money
70 LET C = 1000 : REM Computer's starting money
80 PRINT "We each start off with $1000. The first one to go" 
90 PRINT "broke is the loser. Good luck!"
100 REM
110 REM First, let's ask the player which colour and number
120 REM he wants to bet on, then how much money he wants
130 REM to gamble...
140 REM
150 PRINT : PRINT "Do you bet on Red or Black? "; 
160 INPUT "Type R or B: "; C$
170 IF C$ <> "R" AND C$ <> "R" THEN GOTO 160
180 IF C$ <> "R" THEN C$ = "Red" ELSE C$ = "Black"
190 INPUT "Which number (0 to 25) do you bet on? "; N
200 IF N < 0 OR N > 25 THEN GOTO 190
210 PRINT "You have $"; F' ; " ";
220 INPUT "How much do you bet? "; A
230 REM
240 REM Let's make sure he has that amount...
250 REM
260 IF A > P THEN PRINT "You don't have that much!" : GOTO 210
270 IF A = P THEN PRINT "Going for broke, eh?!"
280 REM
290 REM Now we'll randomly pick a colour, number and amount
300 REM of money for the computer's bet...
310 REM
320 L = 100 : GOSUB 800 : REM Decide whether Red or Black
330 IF L < 50 THEN M$ = "Red" ELSE M$ = "Black"
340 L = 25 : GOSUB 800 : REM Get the number
350 LET M = L
360 L = C : GOSUB 800 : REM Use money left as upper limit on bet
370 LET B = L
380 PRINT "I bet $"; B; " on "; M$; M
390 REM
400 REM Now spin the wheel by deciding another colour and number
410 REM
420 PRINT "The wheel is spinning..."
430 L = 25 : GOSUB 800 : REM Get a number
440 X = L
450 L = 100 : GOSUB 800 : REM Decide colour
460 IF L < 50 THEN X$ = "Red" ELSE X$ = "Black"
470 REM
480 REM This FOR...NEXT loop does nothing - it's here just to provide
490 REM a delay while the wheel "spins"...
500 REM
510 FOR I = 1 TO 1000
520 NEXT I
530 REM
540 REM Print the results!
550 REM
560 PRINT : PRINT"The ball has stopped on...";
570 PRINT X$;
580 PRINT X : PRINT
590 REM Now decide who has won...
600 REM
610 IF C$=X$ AND N=X THEN PRINT "You won!" : P=P+B : GOTO 670
620 IF M$=X$ AND M=X THEN PRINT "I won!" : C=C+A : GOTO 650
630 REM Nobody won...
640 PRINT "Neither of us won that time!"
650 P = P - A
660 IF P < 1 THEN PRINT "You're bust, friend!" : GOTO 730
670 C = C - B
680 IF C < 1 THEN PRINT "I'm broke!" : GOTO 730
690 PRINT "You now have $";P
700 PRINT "I have $";C
710 INPUT "Hit RETURN for another spin...";G$
720 GOTO 150
730 PRINT "Want another go?";
740 INPUT "(Y or N) ";G$
750 IF G$ = "Y" THEN GOTO 60
760 IF G$ <> "N" THEN GOTO 740
770 REM
780 END : REM We need to put this here to stop the program
790 REM "falling" into the subroutine.
800 REM And this is the subroutine which decides a random
810 REM number between 0 and L. Notice that this uses the
820 REM random number function RND(0) and the exact use of this
830 REM might be slightly different on your computer. If you
840 REM syntax error messages or results which aren't random,
850 REM check in your computer's manual to see if the RND
860 REM function has a different syntax.
870 RANDOMIZE : LET L = INT(RND(0)*L)
880 RETURN
890 REM
900 REM And that's the lot!
error message such as RETURN WITHOUT GOSUB IN LINE ... This can cause some confusion at first.

There is not much more to say about the roulette program – it is more a matter of you trying it out – except that you can learn a lot about the subtleties of programming by experimenting with games like this. As you become more familiar with your own computer, you can start to incorporate refinements, particularly graphics (a spinning wheel and ball, perhaps) and possibly even the sound of the roulette ball if your machine has a sound facility. Without doubt, the program as presented could be improved considerably in programming style – it has been kept deliberately simple for the sake of clarity. About the only way it resembles real roulette is that it is almost impossible to win anything.

Program conversion
As has already been mentioned, different computers use different “dialects” of Basic. This is one of the main points of frustration for computer hobbyists, because most computer magazines print program listings, and the most interesting ones always seem to be for machines other than the one you own. Yet, when you try to convert a listing you find all sorts of strange graphics symbols, weird commands, and other puzzling messages, which not only will not work on your machine but also give no clues at all as to their actual purpose.

The two areas in which you will find the greatest difficulties are those which are most machine-specific – screen handling and sound. There is no easy way to convert the graphics written for one machine to run on another. Each machine has different numbers of lines and of characters per line and different graphics resolutions, too. You will also find that a machine with good graphics facilities has commands like FILL or PAINT, yet another type of machine may not have these – or has them but uses them in very different ways. Some machines will allow you to “see” from within a program what is printed at a particular location on the screen, while others will not. Likewise with sound – usually you have to send a series of values to a sound generator chip and unless you can study the other machine’s manual to find out whether it uses the same sound chip as yours, you probably will not be too successful when trying to convert.

Another big problem arises in the use of PEEK and POKE. These cause much confusion to those new to computers but they are, in fact, quite simple. PEEK allows you to examine the contents of a specific memory location, usually given in decimal rather than hexadecimal, and is particularly useful with memory-mapped input/output (see Module 20) and memory-mapped screens. So, X=PEEK (12345) will assign to the variable X whatever value happens to be in memory location 12345 decimal.

POKE does the exact opposite. It allows you to place a value into a specific memory location and again is used to give faster-changing displays on systems with memory-mapped displays or to output something to a memory-mapped I/O port. POKE 12345, 100, for example, would place the value 100 into memory location 12345. You have to be careful when you are POKEing things directly into RAM like this as it is easy to get the address wrong and start overwriting the area that holds your program.

Hopefully at some time in the near future, manufacturers will get together and agree on a more standard version of Basic. One standard has been agreed by the American National Standards Institute but, as it still has not been formally finalized, few manufacturers are interested in using it. Until a more rational day arrives, the only practicable way of converting a program to run on your machine is to buy a copy of the other machine’s manual and to work through it on a trial-and-error basis.

Improving your Basic knowledge
This module has been designed to teach you some of the elementary aspects of programming in Basic so that you can start off with a basis for experimenting and trying out different things.

One of the best ways of learning to program is to think of something you would like to do and then try to do it with the computer, learning from your own mistakes as you go along. Unfortunately, many home computers come with fairly inadequate teaching manuals. They are prepared usually in a great rush as the last process in launching the machine on to the market. It is possible to go further than this brief outline of some of the features of Basic, but the wide variations in the different dialects of Basic make this impracticable here. If you find that the manual with your computer is not of much help when exploring the more advanced features of the machine, then all is not lost. Almost every popular microcomputer now on sale has at least one book explaining its features in more detail than you will find in the manual. Another source of help is computer magazines, especially if there are any dealing specifically with your machine. Finally, you will almost certainly find that there is a computer club within easy reach, and with luck it might be one specializing in your make of computer. Joining a computer club will put you in touch both with experts who can help you and with other novices with whom you can share your knowledge.
A “bug” is a fault of some kind, either in the computer hardware or in the software program that will not work properly. And the process of rectifying a fault is known as “debugging”.

What can go wrong?
When your computer suddenly stops doing what you think it should, or stops working altogether, it is very tempting to blame the hardware immediately – it is, after all, the part of the system you see and touch. But, more often than not, it is the software that is at fault.

Even the most experienced programmers make mistakes and when you are first learning to program yourself mistakes are inevitable. Some of these errors will be easy to find, but there is always going to be the elusive variety, and tracking these down is simply a matter of going over the program listing time and time again.

There are two types of software bugs: syntax errors and logic errors. Of the two, syntax errors are by far the easier to find, especially as the majority of computers will detect them and throw up on the screen a suitable error message, such as “SYNTAX ERROR IN LINE 1240”, for example. Some computers will display an error code, which you can look up in the user’s manual.

Syntax errors
Generally, syntax errors are due to simple typing mistakes (“PRIMT” instead of “PRINT”, for example), and they are both easy to make and to rectify. Some typing mistakes, though, are more difficult to find: “/” (the symbol for division) instead of “*” (the symbol for multiplication) would cause some interesting and difficult-to-find errors if it was hidden somewhere within a long program listing.

Another type of syntax error is the result of using a Basic word incorrectly. Take, for example, line 870 of the Roulette program in the previous module (see Module 27):

\[
\text{LET } L = \text{INT(RND(0)*L)}
\]

It is very easy in this sort of expression to type the brackets incorrectly – either missing one out or putting one in the wrong place – and this is, by no means, easy to spot. Or how about this example:

\[
3520 \text{ LET } X = \text{INT(SQR(Y/(3*X)))*3: IF X=0 THEN GOSUB 4000 GOTO 4420}
\]

You will see on the screen the message SYNTAX ERROR IN 3520 from this and the chances are you will spend a lot of time worrying about all those brackets before you realize that there is a colon missing before the GOTO 4420 part of the line near the end of the entry.

“Falling into” subroutines can cause absolute chaos. It is easily done, too – either by missing out the END statement before a group of subroutines at the bottom of the listing or, even worse, by missing the RETURN statement off one subroutine. This causes you to fall into the next subroutine below it and execute that one, returning only when you hit the RETURN at the end of that one, by which time all sorts of interesting things could have happened for no immediately obvious reason.

While you are in the process of writing a complex program, you will almost certainly be adding and deleting lines throughout the listing. During this process it is surprisingly easy to erase a line to which you have, elsewhere, specified a GOTO or GOSUB. Again, this will be reported when you run the program and the computer finds that it is supposed to GOTO a line that does not exist, but it is annoying, nevertheless. For example, take a program statement like:

\[
120 \text{ IF } X = \text{INT(Y)} \text{ THEN GOTO 500 :}
\]

\[
500 \text{ REM This is a comment}
\]

\[
510 \text{ LET } Y = \text{SIN(J)}
\]

If you are running out of memory space with a long program, you can save some space by deleting all of the REM statements. In this case, with line 500 there, it does not matter that you GOTO it from line 120 – any line beginning with REM is ignored by
the computer, and the program itself will in this case carry on from line 510. But if you subsequently delete line 500 as part of a REM purge, you will see an error message when you try to GOTO it.

Most Basics are quite good at reporting syntax errors, with messages of varying degrees of helpfulness. These range from a curt error code (which you have to look up in a manual) to (in at least one case) a display of the offending line with the error marked for your attention. What your computer cannot do, however, is detect an error in the logic of a program.

**Logic errors**

Logic errors are those in which you have the syntax perfect but you have told the machine to do the wrong thing. A simple example of this type of mistake would be to ask the computer to PRINT the value of the variable X when you really meant X$. It can be immensely frustrating when you know what you want the computer to do and you are certain that you have programmed it correctly, but the machine will still not do it. The computer, obviously, has no way of knowing or guessing what you are trying to do. It merely follows the instructions you have given it and—assuming there are no syntax errors in your program—will do exactly as it is told. If you have given it a program that contains logic errors but is perfect in its syntax, it will execute it—but as far as you are concerned, the result will not be correct.

There are several ways out of this situation, although none of them is necessarily easy or simple. Multistatement lines, for example, can cause some problems. Look at this:

```
1730 LET X = Y: IF X < Z THEN
   X = Z: GOTO 1500
```

With many forms of Basic, the last statement in this line will only be executed if the previous statement (IF X < Z THEN X = Z) is true. If you wanted to GOTO line 1500 regardless of whether X is less than Z, but X happens to be equal to or greater than Z, you will not get to line 1500. In this case you would be safer to write:

```
1730 LET X = Y: IF X < Z THEN X = Z
1740 GOTO 1500
```

This is true if going to line 1500 regardless is what you want. You should test the behavior of your Basic (assuming it supports multistatement lines) with a single program like this:

```basic
10 A = 1
20 B = 2
30 IF A = B THEN C$ = "NO": GOTO 50
40 C$ = "YES"
50 PRINT C$
60 END
```

If running this program produces "YES" on the screen, you could easily fall into this trap with your Basic. A "NO" result means that the rest of a multistatement line containing a conditional statement is executed regardless of the outcome of the condition.

The other major problem is to ensure that variables are given the values they are supposed to receive while the program is being executed. You can usually check this by inserting extra lines to print out the value of any suspicious variables in areas where you expect trouble. You can also insert END or STOP commands at appropriate places. This is slightly cruder but it does take advantage of Basic’s interactive nature by asking it to print the variables’ values directly from the keyboard.

Finding out if the computer is executing the right part of the program can sometimes be difficult. Some machines have a TRACE facility. By typing TRACE ON or TRON before you RUN the program you will see a display of the line numbers as the program is executing. Usually, if your Basic allows it, it is more convenient to put TRON and its opposite, TROFF, around the section you suspect is not being reached. If no line numbers appear on the screen while the program is running, you know the section is not being used. If you do not have TRACE facilities, you can simulate them by inserting messages at the start of each section of the program: PRINT "NOW IN RANDOM NUMBER SUBROUTINE", for example. Do not forget, however, to take them out again once the program is working correctly.

The process of debugging is made easier if you take care with the design and layout of your programs. Start by drawing up at least a written flowchart (see Module 26), which divides the program into functional sections and describes what each is supposed to do. And take care over the design of screen displays—what is to appear on the screen and the choice of color or graphics, for example. When you actually write the program, keep to these sections as far as possible and separate them with a few blank REM lines. And begin each section with one or more REM statements that describe briefly what the section does and what variables it uses or affects elsewhere in the program.
In the early days of computing, the computers themselves were too rare and expensive to run for their time to be taken up with things like playing games - at least that was the official line. The games that were written for the large computers were disappointingly boring when compared to what is available today for micros. For a start, the terminals connected to the mainframes and used by the programmers and operators did not (as is still usually the case) have graphics and color capabilities - things taken for granted on many home machines. So the games tended to be entirely textual.

The micro boom
Once microcomputers began to catch on two things happened: first, more and more people began to write games; and second, the machines themselves were given extra facilities such as color, sound, and graphics, which could be exploited to good effect in games as well as with other types of software.

Computer games are now such big business that they constitute a considerable part of the micro industry. Some of the largest microcomputer manufacturers owe their entire existence to computer games: Atari, for example, made its name with video and arcade games and then went on to produce a range of home computers with some of the best graphics available. For technical reasons, it is not possible to reproduce the high-resolution graphics of an arcade machine’s display on a home computer.

Games development
So, who then is writing all these games? Many are written by talented, and usually young, freelance program authors - and the financial rewards can be massive. But at the other end of the business, large companies have set up whole departments to do nothing but write new games.

Once the outline of the game is developed, professional artists design “storyboards” - a series of cartoons showing the screen displays at every stage of the game. At the same time, the rules and logic

Space Invaders It all started with Space Invaders - ranks of sinister little aliens advancing slowly down the screen while the player manipulated a gun at the bottom of the screen trying to shoot them down before they landed. A massive range of this type of game is now available, taking full advantage of the fast color graphics and, often, sound provided on most home computers. Although quality is not as good as that on arcade machines, home versions are fast catching up. The scenarios for these games are always pretty much the same: anonymous and undesirable aliens advance in waves, and you have to get them before they destroy you with their weapons. The better games allow you to progress through various levels. At the start things are relatively simple, but as soon as you have destroyed every alien, a fresh, meaner wave appears. The result of all your efforts is only a score that disappears when the machine is switched off.
Chess is probably one of the most complex of games and has proved extremely challenging for programmers to put on computers. At one time it was thought that computers could never beat humans. Now, chess programs on large computers are giving Grand Masters a very tough game indeed.

The mania for zapping aliens really took off with the introduction of Space Invaders — sinister ranks of bomb-dropping aliens slowly descending on the screen while the player moves a gun along the bottom of the display, trying to destroy them before being destroyed by them. As an arcade game Space Invaders became a world-wide craze and spawned a whole host of imitators both in the arcades and on home machines. Games like the original Space Invaders have long since been overtaken by more sophisticated ideas, but the basic principle remains — strange entities have to be eliminated before they eliminate you. The better games progress to levels of ever-increasing difficulty once you have destroyed all the opposition at one level, and usually there is a complex point-scoring system.

The third type of computer game — board games — has found considerable application in serious computing research. In particular, chess, one of the most complex of all games, has been used extensively in research into artificial intelligence by researchers with access to some of the most powerful computers available. Squeezing a chess program into a micro, though, does involve making some compromises concerning power and the way computers actually work out which move to make. But increasingly clever programming techniques have produced some very powerful chess programs. Other types of board game have, unfortunately, proved less interesting to play on computers and have not been particularly successful, with the exception of sophisticated versions of backgammon and checkers (or draughts).
For some time, computer graphics were considered an interesting research subject but without much practicable application by the rest of the computer industry. The wide-spread appearance of the micro changed this attitude.

Graphics at home
On a home computer, good graphics in color are now considered essential, and there is a constant battle between manufacturers to provide more and more sophisticated graphics. One disappointment is that home computers cannot (at present) compete with the arcade machines in terms of graphics quality. There are technical reasons for this, mainly to do with the use of domestic television as a display screen. It is necessary to convert the computer’s video signal to a television signal and this involves sacrificing some of the original signal’s resolution. The term resolution refers to the number of lines in the display and the number of dots making up each line – the more lines and dots, the higher the resolution. Some home computers have direct video outputs and special very-high-resolution graphics modes that can be used with a video monitor.

Graphics in the office
Graphics are being used increasingly on business computers, especially now that 16-bit machines, with their extra memory capacity and computing power are now more commonplace. For example, some of the better machines now offer a graphics resolution of 400 lines, each of 800 dots, which is high by micro standards.

Color, too, is becoming important in the business world, although in this market there is money to spare and color business machines generally use proper video monitors. Originally regarded as a gimmick, color on a business machine is rapidly becoming essential as more and more people realize its advantages as an extra information channel, especially when combined with graphics. A screenful of figures produced on a spreadsheet, for example...
(see Module 35), becomes more meaningful when transformed into a bar chart or graph. Many software companies are now producing integrated information-handling packages that allow figures to be presented in a number of graphics modes as well as in tabular form.

**Graphics and memory**

Producing a color graphics display is relatively easy and cheap now that memory chips are both inexpensive and have higher capacities. The normal method now is to reserve a "bank" of RAM — usually within the CPU's main memory area on a 16-bit machine, or as a separate unit addressed through I/O ports on an 8-bit machine.

Normally the job of actually driving the display is done by a dedicated microprocessor called a video controller. The video RAM is so arranged that each bit corresponds to a dot on the screen — to light a dot the CPU places a 1 in the appropriate RAM location, and the dot is switched off by replacing the 1 with a 0. This is a process called "bit mapping" (see Module 20).

Graphics software is becoming very sophisticated, and many home computers now incorporate powerful graphics commands in their Basic interpreters, allowing you to draw and fill in circles, for example, or produce your own character sets if you do not like the ones preprogrammed into the machine. The problem is that manufacturers produce their own commands, which, like the dialects of a spoken language, contain differences in the way they are used. This, combined with the different color ranges and graphics resolutions, makes converting a Basic program for one home machine for use on another very difficult.

**Future graphics**

Producing a television-quality image entirely by computer is not possible on micros — yet. Research now being carried out into computer graphics points to what you may see in the future on these machines. The process of describing an object mathematically so that its "shape" can be stored and manipulated is extremely complex. Producing a display from this information calls for immensely complicated mathematics, but the results can be astounding: a "human" hand, with unnaturally smooth, flawless skin, moving slowly about, the fingers flexing just as they would in real life. Another example is fighter planes flying in perfect formation under the control of the computer's joystick. The real world, however, is full of imperfections, and the biggest problem in re-creating pictures with a computer is to introduce these imperfections.
In Module 27 you saw how a computer can be programmed in Basic. Basic is one among a family of languages referred to as high-level programming languages, so called because they allow you to write instructions using English-like commands. These are easy to remember but they do need to be "translated" by either an interpreter or a compiler into the binary code used by the computer. One advantage of most high-level languages is that they allow you to take a program written and tested on one machine and use it on another, completely different make of machine, provided the second computer also has an interpreter or compiler for that language.

One problem with these languages is that the resulting program is not necessarily as efficient as it could be. An interpreted language program tends to run rather slowly, while a language written in a compiled language, although faster, may take up too much memory.

Program portability
It is possible with most computers, to write programs using the binary instructions of the computer directly, and this results in a very compact, very fast program. The problems are that this is tedious and time consuming and, more importantly, the program is not portable — unless you have written the program to run under one of the standard operating systems such as CP/M (see Module 33), you could have problems transferring it to another make of computer. But there are still some applications for which low-level programming is essential, especially for time-critical routines or for creating ROM-based software when you only have a limited amount of memory space.

What is assembler language?
Writing any program of more than a few dozen bytes in machine code is virtually impossible to do accurately and quickly. Trying to write in binary is out of the question, and even writing in hexadecimal is far from satisfactory. To overcome this there is assembler language, an abbreviated language in which each binary instruction is represented by a mnemonic description. You write the assembler program using a text editor or word processor, save it on tape or disk as a text file, and then instruct a piece of software called an assembler to read the file and turn it into binary machine code that the computer can run directly.

Before you can start programming in assembler language you must study the processor for which you are writing and, often, certain hardware details of the computer system, such as the addresses of the I/O ports, for example. Most processors have their own, incompatible, instruction sets, therefore you have to learn the set of each processor before you can program it. In this module, the Zilog Z80 is being used as the example. If your computer has a different processor the code will not work, but the general principles will still apply and, generally speaking, converting it to run on your processor should not be too much of a problem.

Inside the Z80 processor
Studying the processor before trying to write in assembler language involves examining its internal "architecture". The Z80 contains several "registers" or internal memory locations. There are, in fact, 22 registers, the most important of which are duplicated in an alternative register block and a special instruction exists to allow you to change between the two blocks if you want to (see Module 17). It is not necessary to say anything more about the alternative registers, as anything you can do to the main block applies to the alternative register block as well.

Each of the registers in the main block can hold one byte and each is given a name. The principal register is known as the "accumulator", and it is in this one that most of the main operations take place — it is usually necessary to move a byte into the accumulator before performing an arithmetic or logical operation on it, for example.

Associated with the accumulator is the flags register. Unlike the other registers, the flags register is a collection of one-bit memory cells, two of which are not used. The ones of greatest interest at this stage are the "carry flag" and the "zero flag", which, respectively, show whether the operation just performed in the accumulator left a carry or set the accumulator's contents to zero. These flags are used, by the presence of various instructions, to
decide a course of action. If, for example, subtracting a number from the accumulator leaves a zero, jump to another part of the program, otherwise continue with the next instruction.

The remaining six registers in the main block are general-purpose registers - B, C, D, E, H, and L. As well as being used as 8-bit registers, however, these can also be used in pairs to hold 16-bit values and to perform various operations on them. When the registers are used in this way they are referred to as BC, DE, and HL. As an example of this in practice, HL can be loaded with a memory address and this can then be used to obtain a byte from that address and to load it into the accumulator.

For the moment, the only other register of interest is the “stack pointer” or SP. The stack is an area of memory set aside for specific uses. One of these comes into action when you want to use a subroutine. You saw in Module 27 how, if an operation is to be performed many times, you can write it as a separate section of a program and use it as many times as necessary with the GOSUB command. A similar facility exists in assembler programming, but you CALL a subroutine rather than GOSUB to it. When a CALL instruction is given, the processor saves the memory address of the next instruction after the CALL by placing it on the stack. The stack operates rather like a spring-loaded plate-stacking mechanism found in many self-service restaurants: when you add a plate to the pile the others move down one, and removing a plate causes the whole pile to lift up slightly. This principle is mimicked in the computer by the stack pointer. It contains the address of an empty memory location that can act as a temporary store. You can PUSH a byte on to the stack, in which case the stack pointer is adjusted automatically to point to the next available memory location and the value copied into that location. When a subroutine is CALLED, the processor puts the address following the CALL instruction on to the stack and then begins to execute the subroutine. When it reaches the RETurn instruction at the end of the subroutine, it simply POPs the address off the stack and continues execution from that address. In many programs, initializing the stack pointer (by loading it with a memory location for stack use) is almost the first thing you have to do. It should be noted that in most processors, including the Z80, the stack pointer is set to an address somewhere near the top of the available RAM area and then “grows” downward from there.

Assembler programming in practice
It is now time to look at an example of Z80 assembler programming. On the right you can see a short listing in the standard style used in assembler programming. The columns used are (from left to right): label, opcode, operand, and comment.

A “label” is a symbolic name, usually for an address in memory. There are four examples in the program here – PUTIT and GETIT, for example, are the names of the two subroutines. Rather than have to work out the exact address in memory of these two subroutines, you refer to them in your main program by their names when you CALL them, and when you actually come to write the subroutines you start by labeling them with those names. The assembler then works out the actual addresses and inserts them in the CALL instruction.

An “opcode” is, quite simply, assembler jargon for an instruction of some kind.

An “operand” tells the computer what the opcode is referring to. For example, in the first line of the program proper – the one labeled START: - the opcode is LD (mnemonic for LOAD) and the operand consists of a register name (HL) and a memory address (BUFFER). In other words, this line tells the CPU to load the register pair HL with the address of the label BUFFER, which is defined as an area of 80 bytes by the last line in the listing - BUFFER: DEFS 80. Not all opcodes require an operand – the RETurn instruction at the end of the subroutines, for example.

A “comment” is inserted in the program purely for the programmer’s benefit and works in exactly the same way as the REM statement in Basic. Instead of REM, though, it begins with a semicolon but, just like REM, anything following it on the line is ignored by the assembler. Actually, comments can begin in any column and this is useful for breaking up the program into easily identifiable sections, as you can see for yourself.

A few general points need to be made about this program before looking in detail at what it actually does. Because assembler programming is so machine-specific, the program has been written to run under the CP/M operating system (see Module 33). If your computer is a non-CP/M machine, the program will need some modification before it will work. And, the first two lines of the program are special: they are not part of the program but are instructions to the assembler itself. The first (CR EQU, ODH) assigns the value OD hex (the ASCII carriage return code) to the name “CR”. This means that you can use CR throughout the program when you wish to refer to the ASCII carriage return code. This should make understanding the listing a little easier. And the ORG 100H statement tells the assembler the address in memory at which the program is to start. As this is a CP/M program,
Software · Module 31

ASSEMBLER LISTING

ORG 100H ;The address at which the
CR EQU 0DH ;ASCII carriage return.

; This is the input section of the program

START: LD HL,BUFFER ;Here's where we start, by
; loading HL with the address
; of the memory area where the
; characters are to be stored
LABEL1: CALL GETIT ;Call the subroutine which
; gets a character typed at the
; keyboard.
LD (HL),A ;Store the character.
INC HL ;Increment HL to point to the
; next memory location.
CP CR ;Was the character a RETURN?
JR NZ,LABEL1 ;Jump back to get the next
; character if not.

; This is the output section

LABEL2: LD A,(HL) ;Restore HL to point to the
; first character.
CALL PUTIT ;Call the subroutine which
; displays the character.
INC HL ;Increment HL to point to the
; next character.
CP CR ;Was the last character a RETURN?
JR NZ,LABEL2 ;Go back and get the next
; character if not.

; The program ends here, in this case by returning to CP/M

; This subroutine to get a character typed at the keyboard.
; This is for CP/M-based systems so you'll need to re-write
; this if your computer doesn't use CP/M.

GETIT: PUSH BC ;The C register is used by
; this subroutine and it's
; good practice to PUSH any
; registers used in a sub-
; routine onto the stack to
; preserve their contents in
; case these are needed by the
; main program.
LD C,1 ;The CP/M function number to
; get a character from the keyboard
CALL 5 ;Call the CP/M subroutine to
; read the keyboard - the char
; is returned in A.
PUSH BC ;Save these registers because
; the subroutine uses them.
PUSH DE ;The CP/M function number to
; display a character on the screen
LD E,A ;CP/M requires the character
; to be placed in the E register so we have to
; transfer it there from A
CALL 13 ;Call the CP/M routine to
; display a character.
PUSH DE ;Now restore those registers
POP DE ;but remember that the stack
;operates on the "last in,
; first out" principle so we
;must POP them in reverse
;order or we'll just swap
;their contents!
;Back to the main program.
RET ;Here's where we define the area to be used as a character
;store, called BUFFER

BUFFER: DEFS 80 ;DEFS is an instruction to the
;assembler, not the computer,
;and reserves an area of
;memory (80 bytes in this
;case) of which BUFFER is the
;address of the first byte.

; And that's the end of the program!
it has to start at 100 hex, but your computer may require a different starting address.

Analyzing the program
The program itself is extremely simple, but it does illustrate several important points. In essence, it accepts a series of characters typed in by the user at the keyboard and stores these numbers in an area of memory called BUFFER. The program will continue to accept characters until the user types carriage return, and it will then simply print out all the characters on the screen.

The parts of the program that you may have to alter to suit your particular machine are the two subroutines (GETIT and PUTIT), which, respectively, get a character typed in at the keyboard and put a character on the screen. GETIT waits for a character to be typed in and then places it in the accumulator (this is done automatically by a CALL to CP/M); PUTIT requires a character to be in the accumulator, which it then prints on the screen.

The program starts off by loading the HL register pair with the address of the memory area where the characters are to be stored. This area is BUFFER. It then calls the subroutine GETIT, which returns the character in the accumulator, or A register. The address in HL is then used as a “pointer” (this is signified by the use of brackets around HL in the listing) and the character in the accumulator is transferred to that memory location. The address in HL is then “incremented”. This means that you add 1 to it — so that it points to the next free location in BUFFER. Now you check to see if the character which was typed in was a carriage return, with the line CP CR. This compares the contents of the accumulator with the value of “CR”, which, as you saw, was defined as ODH (the ASCII carriage return code) at the start of the program. If the answer is that the two are equal, the zero flag is set. The following instruction tests the zero flag: JR NZ, LABEL1 means if the result is non-zero (that is, they are not equal), jump back to the address specified by the label LABEL1, and continue from there. So, if the character was not a carriage return, the computer will go back to the CALL to GETIT and wait for another character to be typed in.

If, on the other hand, the character was a carriage return, the program continues to the printing out to the screen section. Here, you begin by restoring the contents of HL to the address of the BUFFER. You then use HL as a pointer again, but this time you load the contents of that address into the accumulator, with LD A,(HL). Then you call PUTIT to print this on the screen and increment HL to point to the next character. You now test the contents of the accumulator to see if it is the carriage return that signifies the end of the string of characters. If it is not a carriage return, you jump back in the program to get the next character from memory, just as you did in the input section.

The program is ended by an instruction that returns the computer to CP/M control, although your machine — if it is not CP/M based — could well require a different address here. There is a difference between the JP and JR jump instructions. JP is followed by a 16-bit address (zero in this case, but it could be a label) and can, therefore, redirect program control to any location in the computer’s entire memory. JR, on the other hand, stands for “Jump Relative” and uses an 8-bit “offset”. The computer “knows” the memory address of the instruction it is currently obeying (it has to so that it can fetch the instruction from memory) and it adds or subtracts the offset to this address to work out the location of the instruction to which it must jump. Its range is therefore more limited (up to 126 bytes before the instruction or to 129 bytes after it with the Z80). But many loops require jumps well within this range and using JR rather than JP saves one byte of memory each time and results in quicker code, too.

Non-assembler machines
This is all very well if there is an assembler for your machine, but many home computers do not have this capability. In this case you have the task of hand-assembling your program and converting each opcode into hex. This done, and the relative jumps worked out, you usually find that it is necessary to convert the hex into decimal (special calculators are available or you can write a simple program in Basic), include them as DATA statements within a Basic program, and POKE these into a suitable area of memory. Generally, there is a Basic command such as USER(nnn), in which nnn is the decimal address of the start of the machine code subroutine to which control is passed. Although tedious, this is worth doing if there are parts of your Basic program that are too slow, particularly in a fast-moving game where Basic might not be able to handle the screen fast enough. Unfortunately, the details of how to do this sort of thing vary so much that it is impossible to go into this subject in any great depth here. Your computer manual should, however, give you more information, and if your machine is one of the popular home computers, you will probably find that there is a book already written explaining the intricacies of this type of advanced programming.
Computer programming is an odd art. A problem exists that needs to be solved, and it has been decided that the solution can and should be mechanized by programming a computer to do the job. The problem and the method of solving it, therefore, must be described in terms that the computer can understand and act on, yet the description, at least initially, must be performed by humans. There exists, obviously, a communications gap. Terms that humans can understand (and use to describe the problem) are unintelligible to the computer, and the language of the computer is difficult or even impossible for a human to digest in any but the most trivial quantities.

Why different programming languages?
Programming languages exist to fill this communications gap. The programmer is confined to a special, limited vocabulary, which the computer can translate into terms it can understand. By analyzing a problem and then describing its solution in this intermediate language, the programmer can make the computer tackle the problem, yet the description (or program) is still in a form that makes sense to anyone versed in that programming language.

The process of designing a programming language, therefore, comes down to analyzing the sorts of things people want to do with computers and writing generalized routines that perform these tasks. Many programs need to display information on the screen, for example, so a word like PRINT can be used in the language to describe the routine that does this. Adding something to qualify it increases its versatility—you at least need to be able to tell it what to print.

Because computers are so versatile, the range of tasks that can be carried out on them is enormous. In turn, this means a wide range of facilities have to be provided if everybody’s needs are to be catered for. Attempts have been made to accomplish this by creating one massive language, requiring a lot of effort for the programmer to learn and the software to translate into machine-usable form. A language of this magnitude is difficult to implement on large machines and impossible on microcomputers.

For this reason, there is a variety of computer programming languages, none of which could be described as the perfect language, but each designed with a particular type of work in mind. Basic, for example, was designed as an easy-to-learn teaching language which would allow non-computer people to use a computer without having to spend a lot of time learning a very specialized new skill first. Pascal was designed as a general-purpose, rigidly structured language with the aim of teaching good programming habits by forcing practitioners to adhere to strict rules. Cobol was originally intended as a layman’s language, but is very much a programmer’s tool for business applications.

In this module you will be looking at some of the popular programming languages, all of which are available for microcomputers. Each has a different slant, a different area of work for which it is best suited. You would not, for example, try to write a word processor program in Cobol if you had access to a C compiler; monitoring laboratory equipment and gathering data from it would be better handled in Fortran than in interpreted Basic, particularly if speed of execution was important.

It is an oddity of programming that each language attracts its own ardent supporters. Programming methods also attract equally fervent support: one school of thought supports the structured approach, in which the entire program must be carefully divided into neat, self-contained sections with control flowing uninterrupted through each section in an orderly fashion. This approach does make a lot of sense but, unfortunately, does not always work out in practice.

The idea behind this module is to give you a flavor of some of the languages, pointing out strengths and weaknesses, and commenting on interesting features, rather than trying to teach you how to program in each one. With the exception of C, all the languages are well documented with “how to” books, and, should any language seem attractive, you will have no difficulty in finding a suitable book on it. Finding a compiler or interpreter for your computer may, however, prove more difficult.

In general, learning a second programming language is a lot easier than learning your first one. Basic is often condemned for a variety of reasons, but it does have a tremendous advantage as a first language—it is very easy to learn and use. You can,
therefore, become acquainted with programming concepts quickly and painlessly, giving you a good grounding to learn another language, if necessary.

C

C is probably the most versatile programming language available for microcomputers. It stems from a British-produced language called BCPL. This language was adapted at the Bell Laboratories in the States to form a language called B and this, in turn, was improved and modified to produce the language we now know as C.

The language C was designed for use with the Unix operating system but it has been successfully implemented on many other operating systems on both mainframes and minicomputers as well as on micros. It is probably the most portable language available. You can write a C program on any computer and transfer the source code (the text file containing the English-like commands that the compiler translates into machine code) and then type it straight into another, entirely different computer with almost 100 per cent confidence that it will work without any modification. The only exceptions are a few very machine-specific sections such as those concerned with screen handling.

At first sight, a program in C appears incomprehensible. Although it does contain some English commands with fairly obvious meanings, it also uses the entire ASCII character set to abbreviate many expressions. Once you begin to learn the language, though, it does start to become clear, and you soon begin to appreciate the brevity of C. Programs can be written in what almost amounts to shorthand notation, which is a refreshing change from languages such as Cobol or Pascal.

C occupies a unique position in the hierarchy of languages. It is a high-level language but, at the same time, allows you to work right down to the bit level, just as though you were working in assembler (see Module 31). It is, therefore, an ideal language for developing systems software (see Module 35). Most of the Unix operating system is written in C and the company Digital Research (which produced CP/M), now uses C for all its operating systems and language compilers and interpreters.

In some ways, C is similar to Pascal. It is a structured language and encourages a modular approach to programming, which makes listings much easier to understand. It is, however, far less authoritarian than Pascal and does not impose the strict rules about program layout that Pascal requires. C also has a less severe and far less verbose syntax than Pascal and the resulting freedom it gives is probably one of the reasons for its popularity – although its portability and versatility are its chief appeals.

A C program consists of a main module and a number of functions, the latter being equivalent to Basic’s subroutines (see Module 27). Apart from the requirement that the main module must be identified by calling it “main”, there is none of the insistence that Pascal enforces as to the position of the main module within the program itself. In Pascal, each function must be defined before it can be used – so a Pascal listing starts with the “lowest” function and ends with the main module. While C, on the other hand, allows you to place them in whatever order you prefer.

All variables used in a module must be defined at the start of that module, and their type must be declared. The types supported are integer, character, and floating point and, on most machines, long and short integer and floating point types must also be further supported. C has all the usual arithmetic operators – +, -, * (multiplication), and / (division) – as well as a “modulus operator” (this provides the remainder after a division), but does not provide the “transcendental” functions found in many other languages, such as sine and log. Instead, it has a range of logical operators, which allow you to work with entire numbers or at the bit level: AND, OR, XOR, for example.

One of the best aspects of C is the economy of expression it allows you. Consider, for example, the following (the “*...*/” symbols are used to identify comments, which is C’s equivalent of Basic’s REM):

```c
int x; /* The variable x is declared as an integer... */
 x = 1; /* ...and given the value 1 */
 x = x + 10; /* x is incremented by 10 */
```

This could be written as:

```c
int x = 1; /* The value of x is defined as it is declared */
 x += 10; /* As a quicker way of saying x = x + 10 */
```

The equivalent of Basic’s IF ... THEN ... ELSE is provided with the if statement, used as follows:

```c
if ( x = 20 )
 x = 1;
else
 x = 0;
```
In C, a number of program lines may be grouped together by enclosing them in the { and } symbols, and these are then treated as a single program statement. For example, all the statements in a program module - main or function - must be enclosed in braces. You can also use them in an IF statement:

```c
if ( x = 20 )
{
  x = 1;
  y = 2;
}
```

C has three ways of controlling loops:

```c
for ( x = 1; x <= 10; ++x )
  y += 6;
```

This is rather like the FOR . . . NEXT loop in Basic. \( x = 1 \) sets the initial value of \( x \); \( x = 10 \) is a test (denoted by the use of the \( = = \) operator) to see if \( x \) has reached the value of 10. If it has, execution of the loop stops; \( ++x \) increments \( x \) by 1. This statement is the equivalent of Basic's FOR \( x = 1 \) TO 10, and the test to see whether another iteration of the loop can be performed is carried out at the top of the loop.

```c
x = 1;
while ( x != 10 )
{
  y += 6;
  ++x;
}
```

The part of the program above is another way of performing the same loop, with the condition being tested at the top again. Note the \(!=\) test, which means "not equal". In this case, the "for" construction is tidier, but there are occasions when the "while" loop is more convenient. And sometimes you want the test at the bottom of the loop:

```c
x = 1;
do {
  y += 6;
} while ( ++x != 10 );
```

Here, the test is carried out after each iteration of the loop, with \( x \) being incremented by \( ++x \).

Values are passed to functions by placing them within the brackets following a function name; if the function requires no values, the brackets are left empty but must still be there. C allows you to use the function call as though it were a variable itself, and you can do this wherever you use a variable. It is quite common in a C listing to see a conditional statement in which one or more of the conditions are function calls or even "nested" function calls - a value being passed to a function in the form of a call to another function.

An oddity of C is that the language itself contains no I/O facilities whatsoever. This seems a tremendous limitation at first. What use is a program that can neither accept nor emit any form of data? Of course, I/O facilities do exist, but they are in the form of function calls rather than words within the language itself. A standard library of I/O functions is provided with each C compiler and this is a major clue to C's portability. By grouping all the I/O facilities together, and standardizing their names and syntax, the language frees the programmer from ever having to worry about the I/O facilities of the machine or operating system being written for. C has the usual keyboard, screen, and printer I/O plus all the necessary functions for handling files, including random access files. A neat feature of C is that the I/O can be redirected either within a program or from the operating system when the program is run, so that a program (the output of which normally goes to the screen) can be diverted easily to send its output to a printer or a disk file.

Probably the most confusing aspect of C concerns arrays and pointers. C allows numeric arrays like any other language, but strings are regarded as arrays of single characters and handled as such. There is no provision for handling entire strings, and this is one of its shortcomings. Unlike many languages, C gives you access to an element of an array not only through a number or variable denoting its position within the array, but also via its actual address in memory, using a "pointer". There is not sufficient space here to explain pointers fully, but they have three general effects: they create unreadable listings, they cause chaos while you are learning to use them, and they provide a faster, more compact and efficient code once you have mastered them.

C is a practical language, but it shares one feature with all other languages - it is not suitable for all computing tasks. It would, for example, be possible to write a business application in C but there are other languages, like Cobol, which are designed for this purpose and are, therefore, more suitable. Like-
wise, many scientific and mathematical applications would be difficult to implement in C because of its lack of transcendental functions mentioned earlier. C excels at the task for which it was originally designed - producing operating systems and some types of applications (particularly word processors), where the versatility of assembler language is needed. C gives that versatility while offering the convenience associated with a high-level language.

Forth

Forth is an oddity among programming languages. At first sight it appears unfriendly, inflexible, and difficult to understand, learn and use. It is, however, one of those things that repays the time spent on getting to know it. When you are familiar with Forth you will realize that it is extremely powerful, very fast, and, because of its unique ability to accept extensions to itself, probably the most flexible of all the programming languages.

Forth was invented by Charles Moore in the USA as a result of his dissatisfaction with conventional programming languages. He felt that the traditional languages simply did not provide the power, ease, or flexibility that he wanted, so he set about devising his own, using an IBM 1130 computer. At that time, the 1130 was described as a "third generation" computer and, as Moore's new language seemed so powerful, he decided to call it "Fourth". Unfortunately, the 1130 only allowed five-character identifiers, so he was compelled to call it Forth instead.

Forth is rather more than a language: it is an "environment", a combination of language and operating system. At its center is the "dictionary", a collection of commands or words. A major contribution to the power and flexibility of Forth is that users can add their own words to the dictionary and incorporate these into the language. As an example, we can go through the procedure for adding a word to Forth. The Forth word "n EMIT", in which n is the decimal value of an ASCII character, causes that ASCII character to be displayed on the screen. If you type "42 EMIT" you get an asterisk on the screen, as 42 is the decimal ASCII code for an asterisk (see Module 13). You can define a new word that will print any number of asterisks on the screen simply by typing "n STARS", like this:

```
: STARS 0 DO 42 EMIT LOOP CR ;
```

The colon tells Forth that the following word is a new one, which you are about to define. You can see the "42 EMIT" embedded in the loop structure that causes it to repeat for the number of iterations specified when you type STARS. The "CR" causes Forth to type a carriage return after printing out the asterisks, and the final semi-colon terminates the definition. After typing in this definition, you can then type, say, "10 STARS" and Forth will print out a row of 10 stars.

You can use your new words as the basis for defining further words, building up levels of complexity. Try this example:

```
: STAR 42 EMIT ;
: STARS 0 DO STAR LOOP CR ;
: LINE 10 STARS ;
: GAP STAR 8 SPACES STAR CR ;
: BOX CR LINE GAP GAP LINE CR ;
```

You have already seen the first two lines. Now you have used these words to define further words. In the third line you define "LINE" to print a line of 10 stars. "GAP" produces a star, then eight spaces, using the existing Forth word "SPACES", followed by another star and carriage return. Finally, you define "BOX" to cause Forth to print first a carriage return, then a line of 10 stars, then two lines (each of a star, eight spaces, and a star), then another line of stars, followed by a return at the end. Then typing "BOX" will produce the image below:

```
**********
*     *
**********
```

This is obviously not the most serious of applications, but you should be getting an idea of how powerful this facility is. Each Forth implementation is supplied with an Editor that allows you to create Forth programs in the form of screenfuls of new word definitions. Having written your program, you can test it interactively from Forth and, once it is completed, you can compile it and add it to the Forth dictionary. It is possible to build up an extremely complex program in this way, one that becomes an integral part of Forth and which can be activated by typing a single word.

Because all new words are compiled when added to Forth (whether or not you add them to the dictionary), execution is extremely fast. A compiled Forth word will not run as quickly as the equivalent code written in assembler, but Forth circumvents this by allowing you to incorporate sections of as-

130
I could type something like this:

```
PRINT 3+4
```

and get the answer 7. Forth, however, uses a different sequence of mathematic operations, called Reverse Polish or postfix notation. (The type of notation used in Basic and many other languages is called infix notation.) The same operation in Forth would be typed:

```
3 4 +
```

in order to produce the answer 7. What happens is that Forth accepts each number, 3 and 4, and pushes them on to the stack. The “+” operator tells it to perform an arithmetic addition on the two top numbers on the stack, so it pops them off, adds them, and prints out the result. You can see this at work in the BOX definition above. Typing “42 EMIT” causes the number 42 to be pushed on to the stack; “EMIT” then grabs the number at the top of the stack, treats it as an ASCII code, and displays that character. Likewise, SPACES pops the first number off the stack and uses that as a loop counter to print that number of spaces on the screen.

Working in postfix notation is at first rather confusing (unless you have a Hewlett-Packard calculator, which also uses postfix notation), but it is a lot easier if you take the time to understand how the stack works and what happens when you ask Forth to carry out an operation using the stack.

The stack is also used in Forth (and in many other languages) for passing values from one operation to another. Sometimes, an operation will have pushed values on to the stack in an order that is different to the order required by another operation. Anticipating this, Forth provides words to manipulate the contents of the stack. You can make a copy of some-thing on the stack without disturbing the stack’s contents, or you can jump down the stack, ignoring what is on top, in order to access something further down. You can also reorder the contents of the stack. Naturally, it takes practice to carry out these operations, and unbelievable chaos can result if you get it wrong.

Many users of Forth are, in fact, enthusiasts and hobbyists, and many Forth implementations are on microcomputers and small minis. But the language now also has a large professional following and has been used to implement some interesting applications, including database management, word processing, controlling film cameras, and even operating an entire observatory.

**Pascal**

Various attempts have been made to produce a programming language that would be all things to all programmers. Nobody yet has succeeded, but Pascal probably comes closer than most others—at least in the micro world—to being a genuine all-purpose language. Pascal was designed by its originator, Nicholas Wirth, following his first-hand experience in defining an “all things” language called Algol 68. Algol 68 was defined by a committee, of which Wirth was a member, as the general-purpose language, and the result was a vast complex language which has not gained particularly wide acceptance.

After this experience, Wirth felt improvements could be made, and so set about designing an improved language incorporating many major Algol 68 features. The result of this is a language with considerable power, which can also be easily implemented on small computers—including micros.

Algol, as its name implies, was intended to allow the programmer to develop an algorithm and then translate this easily into a program. Pascal continues this approach and imposes a very strict structure on the resulting program to ensure the programmer adheres to this principle. As a result, Pascal has gained wide favor in academic circles as a teaching language: students learn good programming habits simply because Pascal forces them to develop and use a particular approach. In addition, Wirth tried to make sure that the language was as portable as possible, although others, subsequently, have developed the language and extended various of its facilities. This has, somewhat, eroded its portability. Currently, its most widely used implementation in the micro world is that developed by the University of California at San Diego, known as UCSD Pascal—there is even an operating system based around this implementation called the UCSD
A computer can do nothing until it is given a list of instructions, called a program, telling it in minute detail exactly what you want done (see Module 27). The process of writing a computer program involves breaking down a task into its basic steps and then describing each step in terms that the computer can understand using a computer language such as Basic. By changing the program you can make the computer perform a new, entirely different task.

**Operating systems**

Just about every program contains a number of common operations. For example, most programs require input from the keyboard at some stage, and this input has to be displayed on a screen to allow the user to check that the typed response is correct. The computer must therefore be programmed to accept characters typed in at the keyboard and be told how to display these on the screen.

It would be time consuming if you had to repeat these types of instructions every time you wrote a program. Although keyboard and screen handling are relatively easy on many computers, operations involving disk handling are complex. Also, the way in which these are handled varies with each computer since every machine has certain features unique to its own make. Therefore it would be necessary to rewrite large portions of each program every time you wanted to run it on a different type of machine.

To overcome this consistency problem, a special type of software known as "systems software" or "operating systems" has been devised. Simply, this is a collection of subroutines that takes care of all the basic "housekeeping" functions, such as keyboard and screen handling as well as disk handling when relevant. When a manufacturer wants to produce a machine using a standard operating system, all that is necessary is to alter the parts of the program that are machine-specific. Depending on the design of the computer and the particular operating system, this could be just changing the addresses of the I/O ports. All a programmer needs then do to obtain data from a file or display a character on a screen, for example, is to activate the appropriate subroutine in the operating system. Once the operating system has performed the task, the computer continues with the program.

Apart from the saving in time and effort provided by a standard operating system, all programs written for such a system can be guaranteed to work on any computer equipped with that system.

**CP/M**

The most widely used microcomputer operating system is called CP/M – Control Program/Microcomputer – and is produced by Digital Research in the United States. Originally, CP/M was produced for computers based around the Intel 8080 CPU (see Module 17). The 8080 was replaced by the Zilog Z80 chip, which is much more powerful but also includes the 8080 binary instruction set as part of its instruction set. Therefore, CP/M, which was the only standardized operating system available at the start of the micro boom, could also be used on Z80-based systems, making it very popular. Although popular, CP/M is not particularly "user friendly". It is, however, a standard and there is a massive amount of software available.

Operating systems play an important role in microcomputing, where the user has direct and total control over the machine and is, increasingly, a person with little or no specialist computing knowledge. The operating system is usually the first thing the user comes across after switching on the computer. At present, microcomputing operating systems are still a long way from presenting the user with a clear, self-explanatory, and helpful interface with whatever applications software (see Module 35) is running.

As an example, when a CP/M system is first turned on and the operating system loaded, the user generally sees a short copyright message followed on the next line by "A >". This by itself means very little until it is explained that the "A" means that the computer is currently using disk drive A and that the " >" symbol is CP/M's way of telling you to type in a command. The technical term for this "A >" message is the "system prompt".

CP/M (in common with other operating systems) has two types of commands. Some are built into the operating system itself – they form part of the CP/M program in RAM – and allow you to perform various
surface on which it moves, usually a large sheet of paper on the floor (or any other convenient flat surface). The turtle is connected to the computer by a long cable and its movements can be controlled by typing in commands at the keyboard.

Logo is an interpreted language so that each command is obeyed as it is entered by the user. Take, for example, the line:

\[
\text{FORWARD 20}
\]

This would move the pen forward an arbitrary 20 units of distance (specified by the user).

\[
\text{RIGHT 90}
\]

This would turn it 90 degrees to the right.

The commands PENDOWN and PENUP do exactly what they suggest, so that the turtle can be made to draw lines as it moves. The sequence:

\[
\begin{align*}
\text{PENDOWN} \\
\text{FORWARD 30} \\
\text{RIGHT 90} \\
\text{FORWARD 30} \\
\text{RIGHT 90} \\
\text{FORWARD 30} \\
\text{RIGHT 90} \\
\text{FORWARD 30} \\
\text{PENUP}
\end{align*}
\]

would cause the turtle to draw a square.

A REPEAT command allows the user to repeat actions a specified number of times. But, like Forth, Logo allows a set of actions to be defined, given a single name, and included in its vocabulary. So, the lines:

\[
\begin{align*}
\text{TO TRIANGLE SIDE} \\
\text{REPEAT 3 [FORWARD :SIDE RIGHT 120]} \\
\text{END}
\end{align*}
\]

define a procedure called TRIANGLE, which allow you to draw a triangle, the sides of which are of the length SIDE. Typing TRIANGLE 50 would then produce a triangle with sides 50 units long. The TO part of the definition tells Logo that the following word is the name of the procedure, while the SIDE part of the definition denotes that a variable is to be used, called SIDE. The second line is Logo's loop command - anything within the square brackets is repeated the stated number of times. In a different procedure, of course, the number of times the action is to be repeated could itself be a variable, typed in with the procedure name just like the 50 with TRIANGLE mentioned above. The :SIDE within the square brackets indicates that the value of SIDE is to be substituted here to tell the turtle how many units of distance it should be moved. RIGHT 120 turns the turtle through 120 degrees.

As you will by now probably appreciate, this ability to extend Logo's commands by defining new ones in terms of existing ones, makes the language very flexible indeed. Experiences from researches working with subjects from four to 18 years of age have shown that, even at the lower age ranges, they very quickly grasp the implications of this. At the upper age ranges, they were defining quite complex procedures with practically no trouble at all and, in fact, were producing remarkably intricate patterns with Logo and the turtle.

Naturally, not every program will work first time, and this poses an additional challenge to the child. The first attempt at a solution to the problem did not work, so what other solutions might there be? What is wrong with the present solution that prevents it from working? These are the types of question that encourage the child to examine the alternatives rather than simply look for the “right” solution. “Success” and “failure” are, hopefully, replaced by the challenge to solve the problem.

The robot turtle is an extremely useful aspect of Logo learning, especially with younger children. From experience, they seem to relate to it easily, sometimes adopting it as some sort of pet. It is, however, rather a slow device and takes up more room than is usually available (as well as using large amounts of paper). It does also tend to be expensive, which is its greatest failing.

Because of these drawbacks, “screen turtles” are very common. These are especially popular on home computer Logo implementation, and take the form of a small triangle on the screen, which the user can move about just as though it were a “real” turtle. This alternative is not only quicker, cheaper, and more economical (especially of paper), but it also allows you to erase any areas you do not want, thereby opening up the possibility of adding a degree of animation to the design on the screen.

Logo contains much more than just the ability to draw lines, however. Its artificial intelligence roots show in its list processing capabilities, and some implementations offer graphics capabilities, allowing the user to define “sprites” (or special graphics characters), which can then be used to build up pictures on the screen.
A computer can do nothing until it is given a list of instructions, called a program, telling it in minute detail exactly what you want done (see Module 27). The process of writing a computer program involves breaking down a task into its basic steps and then describing each step in terms that the computer can understand using a computer language such as Basic. By changing the program you can make the computer perform a new, entirely different task.

Operating systems

Just about every program contains a number of common operations. For example, most programs require input from the keyboard at some stage, and this input has to be displayed on a screen to allow the user to check that the typed response is correct. The computer must therefore be programmed to accept characters typed in at the keyboard and be told how to display these on the screen.

It would be time consuming if you had to repeat these types of instructions every time you wrote a program. Although keyboard and screen handling are relatively easy on many computers, operations involving disk handling are complex. Also, the way in which these are handled varies with each computer since every machine has certain features unique to its own make. Therefore it would be necessary to rewrite large portions of each program every time you wanted to run it on a different type of machine.

To overcome this consistency problem, a special type of software known as "systems software" or "operating systems" has been devised. Simply, this is a collection of subroutines that takes care of all the basic "housekeeping" functions, such as keyboard and screen handling as well as disk handling when relevant. When a manufacturer wants to produce a machine using a standard operating system, all that is necessary is to alter the parts of the program that are machine-specific. Depending on the design of the computer and the particular operating system, this could be just changing the addresses of the I/O ports. All a programmer need then do to obtain data from a file or display a character on a screen, for example, is to activate the appropriate subroutine in the operating system. Once the operating system has performed the task, the computer continues with the program.

Apart from the saving in time and effort provided by a standard operating system, all programs written for such a system can be guaranteed to work on any computer equipped with that system.

CP/M

The most widely used microcomputer operating system is called CP/M – Control Program/Microcomputer – and is produced by Digital Research in the United States. Originally, CP/M was produced for computers based around the Intel 8080 CPU (see Module 17). The 8080 was replaced by the Zilog Z80 chip, which is much more powerful but also includes the 8080 binary instruction set as part of its instruction set. Therefore, CP/M, which was the only standardized operating system available at the start of the micro boom, could also be used on Z80-based systems, making it very popular. Although popular, CP/M is not particularly "user friendly". It is, however, a standard and there is a massive amount of software available.

Operating systems play an important role in microcomputing, where the user has direct and total control over the machine and is, increasingly, a person with little or no specialist computing knowledge. The operating system is usually the first thing the user comes across after switching on the computer. At present, microcomputing operating systems are still a long way from presenting the user with a clear, self-explanatory, and helpful interface with whatever applications software (see Module 35) is running.

As an example, when a CP/M system is first turned on and the operating system loaded, the user generally sees a short copyright message followed on the next line by "A>". This by itself means very little until it is explained that the "A" means that the computer is currently using disk drive A and that the ">" symbol is CP/M's way of telling you to type in a command. The technical term for this "A>", message is the "system prompt".

CP/M (in common with other operating systems) has two types of commands. Some are built into the operating system itself – they form part of the CP/M program in RAM – and allow you to perform various...
simple tasks. You can change to another disk drive by, for example, typing "B: " and hitting the return key. Usually the B drive clicks and hums briefly and the system prompt changes to "B>", to show that you are on the B drive. Typing "DIR" and hitting RETURN will display the directory of the disk drive in use. The directory contains the names of the programs and data files stored on the disk, although CP/M, unlike some operating systems, gives you no idea of the size of the files or how much space is left on the disk. If you are using the A drive and want to see what is on the disk in the B drive without actually changing disk drives, you can type "DIR B: ".

Various other CP/M commands come in the form of separate programs that are kept on disk and run when required. These include utilities to perform operations such as formatting a new disk (see Module 22), transferring a copy of CP/M on to it, and copying other files from one disk to another.

HOW DOES CP/M WORK?

CP/M itself is a piece of software that resides on a specially reserved area of a disk. When a computer is first turned on, a "bootstrap ROM" automatically loads the operating system into RAM and then hands over control to it, whereupon CP/M displays its famous "A>" prompt and waits for the user to give it a command.

The diagram on the right shows how a typical 64 kbyte, 8-bit, CP/M-based system's memory is organized. It is called a "memory map" and shows how the computer's memory is divided up and dedicated to different purposes. The numbers on the side are the hexadecimal addresses of the start of the different sections. The bootstrap ROM resides outside of the main system's memory — when the power is switched on, an area of RAM is switched out and the ROM switched in. Once CP/M has been loaded, the ROM is switched out and ceases to be used, giving the processor access to the maximum possible 64 k of RAM.

At the bottom of the map is a small area between locations 0 hex and FF hex which is reserved for use by CP/M. CP/M itself resides at the very top of the memory. How much memory is used by the CP/M varies. When CP/M is installed in a new system, one section of CP/M must be "tailored" to the system. The level of tailoring depends on the hardware details of the computer — it may mean altering the I/O port addresses only or it may mean more extensive adaptation. Whatever is involved, there is a set number of standard functions that have to be provided in a set way. These functions provide all the basic I/O for the system, such as receiving characters typed in at the keyboard, displaying characters on the screen, and operating the disk drives. This section of CP/M is called the BIOS (Basic I/O System).

The rest of CP/M comes ready to use and sits in the memory immediately below the BIOS. It includes the Console Command Processor (CCP) and the Basic Disk Operating System (BDOS), which handles the way information is stored on disk. The exact amount of memory taken by the whole of CP/M depends on the size of the BIOS, and it is generally between 8 and 10 kbytes.

The area of memory between location 100 hex and the start of CP/M is called the Transient Program Area (TPA). It is in this area of RAM that programs are loaded by CP/M and any part remaining can be used by the program for storing data. When CP/M is started up, whatever is typed in at the keyboard is considered a command. The CCP accepts each character as it is typed and places it in a reserved area of memory until RETURN is typed to indicate the end of the command. The CCP then examines what has been typed to see if it is an "intrinsic" command (one that is part of CP/M, such as DIR). If it is not intrinsic, the CCP assumes it is the name of a program and the BDOS attempts to find, load, and run it. If the program exists it is loaded into RAM, starting at location 100 hex, and CP/M tells the processor to "jump" to that location and begin to execute. If the program does not exist on the current disk, CP/M will retype the command followed by a "?" and show the prompt.

So that CP/M can tell a program from, say, a text file, each one on a disk must be given a name of up to 8 characters plus an extension name of up to 3 characters, separated from the main name by a ".”. You could, for example, then give all text files the extension "LET" and all programs "COM". To activate a program you type its name but not its extension, followed by RETURN. CP/M then looks for a program of that name with the extension "COM", which identifies it as a program.
You can run any program on the disk drive currently in use by typing the program name—CP/M will load it into memory and set it running.

Compared to other operating systems, CP/M has the major feature of simplicity of use. Although it has quirks—the utility to copy files from one disk to another is called "PIP" not "COPY", for example—most users master it fairly quickly. Its big failing, however, is that when things go wrong it offers no aid or explanation to help the user. Suppose, for example, that you try to copy a file from one disk to another without first putting a disk in the second drive. You will see on the screen a "BDOS ERR: BAD SECTOR" message from CP/M, which in itself means very little and seems to appear whenever any problem occurs.

Alternative operating systems

Various attempts have been made to improve on CP/M, with varying degrees of success. A company called Cromemco, for example, produces its own CP/M-like operating system called CDOS (Cromemco Disk Operating System) for use on its own machines. This system is much "friendlier" than CP/M. For example, it is possible to "write protect" a floppy disk by sticking a tab over a special notch in its protective envelope. This allows you to read any files on the disk and to run any programs, but prevents you from writing to the disk. This alleviates the danger of accidentally altering or destroying any information on it. Under CP/M attempting to, say, copy a file on to a write-protected disk would give the message "BDOS ERR...". CDOS, on the other hand, displays a much more helpful "Diskette in drive B is write-protected" message.

Various "menu" programs have also been produced to provide a friendly "face" to CP/M. The idea is that when you first turn on the system, instead of seeing on the screen just the system prompt, you are given a list of all the commands and programs on the current disk in the form of a menu and you select the one you want by typing in its number or moving the cursor to the one you want and hitting RETURN.

Although CP/M is the most widely used operating system, it is not the only one available. It will only run on machines with certain CPUs (see Module 18) and machines using different ones may run neither CP/M nor any of the software written for it. Some non-CP/M microcomputers can, however, be converted to CP/M by adding an extra board containing a Z80 processor and, usually, extra RAM. The Apple II, for example, is based on the 6502 processor which, because it has an entirely different binary instruction set to the 8080/Z80, cannot use the CP/M system.

System compatibility

When the 8-bit microprocessor was the only type available, operating systems were a simple matter. The emergence of 16-bit processors has, however, complicated things considerably. One problem is that chip manufacturers do not bother to make their processing is a software trick to divide the CPU's attention between different tasks to give the impression of it performing several jobs simultaneously. This technique is successful on large powerful computers, but not so effective on microcomputers, especially 8-bit machines in which the processor simply is not powerful enough for the task.

Concurrent CP/M-86

If 8-bit processors cannot cope with multiprocessing, then 16-bit micros offer more hope. Rather than follow traditional mini and mainframe practices and allow several users to share the power of one CPU, it makes far greater technical and economic sense in the micro world to give each user a CPU. There are times, however, when it is useful to perform more than one job at a time (printing a text file while editing another, for example), and this is what Concurrent CP/M-86 is designed to do. It is too early yet to say how successful it is at doing this, but it is a more sensible approach than multi-user micro systems.

CP/NET

This system provides the link between a computer and a networking system. Each user has a screen, keyboard, memory and CPU, but all the machines are linked together to share the expensive components such as hard disks and printers. Linking them together requires an interface to handle communications through the network—and this is the role of CP/NET.

---

**THE CP/M FAMILY**

CP/M-based computers virtually dominate the world of business microcomputing. There is now a whole family of CP/M operating systems that evolved from the original 8-bit CP/M developed in the late 1970s. The existence of a standardized operating system did much to fuel the growth of the micro industry and this in turn boosted the popularity of CP/M.

**CP/M-80** The name of the current version of the original 8-bit CP/M. The operating system has undergone several revisions since its introduction, and the version now in general use is Version 2.2. It is now without doubt the world's most widely used 8-bit micro operating system.

**CP/M-Plus** Originally called CP/M-3, this is the latest version of 8-bit CP/M but is not yet widely available. It offers considerable improvements over CP/M-80 and appears much friendlier. Because of the popularity of 16-bit microcomputers, the 8-bit models will probably become cheaper and CP/M Plus will no doubt prove as successful as its predecessors.

**CP/M-86** This is the 16-bit version of CP/M-80 and is widely available on the new generation of 16-bit micros. From the user's point of view CP/M-86 appears almost identical to CP/M-80.

**MP/M-80 and MP/M-86** These are "multiprocessing" versions of CP/M-60. Multi-

---

136
16-bit processors upward compatible with their 8-bit models - in other words, they provide no way of allowing their 16-bit chips to operate with 8-bit binary instruction codes. Zilog, for example, having dominated the 8-bit market with its Z80, produced a very powerful 16-bit chip called the Z8000, but made no attempt to include the Z80 instruction set within the Z8000 and, as a result, lost a potentially valuable marketing edge. In the same way, the Intel 8088, now the most popular 16-bit chip, cannot run software written for Intel's earlier 8-bit processor, the 8080.

16-bit operating systems

There has been a positive aspect as a result of this non-compatibility. Because a huge amount of 8-bit software will not work on 16-bit machines, system software companies have had to come up with new operating systems and there are now several available for 16-bit micros. Unfortunately, there has still not emerged an operating system that can be described as suitable for the fast-growing army of non-specialist computer users. Three microcomputer operating systems are at present competing for the 16-bit micro market - CP/M-86 (the 16-bit version of CP/M), MS-DOS, and Unix. There is a fourth contender, the UCSD p-System, trailing behind the others.

CP/M-86 is so similar to the 8-bit CP/M operating system that, for all practical purposes, it is identical as far as the user is concerned. You cannot, of course, use 8-bit CP/M software under CP/M-86 control, so there seemed little point in retaining the user-unfriendly appearance of 8-bit CP/M. CP/M-86 runs on 8088-based computers and, therefore, also on those using the compatible, but more powerful, 8086 chip.

MS-DOS is produced by a company called Microsoft. It first appeared as PC-DOS, the operating system for the IBM Personal Computer and is now available for most 8088- and 8086-based computers. At first sight MS-DOS looks like CP/M-86 but, in fact, it is different both in the way it works and in the way it is used. It is also much friendlier than CP/M-86, with easier-to-use commands and more recognizable error messages to help you correct any mistakes. One of the drawbacks of using MS-DOS is that, unless you own an IBM PC, there is a more limited range of software available compared to CP/M-86. So far, the IBM PC has sold well in the United States but not in Europe. This has led to the situation where American users of MS-DOS have plenty of software available for their IBMs while Europeans, using other 16-bit machines, find themselves virtually confined to CP/M-86 packages. But things are moving so quickly that this situation could change soon.

The semiconductor manufacturer Motorola makes what is generally regarded as the most powerful 16-bit microprocessor in use, the 68000. Naturally, the 68000's instruction set is completely different to that of the 8088 and 8086 and, therefore, requires its own operating system and different software. Most 68000-based systems use an operating system called Unix, or close derivatives of it. Unix was originally developed to provide computer programmers with a powerful and easy-to-use "environment" in which to develop software. It is a versatile operating system for computer professionals but is, unfortunately, unsuitable for hobbyists or business users - the very people who will be using microcomputers most. It is possible to give Unix a friendlier appearance and some manufacturers have done this. The real problem with Unix, however, is that there is very little applications software available for it. Being designed as a software-development system originally for use in a basically technical environment, it has simply not received the commercial use for which other types of operating system have been specifically designed. Unix is found mostly in universities, colleges, and software-development houses; because of this, although more Unix-based systems will probably be produced in the future, its popularity with non-experts does not seem likely.

The UCSD p-System differs from the others above in that it can run on any computer, and that programs written using it on one machine can be used without change on another. This has been achieved by using a "pseudo-code". Unlike the compilers that are used for other operating systems, p-System language compilers do not produce machine code for the particular system's processor. Instead, they produce an intermediate but quite efficient code, which is translated as the program is run by the core of the system - the p-code translator. To install a p-System on a new processor, all that has to be done is to rewrite this translator core in the new CPU's machine code.

Despite the wide range of processors for which the p-System is now available, and despite its very obvious advantage of complete software "portability", the p-System has yet to be really accepted by the majority of software producers. One possible reason for this may be that its appearance to the end user is still that of a programmer's system rather than one designed for the non-expert, and it does seem to find its most enthusiastic support among system programmers - not a good sign for the amateur market.
One of the largest and fastest-growing applications for computers today is the storage of information. All sorts of information is stored on computer: tax records, health and employment information, financial data, criminal records, car ownership and bibliographical information are a few examples.

**Database management systems**

There is very little point in putting all this information into a computer if you cannot get it out again easily. And if you are using a computer anyway, you may as well take advantage of its capabilities for sorting through data, comparing things, and taking decisions based on the results of these comparisons.

A large amount of software is currently available to perform this “intelligent filing” of information, and is called “database management systems”.

A database management system allows you to enter information in a structured way, store it in the computer, and then retrieve and examine it. To understand how a database system allows you to perform these tasks, it is helpful to look at a few examples of the more typical microcomputer database applications presently in use.

For the first example, assume that you run a small business and you have, say, 1000 clients covering a large geographical area. Suppose that you decide to introduce a new product to your range (the Mk IV) and you want to pass this news on to your clients in the hope that some of them will want to buy one. In this situation, you could prepare a standard letter, have 1000 copies printed, and send one to each person on your mailing list. Or you could, using a computer and database system, include extra information and produce a more sophisticated and worthwhile mailing:

Name: John Smith
Address: Main Street
Town: Newtown
Occupation: Accountant
Greeting on letters: John
First purchase: 1976 Item: Mk II
Last purchase: 1983 Item: Mk 111

Now there may well be quite a few people like John Smith who have recently purchased models of this product and who, therefore, will not be interested in buying any more just now. Also, not all of your clients are interested in or have a use for this particular type of product – you know, for example, that there are a number of plumbers in your database, and plumbers never buy them, so it would be pointless sending them a letter.

So, with your computer switched on, you put the disk containing the database program in one disk drive and the disk containing the database itself in the other, and use the search facility provided to ask for all the people who have not bought this product in the last six months and who are not plumbers. The computer will then search through all the...
dBASE II

dBASE II is probably the world's largest-selling microcomputer database system. This software system allows the manipulation of small- and medium-sized databases using English-like sentences or commands. It is an extremely powerful database management package, but its power is provided at the expense of ease of use and friendliness. Unlike Cardbox, it does require the user to understand various technical aspects and terms used in computing, and it is more difficult to alter the format of records within the database once you have set it up and started to use it. The power and flexibility of dBASE II can be extended considerably with its built-in programming language. This allows complex applications to be constructed, and the result can be an easy-to-use and foolproof system suitable for even the most computer-native user.

records in the database, examining the occupation of each person, and ignoring those who are plumbers as well as those who bought within the last six months. The people left after this selection process are potential purchasers, so you can next ask the computer to print out their names and addresses. You can use another piece of software called a "word processor" (see Module 36) to then produce apparently personalized letters for each client automatically.

As another example of a database system in operation, imagine that you are a keen photographer and that you have a collection of thousands of transparencies accumulated over the years. Trying to find one image out of this many could take hours of valuable time and much effort. Now, with each picture carefully numbered and each with an entry in the database, if you want a picture of, say, Aunt Mary, all you need do is ask the database to tell you the numbers of all the transparencies that have that person on them. You can, though, be a little more specific and ask for the reference numbers of pictures that were taken between particular dates or in certain situations. The only limiting factor here is the amount of description you assign to each entry in the database.

Different types of database

By modern database standards, the examples above represent fairly simple applications that most database systems could handle with ease. Database systems for micros now range from quite basic to extremely sophisticated. The latter have such complex commands that using them to their full potential involves learning what almost amounts to a new programming language.

An example of the basic system is the now popular Cardbox package. It allows you to "draw" a "filing card" on screen, divide this up into boxes, give each box a caption, and then enter information, which is stored in a file on disk. Its virtues are ease of use and flexibility. If, for example, after you have set up your database you decide you do not like the way the information is displayed, you can easily alter the format or devise a new format that displays only part of each entry. Cardbox, however, cannot be called a true database, as there is no way to sort the contents of your database into, say, alphabetical order, and neither can it perform mathematical operations on any of the data.

At the other end of the range there are systems like dBase II, which allow you to do almost anything with your information. dBase II incorporates its own programming language with the facility for developing very sophisticated and entirely automatic systems which can be used by somebody with no computer or database training. It would be possible to hold all your business records in one dBase II database and access information for different requirements - invoicing, cashflow projections, general accounting, and mailing lists, for example.
Spreadsheets are fast becoming an accepted part of the business microcomputer world. Basically, a spreadsheet is a software package that facilitates the types of business calculations that would normally be carried out with paper, pencil, and a calculator. You have to imagine that your monitor screen is a sort of window, and through this window you can see a large sheet of paper. This sheet of paper could be of any length, and is divided up into a grid of rectangles. Each rectangle is known as a “cell”, and each cell can contain facts and figures about your business.

**Spreadsheets in practice**
To see where a spreadsheet might be of assistance to you, try to imagine that you run some sort of manufacturing business. Calculating things like sales targets is an integral part of this, and many other, types of company. You might also want to know how much your income from sales should grow each year to make sure that profits keep rising despite increases in the costs of raw materials, labor, and any other obvious overheads such as heating and lighting.

Your financial expert tells you that, as far as can be assumed, the costs of materials will increase at a rate of 5 per cent a year, labor probably by 8 per cent, and overhead by about 12 per cent. Using these figures and a calculator you could probably work out in a few hours that something in the region of a 4 per cent growth in sales is necessary to keep profits acceptable. But with the manual type of system described here, any change in any of the assumptions necessitates a complete recalculation.

In contrast to this, a computerized spreadsheet package allows you immeasurably more flexibility. Using the same type of situation described above, a spreadsheet display may look similar to the sheet of paper used for manual calculation. But there the similarity ends. Every time a factor affecting your business performance changes, all you need do is move the cursor down to a rectangle marked, say, “Labor increase”, type in the new percentage, and the computer immediately recalculates all the values in all the rectangles affected by the change.

You would then be able to see at a glance exactly what your financial position is likely to be at some future date using the most recent information at your disposal. It is also possible with a spreadsheet to experiment with the expected lowest and projected highest figures for all of the variable elements, and seeing how profits will be affected and watching the effect of pushing the growth in sales by varying amounts.

**How spreadsheets work**
This is just one example of a spreadsheet at work, and a deliberately simplified one. But it should be
enough to give you an idea of what these programs are designed for. They allow you to set up a matrix of cells and to define the relationships between all the elements. Changing the value in one cell will then alter automatically the values of all other cells that are mathematically linked to it. This degree of flexibility allows you to play "what if" speculation, with the results of your experimentation visible on the screen immediately.

Most spreadsheets now allow you to save the data on disk, print it out, link it with word processors, and even turn the numerical information into graphs, bar charts, or pie charts, where unusual trends or differences can often be more easily identified than when presented in the middle of a mass of figures.

Probably the best known and most successful of the spreadsheet programs is the original one – VisiCalc – designed for the Apple II. This one piece of software was responsible for a very considerable number of sales of the Apple II computer. People were prepared to buy the hardware in order solely to run this one piece of software. Observing the success of VisiCalc, the rest of the microcomputer software business was quick to realize that there was money to be made and dozens of spreadsheet packages soon appeared – most claiming to be more advanced than VisiCalc in some way, either in terms of ease of use, more features, or better mathematical capabilities. In the computer industry, these rival packages soon became known as Visi Clones.

VisiCalc, in addition to selling a lot of hardware, was also at least partly responsible for the infiltration of microcomputers into many big companies. Executives, frustrated by the inflexibility of their own data processing departments, suddenly realized that they could do what they needed to get done far more quickly and cheaply on their own desktops with a microcomputer than by going through official channels and waiting for internal departments to come to decisions on things like costing and feasibility.

**Spreadsheet applications**

Today, there is a spreadsheet available for just about every business computer on the market, and even some home microcomputers have cut-down spreadsheets available for them. The increasing use of color on both business and home machines adds a new dimension to spreadsheets (and other software packages) with, for example, negative amounts of money being highlighted in red.

It would be a mistake, though, to assume that spreadsheets are useful only for accounts work. In fact, they seem to be used for an astonishing variety of tasks – any that require the handling of figures which interact in complex ways.
Computers have long been used for handling and storing text but, until the micro boom, applications were limited. Using a big, expensive computer for writing sales letters, magazine articles, or books did not make any economic sense.

The impact of micros
Microcomputers have changed completely our way of regarding computers as text-handling machines. It is now perfectly reasonable to buy a computer to do nothing more than handle text and, in fact, there are many companies selling machines that are “dedicated” to this task and cannot be used for any other application. Now, every business microcomputer, and many home machines as well, can be used for word processing. Word processing has come to be regarded as a major application for small computers. What, then, does word processing involve, and what are the strengths and weaknesses of a microcomputer word processing package?

Word processing in practice
At its most basic, word processing involves little more than typing in text at a computer keyboard (see Module 20), then being able to store it on disk (see Module 22) or tape (see Module 21), retrieving it at a later date, and printing it out on paper via a printing unit (see Module 23).

If this was all that it was possible to do with a word processor, it would be uneconomical to buy one, unless you wanted to produce multiple copies of a document and did not, for some reason, want to have them printed using the traditional method.

But, in fact, word processors are a lot more versatile. For a start, once you have typed in your text, you can read it back on the screen and correct any typing mistakes you might have made. Because the text is held in the computer’s memory, alterations involve changing the contents of the appropriate memory locations only. Likewise, if you decide you do not like the phrasing of a sentence, you can change and polish it, and keep reworking it until you are satisfied. All this is done on screen without a single piece of paper being used.

Taking things a little further, you might decide that the order in which you have written the text is not the way you want it. Again, no problem. You simply have to position the cursor at the start of the offending paragraph, press a control key or code, move to the paragraph end and press another key to tell the computer that you want to do something with this entire block of text. If the text is to be deleted then the computer will remove it. If it is in the wrong place, move the cursor to the position it should be, type another code, and the entire block is moved to the new position (and the remaining text moved about as appropriate).

Suppose, for example, that you are writing a long article in which the words “Los Angeles” appear frequently. It can be time consuming to type these out each time, but you do not want to use abbreviations. All you need do is type “LA” while writing the article and then move back to the start of the text. Typing the code or key for the “search and replace” command will activate one of the most convenient features of word processing: the computer asks “Replace what?” and you type in “LA”; “With what?” asks the machine, and you reply “Los Angeles”. Press the RETURN key and immediately every occurrence of “LA” turns into “Los Angeles”.

You can use the same facility to find something buried in the text, saving you the time and effort of reading through it. Suppose part of the article concerned earthquakes, all you need do is ask the computer to look through the text for you and display the section in which earthquakes first appear.

Word processing refinements
The basic idea of word processing is that you write, check, and polish everything on the screen and only commit it to paper when you are satisfied. But the word processor’s abilities are not limited to the typing and editing stages, for you also have considerable control over the appearance of the text when it is eventually printed on paper.

For a start, just about every word processor will contain such basic features as justification of the right-hand margin, centering text between margins, automatic page numbering, and page headings and footnotes. Depending on the type of printer used, effects such as underlining, bold or italic typefaces, and proportional spacing can also be included.

The range of facilities available on some micro-
THE MICROWRITER

This text-handling device is designed to be used with one hand and is extremely portable. Characters are entered—appearing on a single-line display—by pressing the keys in various combinations. One key is provided for each finger, and there is a second thumb key that activates different control functions. The Microwriter can produce the entire ASCII character set. It also has its own built-in rechargeable batteries and enough RAM to store about 1500 words of text, which is retained even when the machine is turned off. Interfaces allow the connection of a recorder so that the text can be dumped to or loaded from tape, and there is an RS232 connection.

Do you need a word processor?

Word processors are fun to use but they can be expensive. Even if you own a computer already you will have to buy, in most cases, a word processing package to go with it. Unlike database and spreadsheet software packages, the uses and benefits of which are obvious and therefore easily assessed, word processing is a little more difficult to evaluate. Certain types of work, however, do lend themselves particularly well to word processing and, indeed, some are only practicable with a word processing package. Other types of text handling are more easily tackled with an ordinary typewriter.

If your business involves writing the occasional one-off letter or invoice, then there is no point in considering a word processor—it would be a waste of money. Word processing comes into its own when you have either a large volume of repetitive work or you want to produce large documents such as reports, articles, and books, which are likely to require extensive revisions before the final version is printed.

As an example of word processing at work, consider the type of situation discussed in the Database module (see Module 34)–a sales-based company sending out new product information by mail to customers. There already exists a database containing the name, address, telephone number, and other details of each client, some of which can be used to compile a mailing list. Assuming you have a suitable database system, you can produce a file on disk containing details of all the people to receive letters—name, address, and type of greeting. Using the word processor you write a standard letter, but insert special symbols (these will vary from one word processing package to the next) instead of the
name, address, and salutation. This information is also stored on disk, but in a separate file. Then, using the printing section of the word processor, you instruct it to print out the letter repeatedly, reading the names and addresses file as it does so and automatically inserting the name, address, and salutation in the right place on each copy of the letter. The result is a series of apparently individually typed letters, each one personalized.

Some word processors allow you to go a stage further. Suppose you have a range of several different products and have taken care, while compiling the database, to include details of which products each client is interested in. This information can be included in the file of names and addresses the database produces for the word processor. You can then write the framework of a standard letter and, in a separate file on disk, a number of paragraphs describing each product. The word processor can then print the framework letter each time, and read not only the names and addresses file but also the details of the products in which the client is interested. The processor then selects from the separate file the relevant paragraphs you have written describing your range of new products, and inserts these, too.

It is important to note, however, that not all word processing packages have this flexibility, and you will sometimes find difficulty in producing a suitable file from the database that can be read by the word processor. Generally, if you have an application like that described above, it is best to look at the “integrated” packages now available. These are collections of database, spreadsheet, and word processing programs, all designed to work together without compatibility problems.

Word processors are also gaining increasing acceptance with people who write extensive one-off documents. Technical and other report writers, for example, find that the ability to move blocks of text around and to correct typing mistakes makes their job much easier. These users often have complicated formatting requirements such as indented paragraphs, spaces left for illustrations or diagrams, and unusual layout problems. With a word processor, draft copies can even be printed and circulated to others involved for their comments and amendments, which can then be typed in for inclusion in the final draft.

**Hardware for word processors**

If word processing is going to be your main or only application for a computer, you need to take special care in choosing the right type. Assuming that your choice of computer is not dictated by the availability of some other software that you need, then the most important aspects of hardware selection from the word processing point of view are the screen and the keyboard.

Not surprisingly, word processing involves spending sometimes hours constantly referring to the screen, and so it is important that your machine has a clear, legible, and steady display with no flickering. One popular microcomputer comes with its own word processing package that works entirely in upper case characters. Apart from the fact that the word processing package itself is not convenient to work with, the constant appearance of upper case on the screen is not pleasant to read.

To begin with, the display must have legible upper and lower case letters with proper “descenders” (the tails on letters like “y” and “g” should actually go below the line – they are unattractively squashed up on some machines). You also need at least the “standard” display of 24 or 25 lines of 80 characters as anything smaller than this quickly becomes irritating.

Another important aspect of a screen is that it should have a non-reflective surface and that it should swivel and tilt to allow you to use it in the most comfortable position. And the display should have brightness and contrast controls within easy reach of the operator. The color of the display is another aspect you should consider: the most usual color combination is green characters on a black background, but this can also be in reverse. Other combinations include black characters on a white background (which can be too contrasty and lead to eye strain), or amber/orange characters on a brown background. More unusual combinations include amber on purple.

Like the screen, the keyboard is also something you should consider carefully before buying. It should be separate from the main computer so that you can position it comfortably, and the main part of the unit should be to standard typewriter pitch and layout. The “feel” of the computer keyboard is often different to that of a typewriter and the one that suits you is very much a matter of personal taste. There should, however, be at least four cursor keys (up, down, left, and right) and it helps to have other commonly used functions available at the press of a single, special dedicated key. This is especially convenient for things like deletions and insertions. Some machines have taken this too far, however, with up to 33 special keys, each of which has two functions (shifted and unshifted), making it a real test of patience and memory to find your way round. One processor available at the moment for the Japanese language has over 100 keys with 12 characters on every key.
Artificial intelligence (or AI) is probably one of the most sinister-sounding areas of computing as far as the non-expert is concerned. It conjures up visions of super-intelligent robots or massive computer complexes designed with the sole purpose of controlling and manipulating their human creators. All of this, of course, is in the best possible science fiction tradition.

The reality is considerably less sensational, although very interesting nevertheless. It is debatable whether it is even possible, let alone necessary, desirable, or useful, to attempt to reproduce the human brain in electronic form. And artificial intelligence as it is developing now, certainly does not seek to create an electronic equivalent of the human brain. Those areas of AI most intimately connected with this sort of work seem more interested in using computers to model certain aspects of the brain in an attempt to understand more about the way the brain works. Such models as have been constructed are very crude when compared to the real thing. It will be a long time before computer science develops to that degree of sophistication.

Currently, two major trends seem to be emerging from research into artificial intelligence. Of immediate practical application is the work being pioneered into “expert systems” and “knowledge engineering”, while the other major area is concerned with producing software, which, by acting in a more “intelligent” way, gives the user additional control over the machine and provides greater flexibility and versatility.

Expert systems

An expert system is one that can analyze a problem in the light of a “bank” of knowledge about a particular subject and, drawing on this knowledge, “decide” on the most likely solution. One area receiving much attention from the expert systems designers is the field of medical diagnostics. Several medical expert systems have been produced and experimented with, although these are not yet available commercially. The aim of the systems is not to replace the doctor with a computer, but to provide a doctor with assistance in areas that might be outside his or her particular area of expertise. Current medical expert systems concentrate on specialized areas that are unlikely to be familiar to the average family doctor. The program works with the doctor being questioned by the computer about a patient’s symptoms. Each answer is used by the machine to narrow down the search through its knowledge bank by a series of IF . . . THEN type of decisions until something is found that matches all or most of the conditions for a particular diagnosis. The computer might list a number of possible alternatives with a probability rating for each one.

An important feature of this kind of system is that it allows the user to interrogate the computer, asking it why it has arrived at its conclusions. The computer responds to these questions by showing how the information supplied to it meets certain “rules” within its knowledge bank and, therefore, gives the user the chance to follow the machine’s reasoning and to make sure that the information in the knowledge bank is accurate and up to date.

It is important to realize that a system like this is intended to aid the doctor. A doctor might be faced with a number of symptoms that point to something outside his or her area. Rather than immediately refer the patient to a specialist, hoping that the right specialist has been selected, the doctor can interrogate the expert system and obtain a guideline as to whether the suspicions are likely to be correct, reducing the risk of wasting the specialist’s time with a wrong diagnosis.

So far, these expert systems have achieved remarkable accuracy, often as good as that of a human doctor, provided the solution to a problem lies within the knowledge bank. But an important aspect of expert systems is that should information be missing from the knowledge bank, or should some of the information there subsequently be proved wrong, the base can be added to or altered. And considerable work is being done into the way the knowledge bank is built up in the first place. One aim is to allow an expert in a particular subject to “teach” a computer everything he or she knows, and for the computer then to structure this into an arrangement that can form the knowledge bank. In some areas, notably those concerning the diagnosis of certain plant diseases, expert systems that have been allowed to use their knowledge bank to develop their own rules of diagnosis, have had more
success than when following more traditional methods and practices.

**Geological expert system**

Another interesting and extremely successful expert system is one called Prospector, developed at the Artificial Intelligence Center of SRI International in California. The Prospector system is designed to imitate as far as possible the reasoning processes of a geologist in deciding whether or not a given region contains a particular type of mineral deposit. For each type of deposit of interest, a knowledge bank is built up describing in minute detail the regions already known to contain the deposit.

In a typical Prospector session, a user (generally a field worker) would describe the types of rocks found in a particular region and supply other geological information and observations in response to questions posed by the Prospector program. Prospector would then use the relevant knowledge bank for the mineral being searched for to decide the likelihood of that region containing the target mineral.

The aim of this expert system is to give the user the equivalent of direct access to a number of experts, each with in-depth knowledge of a particular class or mineral deposits. Other types of expert systems presently under development cover such diverse subjects as computer design and fault finding, structural engineering, mathematics, organic chemistry, and knowledge engineering. The possible areas for development are numerous.

One fact, though, emerges from a look at the current developments in artificial intelligence in general and expert systems in particular: the easiest applications are those for which there exists a definable group of rules that can be expressed in computing terms. Many human activities and occupations do not fall neatly into this type of category and efforts to computerize these are, therefore, less successful at the moment.

**Future trends**

The possibility for creating ultra-intelligent machines does already exist and this has posed some awkward philosophical and moral questions. It is not impossible that we shall soon be able to give a computer the ability to perform medical diagnosis with a far higher success rate than that achieved by a human doctor, for example. A computer, after all, is designed to be able to store vast amounts of information and recall it practically instantly and without error — something people cannot do. As the cost of computing power and memory devices fall, and as the power (in terms of ability) of the computer increases, it will become cheaper to produce these doctor-computers than it is to train new doctors. And if the doctor-computer can perform more accurately than a human and cost less to produce, should not a large part of this aspect of the medical profession be computerized? And if this were done, what would patient reaction be likely to be — would patients trust the machines, or would they rather deal with more fallible human doctors? Some initial research into these types of questions have thrown up some interesting results. Patients have indicated that they tend to prefer to discuss their medical problems with an anonymous machine and they are, in fact, likely to be more honest in their replies to “sensitive” questions, such as drinking and smoking habits, than they are with a human doctor. Other types of questions raised by this current development concern the fallibility of the programs. If, for example, the machine makes a wrong diagnosis and a patient dies as a result, would its creator — presumably a computer engineer, manufacturer, or programmer rather than a doctor — be either protected or liable in law just as a human doctor would be?

Looking even further ahead, it has been suggested that the falling price of computing hardware and increasing sophistication of computing software could result in extremely powerful computers becoming so cheap that we could be using them for all sorts of unusual applications — probably the most bizarre being the suggestion that android copies of dead relations could be created, which not only looked like the deceased, but also spoke and acted like them. On a less morbid note, there is the possibility of electronic pets: a small device, covered with fur, and which would be pretty “blank” when bought. It could then be trained to recognize its owner’s voice, to come when called, and would react to its owner’s moods, providing entertainment, sympathy, or just companionship as required.
Only a few years ago, the idea of anyone having a computer in the home was unimaginable. Computers at that time were simply too expensive and too big for any individual to consider buying — and what, after all, would anybody do with one even if they had the money and space? Today, though, several million people worldwide have access to a computer in the home, and their numbers are being swelled massively each year as more and more cheaper, smaller, and better computers appear. Home computers are no longer regarded as curiosities, but rather as another desirable and entertaining piece of home equipment, along with the hi-fi, the color television, and the video recorder.

The first incursion
The first major invasion of the computer into the home began with the video games machines (see Module 29). The very early models, by today's standards, were crude devices capable of playing only one rather elementary game; they plugged into a domestic television and produced a fairly boring black and white display. But despite this they were wildly successful. These models were soon replaced by machines offering more facilities: a choice of games and color displays, for example. Often, the range of different games was extended by plug-in "firmware" packs. These machines could not accurately be described as computers as they were dedicated to only one basic task — playing games. The games could be altered, but generally they were not programmable by the user to perform any other functions.

True home computers started off almost as crudely, offering a black and white display and a fairly limited version of the Basic programming language. Software companies were quick to spot the potential offered by this new development, and started to produce versions of popular video games for these machines. The advances in technology allowed hardware companies to make better machines, with more memory and extra facilities such as color and sound, and the games available took advantage of these, becoming correspondingly more and more sophisticated.

The two principal trends in computer technology continue: the hardware simultaneously becomes cheaper and more powerful. In a period not longer than about five years, today's sophisticated home computers will appear as crude as the video games of five years ago now look to us — probably even cruder. At the same time, computer technology is being adopted by manufacturers of a wide range of domestic electrical equipment. Microprocessors, for example, are now common in televisions, radios, hi-fis, video recorders, clocks, watches, telephones, cameras, and many other everyday items. Computer technology is pouring into the home from all directions, frequently without any outward sign other than the increased versatility gained by those devices incorporating microprocessors.

The not-so-ordinary television
As an example of what is in store, we can take a look at what has happened so far to the domestic television set and what is likely to happen to it in the future. Currently there are televisions offering, through microprocessor control, remote control over channel selection, color, brightness, volume, and so on. Internally, functions such as automatic frequency control and color balance are being taken over by chips, too, to provide a crisper, more accurate picture. In several countries, the broadcasting stations offer extra facilities superimposed on the television signal, which allow suitable televisions to receive textual information in addition to normal programs. Types of information typically available include travel, weather, news, recipes, share prices, and foreign exchange rates for currency. Also, a particularly ingenious enhancement provides deaf viewers with automatic subtitles for some programs.

At an even more sophisticated level, the television can be coupled to the telephone system through a special adaptor and then act as a computer terminal. Using a very simple keyboard arrangement, the user can access thousands of pages of information, which are displayed on the television screen. The British Prestel system pioneered this facility and it has since been developed and refined by many other countries, principally Japan. Prestel itself was originally conceived and implemented chiefly as a consumer service, but the high price of the special television sets and adaptors required, plus the additional costs of using it (tele-
The future · Module 38

phone charges, computer connect time, and the charges for some frames of information) have seriously inhibited its growth in the home. In the States, the three major videotex services are offered by Compuserve, The Source (owned by Reader's Digest), and The Dow Jones Information Service (a purely business data bank).

Because these types of system use the telephone cables, they offer the users an "interactive" service. A major envisaged domestic use of this system and similar ones is for "armchair shopping". A large department store can display descriptions and, eventually, photographs of its goods through the system, and these can be ordered simply by keying in a charge card number. The goods will then be delivered straight to your front door. Needless to say, safeguards must be built in to prevent fraud or even children ordering store-fulls of goods while their parents are not around. This type of service has proved very popular with travel agents, as it allows them to monitor seat availability on, say, a particular flight, and then to reserve seats if this is required by customers.

Despite its tremendous potential, though, these systems have suffered from the chicken-and-egg situation – few domestic users seem willing to pay for the service while it offers comparatively little of interest to them, and the information providers are reluctant to go to the expense and trouble of putting domestic information into the service while there are so few homes using it.

There are, though, even more advances to come for the television set. To begin with, the method currently used for broadcasting signals is restrictive and often provides unsatisfactory results. Television signals can only travel a limited distance and the provision of good-quality signals over a large geographical area requires an extensive network of television transmitters. Two ways of overcoming this are about to come on to the scene in a big way. The simplest and probably the cheapest, is to broadcast the signals from a geostationary satellite positioned hundreds of miles overhead. Special receivers on the ground could pick up these signals and feed them to your television set. For technical reasons, this could greatly increase the number of channels available, although it poses some problems in areas like Europe, where one country does not necessarily want to be bombarded with signals from its neighbor’s satellite.

One alternative is the cable television system. This relies on linking every home with some form of cabling – fiber-optic cables seem the most satisfactory – and providing digitized television signals that can be decoded and displayed on an ordinary television set. This system provides more channels, each with a far superior picture quality and with top-quality stereo sound. The cables are capable of carrying much more than television transmissions, though. Radio signals, also in digitized form can be provided and – of particular interest here – computer data can also be carried. This facility could also be interactive and, being independent of the telephone system, it would operate far more efficiently, cheaply, and quickly. The same cable could also replace the existing telephone cables and provide a much higher-quality service, both in terms of sound transmission and facilities, than current telephone systems allow.

**Further possibilities**

Most of the technology necessary for these advances is already available and, in some cases, already in service. What takes time is the installation of a complete and suitable cable network. With a cable network installed, the domestic users will have access not only to a potentially massive range of broadcasting and telephone services, they will also be able to link their television sets into a variety of computer-based services, offering, say, on-line access to electronic encyclopedias, full armchair shopping facilities, and a huge variety of information services from the simple (such as weather and travel) to the complex (such as scientific and business information). In addition, there will be extensive electronic mail possibilities: rather than write a letter on paper and mail it in the conventional way, many people will be able to type at a keyboard connected to their television and send letters for storage at a central computer, with the "address" (some sort of coding system will have to be evolved) of the recipient. The latter will switch on his or her television and press a special button to interrogate the central computer, asking it whether there is any
mail to be read. The mail would then be transferred through the cable system for display on the television and might even be printed out on a small printer attached to the set or transferred to a disk or tape in the home computer.

As you are probably beginning to realize, to be able to cope with all these extra services that will be provided, a considerable degree of computing power will have to be incorporated into the television set. Several television manufacturers have already started to look at the potential problems involved. Prototype sets are now being developed that incorporate their own video games computers and the next step in their development will be the provision of a complete home computer built into the television set, with a full keyboard communicating with the set through an infrared remote control link. Incorporating a computer in a television would allow a much better quality display to be provided than is found on current home computers. At the moment, home computers that you plug into a domestic television set have to convert their video signal into one that appears (to the television) as though it were coming from a broadcasting system. The television is then tuned into this signal just as though it were being tuned into a television station. This procedure limits severely the quality of the display and is one reason why the home computer equivalents of the popular arcade-type games never look as realistic.

The video revolution
The intelligent television will also form the centerpiece of another technological revolution, which is due to burst on to the domestic electronics scene within the next few years—the video disk.

Currently, video signals can be recorded on magnetic tape using analog technology with very high-quality results. Unfortunately, the equipment necessary to do this is massive, complex, and expensive, and is, therefore, confined to professional broadcasting companies and video producers. For domestic use, much less satisfactory techniques have to be used and the resulting pictures, despite considerable technological advances in recent years, still remain fairly bad in comparison to the professional machines. The video disk, however, allows prerecorded films and television programs to be displayed in the home with picture and sound qualities that far exceed those available with domestic video tape machines.

Laser technology
Several video disk technologies already exist, but the most promising of these seems to be one that

THE DIGITAL AUDIO DISK

For the computer industry, the digital audio disk offers an acceptably high information capacity of about two megabytes per disk—although not as high as the video disk with approximately two gigabytes. Currently, as it is purely an audio product, the DAD system offers no way of allowing a computer to interface with the player, but it seems very likely that this facility will be added to most players quite quickly. Although the rate of data transfer is fairly slow by computer disk standards, it is acceptable for all home and many business applications. Unlike conventional computer disks, DADs can be mass produced, enabling games and other home software to be distributed at very low cost indeed. And, of course, the medium is far more robust and reliable than present computer disks, which is another major incentive for the software industry to use them.

Disk player The digital audio disk player (right) is the latest example of the use of microcomputer technology in the home. Digitized sound is held on a special disk (see above) and is converted to an analog signal by the player’s electronics. Only about ten years ago, the equipment to do this would have filled an average bedroom. Today it is all handled by a few microchips within a unit small enough to sit on a bookshelf.
Laser player Both video and audio disk technologies use the same basic principle. The beam from a tiny laser is focused on the disk. As the disk rotates the minute “pits” and “bumps” representing the digitized sound cause variations in the reflected beam. A photodetector turns these variations into electrical pulses and microelectronics then take over, converting the streams of pulses into an audio signal. A constant data rate is maintained by varying the disk’s speed from about 500 rpm (when the beam is reading from near the disk’s center) to about 200 rpm (as the beam reaches the outside edge).

The future · Module 38

VIDEO DISK PLAYER

Video disks offer a superior picture quality to that available from domestic video tape recorders. The basic principle is the same as that used by the digital audio disk (see facing page). But because video signals contain far more information than audio signals a much larger disk is necessary to hold one hour of video. This, in turn, allows a video disk to store massive amounts of computer-readable information—either text, graphics, or a mixture of both.
uses a tiny laser beam to read digitized video signals from a 14 in diameter disk in a special player. The digitized signals are recorded as a pattern of marks on a reflective material. This is then covered in a transparent plastic coating to protect it from any sort of accidental damage. The laser beam focuses on the shiny surface (and in so doing ignores any small scratches on the protective outer coating, because these are out of focus) and the reflected beam, which flashes on and off according to whether the laser is shining on a mark or the shiny surface, is turned into a series of pulses by a photodetector. The digitized signal, which contains both video and stereo sound information, is converted into analog form and fed into the television set as though it were a television signal. The result is a television picture vastly superior to that provided by a tape-based machine, with the added bonus that the disk does not wear out as nothing actually comes into contact with it while it is being read. Analog video tapes, already unsatisfactory even when new, can very quickly wear to the point at which they become unusable, particularly if they are used frequently to “freeze” the signal — the picture from a video disk can be frozen indefinitely with no effect on the disk whatever, and without the “noise bar” that many tape machines produce.

In order to record one hour’s worth of digitized video signal, an immense storage capacity is needed. Most video disks provide the equivalent of two gigabytes (two million megabytes) of information and this point is the clue to the other big potential use of video disks — robust, cheaply mass-produced carriers of vast amounts of computer-readable information.

As an example of the potential of video disks, try to imagine you want to create an educational disk that will teach people about, say, wildlife. On one video disk you can record textual details of the various animals or plant life. You can also record maps showing the parts of the world they inhabit, as well as record digitized photographs and even film clips showing what the animals or plants look like or any other points of interest about them. In addition to all this, you can record a very large computer program. To use the disk, a special disk player would be required, and this would be connected to an intelligent television and, via the necessary interface, to a home computer, which uses the television as its display peripheral.

Suppose, for example, that you want to learn about elephants. You tell your computer this and it uses the computer-readable information on the disk to search out and find all the available material relating to elephants. Text describing the animal is displayed on the screen, accompanied by photographs and film clips. But you can go much further than this with your disk-based software. Having presented a certain amount of information, the computer could then proceed to question you about what you have seen. It could analyze your answers and then, depending on whether or not you have absorbed the information, will either progress to teaching you more facts about your chosen subject or re-present the unabsorbed items of information in the hope that you might learn them by seeing them again in a slightly different way.

The potential offered by video disks in acting as learning aids and information banks is tremendous when the computer and television set are linked to a disk player in this way. And the video disk will also allow us to indulge in some really fantastic games. As a foretaste of what is in store, consider an American experiment in which photographs were taken at five-feet intervals along every street in a small American town. These pictures, together with information about the layout of the streets were then put on to a video disk and, when linked up to a computer with the necessary software, players could “drive” through the town, turning down whatever streets took their fancy. Considerable effort is now being undertaken to develop some spectacular adventure-type games using similar techniques.

Video disks and players have been commercially available for some years but have, so far, not enjoyed a great commercial success, despite the considerably better product they produce. A major reason for their lack of popularity has been the fact that they are playback devices only, while tape machines allow people to record material from the television. Disks that can be recorded on only are thought to be unacceptable for domestic use, although they exist and are gaining favour as an archiving medium on large computer systems. Currently, work is taking place to develop disks that can be used just like tapes, and although some now exist in prototype form, they — and the resulting video disk boom — will not be generally available for a few years yet.

The compact disk
A good application of computer technology in the home — one that will actually turn full circle by benefiting the computer industry — is the digital audio disk, or compact disk.

Ever since Edison pioneered the phonograph, people have been trying to develop a way of recording sound and playing it back with a result as close as possible to that of the original. Through some very intensive research and development efforts, modern
hi-fi equipment has reached high standards of audio reproduction. In particular, amplifier and loudspeaker design has achieved a standard of sound reproduction the like of which Edison probably never dreamed possible, but until now the weakest link in the system has been the medium through which sound is distributed — the record itself.

The invention of the long playing record undoubtedly advanced the state of the art considerably and a modern, carefully produced LP in good condition can achieve remarkably accurate sound reproduction provided it is used with a turntable, pick-up arm, stylus, and cartridge of a very good standard. All these components in the system demand exceptionally high levels of engineering to produce top-quality sound reproduction, and even then the sound quality emerging from the system depends on the record itself.

Analog records — even the highest quality, direct-cut LPs — still use the same basic technology used by Edison. The sound is made to control a cutting head that produces a groove in the surface of the disk. This groove has "wiggles" in it, which correspond to the sound being recorded. On playback, a needle tracking along the groove is "wiggled" correspondingly, and this movement is translated into sound, originally by means of a massive horn to amplify the movement of a diaphragm connected to the needle, nowadays by electronics. But no matter how carefully the recording process is carried out, there are always imperfections that will mar the effect. In producing a modern LP, the original sound is usually recorded on tape, mixed with other sounds, and then re-recorded. A master disk is then cut, and from this several copies are made, which, in turn, are copied to produce the molds from which the records you buy are made. At each stage in the process, a slight reduction in quality is introduced into the recording. The finished LP will still give acceptable results if great care has been taken in its manufacture but only for a little while. Repeated playing wears the groove, dust and scratches on the surface cause noise, and soon the sound quality has noticeably deteriorated.

The digital audio disk (DAD) removes all of these problems at a single stroke. The original sound is digitized as close as is possible to its original source. It can then be mixed and re-recorded in digital form as often as is required without any reduction in the quality whatsoever. In fact, many analog LPs are now produced with this technique, and the sound is only converted back to analog signals when the master disk is cut.

The digital audio disk is completely different to the analog disk. For a start, it is much smaller — 4.7 in in diameter — and is silver colored. It is, in fact, a disk of aluminum sandwiched between two transparent, protective disks. Although it appears perfectly shiny, the aluminum actually consists of a very long spiral groove composed of tiny pits in the metal's surface. These pits are the digitized sound — a series of 1s and 0s — and they are read in a special player by a tiny laser beam.

The laser is focused on to the aluminum (and in doing so any small scratches are rendered out of focus). Each pit represents a "1" and momentarily cuts off the reflected laser beam. A shiny area, representing a "0", reflects the beam. The fluctuating, reflected laser beam shines into the photodetector, which converts the flashing light into electric pulses. These pulses are then decoded by electronic circuitry and converted into stereo analog signals, which can then be amplified and played back through loudspeakers.

Because the digital signal relies only on the binary 1s and 0s, represented by the presence or absence of an electrical signal, it is far less susceptible to distortion than a corresponding analog signal — any mixing or re-recording in the studios has no effect on the signal quality. The actual disk manufacture involves much higher quality control standards, however, and is generally carried out in a specially clean atmosphere. With a good-quality amplifier and loudspeaker system, the digital audio disk produces sound far superior to anything possible with analog technology. Because any small scratches or imperfections on the disk surface are ignored by the laser, and because some clever electronic tricks can compensate for even quite sizeable scratches, finger prints, and dust particles, the DAD is a far more robust distribution medium than the LP. Its smaller size also means that it is perfectly suitable for use in both portable and car players, too, and is easier to store.

The digital audio disk offers such a tremendous advance in home audio that without any doubt it will do to the LP exactly what the LP did to the old 78 — eliminate it. Within a matter of a few years, DAD players will have taken their place in every serious sound system, and shortly afterward will be universally accepted by the public.

Because DAD players will be so widely used, the system is of great interest to software producers as a way to distribute their products. One massive problem facing the software industry is theft; it is very easy to make a copy of a computer disk or tape. Only a few countries offer any legal protection against software theft and a medium that cannot easily be copied is something the industry has been looking for eagerly.
Although the spread of computer technology will have widespread effects in our homes (see Module 38), it is at work that the first and biggest impact will be felt. In many areas of the business world, of course, the process of computerization is already well underway.

Ever since the Industrial Revolution, advances in technology have been affecting how we work and, often, the types of work we perform, and there have been correspondingly large social changes as a result. The so-called micro-revolution – the increasing use of very low-cost computing power – will have an even greater effect, but it is important to realize that there is a fundamental difference between this and previous technological advances.

Previously, new technologies have enhanced man's muscle power, allowing huge loads to be moved with the minimum amount of effort, for example. Computers, on the other hand, extend man's mental powers; not, as is sometimes suggested, by taking away the need to think, but by performing complex, routine, repetitive actions quickly, accurately, and efficiently, or by making decisions and carrying out certain courses of action on the basis of massive amounts of information, which would either overwhelm or simply take too long to sift through.

Computers already at work

Computers are being put to work in the most unlikely places as well as the more obvious ones. Everywhere, small businesses are finding that they can not only afford to buy a computer, but that often they cannot afford to be without one. With the microcomputer industry barely in its adolescence, let alone maturity, the quality of both the hardware and software available is very patchy indeed. Also, the lack of general education about computers and their abilities (and limitations), combined with the sales “hard sell” endemic to the computer industry, means that it is not only easy for the computer-seeking business person to buy a totally unsuitable machine, but also for it to cause havoc when it is installed and working.

As you have probably gathered elsewhere in this book, the overwhelming trend is for microcomputers to become simultaneously cheaper and more powerful. At one time, desk-top machines with an 8-bit processor, 8 k of RAM, and a single cassette deck for storing and loading, were being sold as business machines and actually used for serious computing work. Today, even a home computer enthusiast would demand better facilities than this, especially in terms of memory capacity. In the office now, the traditional 8-bit, 64 k RAM, and twin-disk microcomputer is commonplace, with 16-bit machines, usually offering at least 128 k RAM as standard, just beginning to take over the upper end of the small business market. Soon, the 16-bit machines will be the standard, and we will be seeing 32-bit computers beginning to be introduced, offering massive computing power.

One of the major problems faced by the industry as a result of the hardware advances is that the software suppliers are having a hard job keeping up. When affordable 16-bit machines appeared there was virtually no software available for them other than hastily converted 8-bit products. Now, with 16-bit machines firmly established, there is still a lot of software to be provided, although much has been appearing in a remarkably short time.

With 32-bit micros just around the corner, the software industry is beginning to come under considerable pressure. To begin with, it has to be ready and able to provide software for the next generation of machines as soon as they appear – yet it is by no means certain which machine will be successful and what sort of facilities they will offer. So how do you go about designing software that will run on them? This situation is bound to result in an unfortunate lag between hardware and software availability.

At the other end of the industry is the user, rapidly becoming knowledgeable and powerful enough to start exerting pressure on both hardware and software manufacturers to provide the products he or she wants, rather than the ones the industry decides to provide. What the average business person wants at the moment is to be able to walk into a local office equipment shop with a problem and walk out again with a piece of equipment that will solve that problem. At the moment this is not how things work. Instead, it is discovered that the new piece of equipment, even if it will solve the problem, requires a lot of computer-orientated knowledge to be assimilated – from the business
point of view, not an ideal situation. Fortunately, the industry is beginning to realize this and is slowly taking steps to do something about it. In particular, the availability of more computing power, as promised by the 16-bit and 32-bit processors, plus the availability of these CPUs to address much larger areas of memory, combines with research into artificial intelligence (see Module 37) to provide much "friendlier" software. The use of graphics displays to allow users to select from a range of different applications is one of the ways in use to make things easier for non-experts.

The major trends in business computers themselves, then, are fairly obvious: they will become cheaper and more powerful, with increasing amounts of RAM being offered as standard. High-resolution, full-color displays will also become standard as these offer far superior ways of presenting information. And the software these machines will use will become progressively easier for the layperson to use until the off-the-shelf, plug-in-and-go software becomes a reality, requiring no more knowledge then buying a typewriter does today. Simultaneously, a massive revolution will be taking place in every area of work, even those for which computerization seems unlikely or impossible at the moment.

In the office
Office automation is one of the most visible areas of computerization, and it is also one of the oldest. Ever since commercial mainframes became available, computers have been used increasingly to automate work that previously was done manually. Probably the most widespread use of computers is in areas of business like payroll calculations, stock control, and general accountancy, where they have become firmly established.

Because computers are good at handling numbers, it has been the accountants who have usually been the first to benefit from computerization, with much of the tedious record keeping and bookkeeping work being removed from them. One big advantage of computerizing an accounting system is that at any given moment it is possible to obtain an accurate picture of your financial situation - how much you are owed, how much you owe, how much is in the bank at the moment, and what cashflow problems you might be faced with in the near future. It has been said that such is the rate of growth of some companies producing computer equipment, that they simply could not survive without the very extensive use of computers to keep their finances under control.

But the availability of low-cost computing power has affected other, less traditional areas, too, and will continue to do so on an increasingly wider scale. The most spectacular area of office automation has been in handling information. It started with word processing, in which the typist's skills are enhanced considerably through the use of computing power to input text, alter it, arrange its layout, and perform a large number of other operations such as the automatic generation of "personalized" letters using information held on each customer in a general-purpose database (see Module 34).

Every business requires a certain amount of information in order to function. With some businesses this is fairly simple - a list of customers' names and addresses, details of any financial transactions, and stock control details. With many other businesses, however, the information required at all levels to run the business successfully and efficiently is surprisingly large, and many people spend a lot of time simply gathering, collating, and assimilating it. Through the use of computers, much of this work can now be successfully automated and the people formerly involved in this information handling can devote more time to actually running the business.

Networks
The trend now is to provide your company with a central database of information, which can be accessed by everybody who needs it. Generally, the database is held on a mainframe or minicomputer and is accessed through terminals or microcomputers on people's desks. The growing use of computer networks allows information to be accessed easily and efficiently and it will not be very long before a microcomputer on every desk is the norm rather than the exception.

When computers were big and expensive, providing several people with simultaneous access to
COMPUTER NETWORK SYSTEMS

In the world of mainframes and minicomputers, the cost of the hardware makes it essential to allow many people to share the system simultaneously. This is achieved with the use of specially written software. With a microcomputer, however, the equipment is cheap enough for a company to be able to give a processor to everybody who needs one. But it may still be necessary for these machines to be connected together. One reason for doing this is to allow several users to access, say, a database held on a hard disk unit; other possibilities will give several people access to expensive items like a daisywheel printer or to send messages (known as "electronic mail") to each other.

A group of microcomputers can be

Xerox networks
The Xerox Corporation has started to market a highly sophisticated series of local area networks (see photograph far right) – the 8000 Series – featuring a device known as an Ethernet. Basically, the Ethernet is a coaxial cable to which attachments are made by "taps", which pierce the central core, and "transceivers", which screw into the tap (see bottom right). Ethernet not only allows computer data to be exchanged between machines, but it can also carry digitized speech and video information, too. Because of this, it is still rather expensive by micro standards and, in particular, the electronics needed to form the actual interface between computer and network are complex and costly. Shortly, though, this interface will have been reduced from several printed circuit boards to a handful of chips and the price will fall significantly. For shared devices like a printer (see right) a controlling unit is necessary to stop system users trying to access the printer at the same time.
connected together with a “local area network”. This is simply a term for a group of interconnected micros within the same room, or building, or company, as opposed to larger computer networks that use high-speed data communications lines and satellites to link computers, which can be hundreds or thousands of miles apart, and even in different countries.

Several network technologies already exist and, so far, with the industry still in its infancy, no single system has yet emerged as the industry standard. Most good network systems will have facilities for providing the types of features mentioned above. Some others may also be more sophisticated and have such things as interfaces to telephone modems (see Module 24) so that users can dial up other computers, and “gateways” or interfaces that allow users of one network access to other computer systems or local area networks.
the computer meant using elaborate software to divide up the time the CPU spent on each job. Jobs were dealt with in rotation, the CPU spending a short time on one, then switching to the next, and so on. With a large computer, the CPU is powerful enough to do this so quickly that each user thinks he or she has sole use of the machine, and so there is no bottleneck created.

Many attempts have been made to do the same thing with microcomputers, but with less successful results. On an 8-bit machine, the CPU has barely enough power to service two users at the same time. Even some of the 16-bit processors are hard pressed to provide a useful number of people with a reasonable service. In any case, the whole exercise is pointless on a microcomputer for both economic and technical reasons. If several people wish to compute, they must each have a screen, a keyboard, and an area of memory. These are all expensive components that must be provided. The CPU in a micro is the cheapest component and you might just as well give one of these to each user, too, instead of trying to share it out. You, therefore, end up giving each person a microcomputer. In an office, however, you often find that a number of people need access to the same information. It makes sense, then, to keep this information in one location - on hard disk, for example - and allow all users to access it. To do this, you need a network, a means of connecting up all the machines together and linking them to a central data store - and this type of information sharing is going to be one of the obvious future trends.

Several different network technologies already exist, of which the most interesting is the Ethernet system developed by the Xerox Corporation. Ethernet is still very expensive (by micro standards) to implement, as it allows the exchange of voice and video information as well as computer data. Other companies, though, have produced similar, cheaper systems (see Module 3), and Ethernet itself is becoming cheaper all the time.

Once a network has been established, all sorts of interesting possibilities become available. For a start, you can share other facilities as well as your common database: expensive daisywheel printers, too costly to be given to every user, can be slotted into the network and shared. Messages can be sent from one machine to another, allowing full electronic mail capabilities to be implemented. And extra processors can be incorporated to act as "gateways", either to other networks or to mini and mainframe computers storing very large databases. Network technology is a fast-growing area of microcomputing and many manufacturers are now announcing computers with built-in networking capabilities as standard.

**Communications**

Hand in hand with networks come advances in communications. You saw in Module 38 that plans are well advanced in many countries to join every home up to a cable network. This network will distribute not only digitized television, radio, and telephone signals, but will also be able to carry computer data, too. This will allow many more people to work as effectively from home, using their own microcomputer, as they could in the office. Communicating with colleagues would be by telephone and electronic mail, and you would actually appear in the office in person perhaps only once a week or less. Although this situation may still be some years in the future in many countries, it will have wide-reaching effects in those cities where it is accepted. For one thing, with more people staying at home to work, there will be much less need for office accommodation - perhaps only a small core of people will have to travel into the office and work there every day, so the demand for office space will shrink, and so too will transport requirements, with resulting savings in fuel consumption and corresponding reductions in pollution. On the other hand, the design of homes might well have to change, as people start to demand more space in which to work in peace, away from the distractions of the home.

The office worker of the future, whether based actually in an office or at home, will have immediate access to vast amounts of information, not only generated within his or her own company, but supplied by outside sources - "information utilities" - as well. Advanced software techniques, again drawing on current artificial intelligence research, will allow this information to be scanned, sorted, collated, abbreviated, and generally utilized very quickly, thus simplifying the task of keeping track of everything in a world where the amount of available information is increasing exponentially, as - almost - is the requirement for dealing with it all in a sensible way.

**Future factories**

Industry offers massive - and highly controversial - opportunities for the wide-spread introduction of computerized technologies. Already, industrial robots are finding their way into many plants, where they perform tirelessly and effectively, never stopping to rest or sleep, and needing less stringent heating and lighting requirements.

Already in Japan, there are several fully automated (albeit small) factories, where raw materials...
are delivered at one end and finished, packaged products collected at the other end. In between, highly sophisticated machinery, including robots, works under computer control, 24 hours a day, seven days a week, totally unsupervised by humans apart from occasional maintenance checks and repair work. In one such set-up, the owner spends most of his time with business connections and makes only one brief tour of inspection a day to make sure all is going according to plan. A complex monitoring system linked to the home warns instantly, day or night, if anything should go wrong. The entire production, from machine control to stock control, invoicing, and order handling is taken care of by computers. In the event of a major problem, the controlling computer will shut the plant down and wait for instructions.

Many people find the prospect of automation to this degree very disturbing. Trades unions are only one group among many that are particularly alarmed at seeing jobs taken over by computers in this way, although there is probably little if anything they can do to stop what has now become an inevitability. Widespread social reorganization and problems, including the entire revision of society's attitudes to those who are in work and those who are not, are only a few of the results of these trends. But it would be as wrong to pretend that facing these problems will not be difficult as it would be to hope that the situation will never arise.

Medical trends
As you saw in Module 37, extensive research is now taking place into the possibilities of providing doctors with expert systems to help them diagnose illnesses in areas outside their specialization. And these trends might eventually lead to the widespread use of computers to replace expensively-trained doctors.

An interesting development pioneered at the National Physical Laboratory in Britain was the medical interviewing computer, MICKIE. This was specifically designed to relieve the doctor of the time-consuming work of asking each patient the type of "where does it hurt?" questions. Before seeing the doctor, the patient would sit at a simplified keyboard and terminal and work through a series of questions designed to elicit this sort of basic information. The doctor would then be provided with a printed-out summary of the patient's symptoms, and he or she could then concentrate on these instead of having to spend valuable time persuading the patient to tell what the problem was. Several interesting points emerged from this. The doctors liked the system because it allowed them to use their time more efficiently. And patients liked it, too, as they felt they could take their time and because the computer seemed more interested in what they had to say than did most doctors. It was also discovered that patients would describe symptoms and discuss problems with the computer that they were too embarrassed to mention to a doctor, particularly if they were suffering from sensitive complaints, such as sexual disorders or those resulting from excessive alcohol intake. The computer consistently elicited more honest replies than did the human doctor, even though the patients knew that the doctor would see the results of their conversations with MICKIE.

Computers are beginning to play a major role in the administration of medical practices, too. Patient records can now be kept more efficiently on computer than on file cards, and computer records allow doctors to search for patients suffering from similar symptoms, for example, or for those taking similar drugs. This means that if a new treatment is discovered for a particular ailment, the doctor can immediately identify all the patients who are likely to benefit from the new treatment. Conversely, if an existing drug is found to have undesirable side effects, the doctor can quickly track down all the patients using that drug and notify them. The system can even discover such side effects by comparing patients' symptoms and medications. And, of course, a properly organized system can take care of the financial side, sending out patient billings automatically, a facility that is appealing to many busy doctors.

One worrying aspect of the use of computers in medical administration has been concern over the privacy of sensitive, personal information held on computers. In fact, elaborate systems of password protection can deny access to any unauthorized individual far more effectively than can a lock on a filing cabinet. Also, the doctor is bound by professional ethics to ensure that the confidentiality of patients' records is maintained - no matter how they are maintained - and the computer makes this task simpler and more secure.

Educational trends
Computers are already involved in education in two ways: it is considered necessary to give everybody at least a passing familiarity with computer technology as part of their general education; and computers can also be used in the classroom as extremely intelligent teaching machines.

Teaching students about computers is spreading rapidly in the Western world. Many countries now insist on a certain amount of computer science edu-
cation as a mandatory part of each school's curriculum, and it is not at all unusual to find this side of a child's education starting as young as six or eight; by the time they are about twelve, these children have become quite expert in matters dealing with computers and teachers report that this has beneficial side effects throughout the education system.

An entire computer language called Logo (see Module 32) has been developed to teach children not only how to use computers to solve problems, but to train their thought processes to cope with general problem solving. Logo takes advantage of all the advances provided by artificial intelligence research and psychology to bring about these aims. Its approach is to present itself in much broader terms so that the child develops a useful attitude to problem solving in general. The system provided by Logo is interactive, with the child being able to modify the programming through using it. Logo's principal teaching device is called a "turtle" - a small robot on wheels attached to the computer and under computer control. Using the keyboard, children can make the turtle execute highly complex movements and draw designs using a pen attached to an articulated arm.

Computers make excellent teachers of other types of subject, too. A lot of work is now taking place to develop highly interactive teaching software. With the advent of video disks (see Module 38), this will become easier, but it is still possible to provide some highly sophisticated learning packages using relatively simple computer hardware.

A particularly ambitious project has been developed by the Control Data Institute. Called Plato, it is a massive educational system covering a wide range of subjects and makes full use of on-screen graphics. Special terminals are required with touch-sensitive screens. These are used to overcome the need for "keyboard fluency", which many other systems require. Unfortunately, Plato, reputedly an excellent system, is only available currently on mainframe machines, although at some time in the future it is possible that micro-based versions may be produced, probably offering scaled-down versions of the full Plato system.

An unfortunate aspect of the use of computers in schools is that there is often a deep suspicion among the teaching staff toward the new technology. Because the technology is so new, few teachers have received any special training with or about computers and, therefore, feel unable to cope with them adequately. The current generation of teachers working in schools has not enjoyed the advantage of growing up with computers, even if only at the games-playing level, and frequently the children know more about the subject in general, and the computer in particular, than do the teachers. This, in itself, is more than some teachers can cope with. At other levels, too, teachers feel threatened by the incursion of computers into the classroom, fearing (perhaps with good reason) that the machine will prove better at teaching than they themselves.

Although it is possible to provide an educational computer, which, by and large, would perform better than most human teachers in terms of imparting information, the situation where it might be used to replace the human teacher is still a good few years away. When the technology is installed and working, teachers will probably find that - as in so many other occupations - the computer will take over a lot of the more repetitive, less-interesting aspects of the job, and, as a result, enrich rather than destroy the human role.

Experience, although it is only of a very limited nature so far, has shown that many children actually find learning from a computer more enjoyable than traditional learning techniques. Not only does the computer itself add an element of fun and novelty to the process but, with careful software techniques, it is possible to tailor each individual child's learning to suit the capabilities of that child. This flexibility is something that no teacher with a classroom full of students can ever hope to achieve. Each child could thus progress at the pace that suits him or her best, neither holding up the others nor being held up by them. At the same time, the teacher will be able to monitor the progress of the class, both as a whole, to ensure that overall aims and objectives are being achieved, and on an individual basis, identifying children who are having particular learning problems in certain areas and devoting time to helping those students while the others continue unhindered.
CHOOSING YOUR COMPUTER
HOW TO CHOOSE YOUR COMPUTER

When you first entertain the thought of buying a computer, deciding which one can seem an almost impossible task. With at least several hundred models to choose from, most people simply do not know where to start looking. This assumes that they even know what they are looking for in the first place. To make the decision-making process even harder, there are new machines being announced almost every week, with rumors constantly flying around about models on the verge of launch. How, then, do you make sense of all this?

Make the decision to buy
Rule number one is to ignore completely all those rumors about what may be coming out tomorrow, next week, or next month. It is in the very nature of the computer business—and in the microcomputer part of it in particular—that no matter what you buy today, there is always something more powerful about to be launched and at a cheaper price. Accept this at the start because it is inevitable; if you wait for another machine to appear, well, when it does, you will get to hear of something even more interesting coming along in a few months, and you will never get around to buying one.

Why do you want a computer?
The next step is to decide what sort of computer user you are going to be. Things are very much easier if you just want a home computer and you are not interested in any possible business uses. There are so many machines to choose from, it is really up to you to decide how much you are prepared to pay and what sort of facilities you require—memory size, color, sound, graphics, and so on. Do not forget that, although it may be very tempting to opt for the latest machine on the market, slightly more established microcomputers will have a greater amount of software available for them, and will probably also have had nearly all the hardware problems (or bugs) sorted out. Something, unfortunately, that cannot often be said about a machine new to the market.

Is your computer meant for business?
If you hope to use your home machine for some business uses as well, you will have to think a little deeper about the situation. What exactly do you want to do with the machine? If, for example, you want to use it for word processing (see Module 36) then, unless your word processing requirements are very small, a home computer will not really be suitable. Likewise, if you have large amounts of information to store and you need to be able to access it quickly and easily, again you could find using a home machine too limiting—especially if it is not equipped with fast-access facilities like disk drives (see Module 22).

While it may be tempting to save money by starting off with a home computer, and while home computer manufacturers frequently boast that their machines are suitable for business applications, you could well learn to regret your purchase. This is especially true if your livelihood depends on the business application to which you intend to put the computer.

Buying a computer strictly for business use involves some very careful planning and consideration. First, do not look at the brochures covering computer hardware at all—at this stage it is almost completely irrelevant. Initially, you must take a careful look at your business—how it is presently structured, which aspects of it are causing the problems, and how those problems could be solved. It may be that you can solve them quite easily with a little reorganization, and that a computer is not the answer after all—we have all been conditioned to the way of thinking that the computer is the answer to all our problems, but it most certainly is not. At present, there is only a rather narrow field of applications ideal for computerization.

What are computers good at doing?
The areas of your business that computers may be suitable for are those that largely involve a lot of repetitive work. Some examples of this type of job are producing and updating product listing and mailings to customers (see Module 34), the preparation and despatch of invoices, and the handling of general and routine accountancy work (see Module 35). Or in another type of business, you may have equipment that needs constant monitoring and survey results, for example—that need to be sorted out and collated. Another important area where a
Choosing your computer - Module 40

40

computer can be of assistance is in the field of financial modeling.

Having identified the possible areas that may benefit from computerization – perhaps with the help of an independent computer consultant, if necessary (never ask the advice of a computer salesperson at this stage as your objects may well be different) – you can then start to think about how to implement the new systems.

What software is necessary?
Assessing software is no trivial task (unless the application is fairly simple). But, like computer hardware, the volume of software on the market suitable for your type of intended machine can be pretty daunting. What you must do is visit as many specialist software stores as you can, see plenty of demonstrations, ask questions and – very important – try out the software for yourself.

If you are not the person who will ultimately be using the computer, you must let your employee or colleague try it out to see how he or she feels about it. While a good salesperson will work closely with you in assessing the suitability of a certain type of software, it is you who must make the final decision, so do not be persuaded until you have no more doubts left. Software selection may be more in perspective if you bear in mind that it may cost you up to half the price you pay for the entire hardware set-up.

Relating hardware or software
Once you are sure you have found the software you feel is suitable for your particular application, you will probably find that your choice of hardware has been narrowed down very considerably. Your choice might even be as narrow as one machine, especially if your application is so out of the ordinary that only one piece of software exists. This could be either a good or bad situation, depending on your point of view. Next it is a matter of more practical considerations: has the machine of your choice sufficient disk capacity (see Module 22)? Are the display and keyboard facilities (see Module 20) sufficiently good quality? Again, if you are not the one who is going to be using the machine, you are not the one to make the decision. Staring for long periods of time at a poor-quality display can be exhausting. Also, the keyboard should, ideally, be separate from the main console so that it can be positioned comfortably for the operator. Screens, too, should tilt and swivel so that they can be viewed from the best angle. The color of the display should also be considered. And the best color may well depend on the particular type of lighting it is going to be viewed under (see Module 20).

Another important consideration when buying is after-sales service. If your computer malfunctions while producing your monthly invoices, for example, you might find your cashflow adversely affected unless you can get the computer working again very quickly. When you start looking at computer systems you must decide whether you can really afford it – the initial purchase and the running costs. And what exactly is included in the price? Does it include operator training, or spare disks, or the initial stationery for the printer (see Module 23)? If you are buying a home computer system, you will be surprised at the extra percentage cost these little items can amount to. So investigate these details fully before making a final decision.

In the following pages of this section, you will find details of some – but by no means all – of the computers available, for both home and business use. Some are produced by the giants of the microcomputer industry like Apple or IBM. These have, therefore, been given extra coverage, including some background information about the companies involved. Also included in this section are important details on computers from some of the smaller manufacturers. Again, because of the vast number of computers available, only a limited number can be treated. A proper survey of all computers that are worth considering is the subject of an entire book in itself, and space is simply not available here, unfortunately, to do full justice to the market. It should not be taken as a criticism if a particular model of type of computer has not been included.

Microcomputer prices
The prices of microcomputers are very subject to fluctuation, most often downward. The prices quoted in this book are the most up to date available, but you should bear in mind that since there is fierce competition at all levels of the market price cutting occurs.

All prices given are approximate and represent in most cases the recommended retail price of each machine’s basic configuration. This means the machine alone, without any of the items listed as optional in the technical specifications. It is particularly important to remember that the disk drive, which augments the machine’s memory, the printer, which provides a print-out of the computer’s operations, and printer paper, all add to the machine’s cost, substantially in the case of the first two items. However, while the printer is a separate element in the computer system, except in some portable models, the disk drive is built into most business machines.

163
In 1975, in California, Steve Jobs and Steve Wozniak, two young computer enthusiasts excited by the potential of the newly introduced microprocessor, used an inexpensive 6502 chip as the basis for the machine they named the Apple I.

Having impressed the local computer club and encouraged by selling some 50 machines, they set to work on an upgraded model with a proper case and a keyboard. The Apple II met with immediate acclaim and was largely responsible for the boom in microcomputers. Very soon it was to be found in offices, laboratories, schools, and homes across the USA and in many other countries. This remarkable success also generated a whole sub-industry producing Apple-compatible software and hardware.

Although the machine is outdated now, it still sells well because of its vast range of software. The Apple II Plus was an updated version and was itself followed, after the introduction of the Apple III, by the IIe, the latest in the II series and offering considerable improvements over the original model.

The intervening Apple III had none of the success of its predecessors and in fact proved to be a disappointment. The price was high, disk capacity was limited, and unless you used a program allowing it to run Apple II software there was little software available for it. There were technical problems too, such as faulty sockets which allowed chips to work loose. By the time Apple had resolved these difficulties, it needed a new machine to boost its image.

This machine is the Lisa. It displays its functions pictorially on the screen and the user selects the one required by using a "mouse". The mouse is a small box on rollers which is moved over the adjacent desk top, causing a pointer on the screen to select the desired function. To activate that operation a button on the mouse is pressed.

**Apple II Plus**

The Apple II Plus is an improved version of Apple's most successful machine so far, the Apple II. Its impact was largely due to its innovative design and versatility, being able to handle small business, home, and educational applications. There are more hardware add-ons and a greater selection of software packages available for the Apple II than for any other micro, ensuring that you can adapt your machine to undertake the vast majority of microcomputing tasks.

However, the Apple II does have certain disadvantages. It is now technically rather outdated and consequently slower in operation than many more modern machines. The keyboard is too clumsy to be used comfortably for word processing on a large scale, and the screen display can be difficult to read. Adding a Z80 card to allow you to run CP/M programs, and an 80-column display card can alleviate some of these problems. But many newer micros now offer better value for money.

### TECHNICAL SPECIFICATION

**HARDWARE**

- CPU: 6502
- RAM: 48 k, expandable to 64 k
- Display: 24 lines x 40 characters, upper case only; 15 colors in low-resolution mode (40 x 48), six colors in high-resolution mode (140 x 192), or monochrome high-resolution (280 x 192)
- Keyboard: 46 keys
- Disks: Optional; up to eight 5¼ in disk drives, capacity 143 k, integral cassette interface supplied as standard
- I/O: Built-in games paddle socket, eight internal slots for plug-in expansion cards, extensive range of plug-in cards available as optional extras

### SOFTWARE

- Operating system: Apple's own disk operating system (with optional disk drives, UCSD p-system (optional), CP/M 80 (with optional plug-in Z80 card))
- Languages: Integer Basic, AppleSoft Basic; optional Fortran, Pascal, Cobol (with Z80 card), assembler
- Applications: Large range available for almost all applications

### USE

- Business, education, science, home

### PRICE

- Approx. £700
APPLE IIe

This latest version of the Apple II brings this popular machine up-to-date by giving it the sort of features that have become standard on other 8-bit machines at the low end of the business market. These include a full 64 k of RAM, a good keyboard with proper cursor control keys, and a neater way of plugging in accessories.

Internally, the changes have been extensive. Advancements in chip design allow the IIe to use only a quarter of the number of memory chips that went into its predecessors. The machine is more reliable, uses less power and generates less heat.

Old Apple II software will run on the new machine but many software houses are revising their products.

APPLE III

The Apple III was designed to provide the business user with a more powerful machine than the Apple II. It has a sophisticated operating system and many useful features as standard. The display and keyboard are a considerable improvement on the Apple II, and its relatively large RAM makes it possible to run quite elaborate programs.

However, the Apple III has been beset with problems. Shortly after its launch it had to be withdrawn because of technical defects. This gave a head start to many of Apple's competitors who were able to produce better and cheaper machines before its relaunch a year later.

The Apple III has never recovered from these early setbacks. Sales have remained relatively low and as a consequence there has been less software specially designed for the Apple III than for the highly successful Apple II. While you can use Apple II software, this is unlikely to exploit fully the additional (and expensive) features of the Apple III.

TECHNICAL SPECIFICATION

HARDWARE
CPU 6502A with additional Apple circuitry
RAM 128 k, expandable to 256 k
Display 24 lines x 80 characters (monochrome), 24 x 40 (16 colors); 192 x 280 or 192 x 560 monochrome graphics, 192 x 280 limited 16-color graphics, 192 x 240 unlimited 16-color graphics
Keyboard 74 auto-repeating keys, including numeric pad and cursor control
Disks Single 5 1/4 in built-in drive, capacity 143 k (up to 3 extra drives can be added externally); optional hard disk drive, 5 megabytes (up to four can be added)
I/O Serial printer and Apple Silentype thermal printer ports; games joysticks port standard; four internal expansion slots; optional Centronics parallel port

SOFTWARE
Operating system SOS (Apple's own operating system); Apple II disk operating system emulator
Languages Business Basic, Pascal, Cobol, assembler
Applications Spreadsheet, word processing, mailing list, business graphics

USE
Business and education

PRICE
Approx. £2400
Apple designed the Lisa to provide an "executive workstation" that is more "friendly" and easier for the non-expert to use than business systems have been up until now. One of its many hardware innovations is an input device known as a mouse. This is a small box connected to the computer that controls a cursor on the screen as you move it about the desk and allows you to select a pictorially represented menu option. You need only use the keyboard for specific tasks such as word processing.

Specially-written software (the Lisa is not software-compatible with the II and III) using the machine's extensive graphics capabilities, allows you to carry out all standard business applications, including spreadsheet, list management, and word processing.

The Lisa is a very powerful machine. It uses a 16-bit processor and has one megabyte of RAM as standard (far more than any other personal computer).

In spite of its attractions, the Lisa is not an automatic
best seller. It is very expensive and does not have color, a feature that most people expect from a machine in this price bracket. Most buyers will probably settle for a machine that handles the same range of applications in a more conventional way at half the price.

**TECHNICAL SPECIFICATION**

**HARDWARE**
- CPU 68000
- RAM 1 megabyte
- ROM 16 k

**Display**
Up to 40 lines x 132 characters, 720 x 364 graphics

**Keyboard**
77 keys including numeric pad and cursor control

**Disk storage**
The Lisa has twin 5½ in floppy disk drives built in. These allow you to store up to 1.7 megabytes of information. In addition, you can link up an Apple hard disk drive for an extra 5 megabytes of storage.

**Disks**
Integral twin 5½ in floppy, capacity 860 k per disk; optional 5 megabyte hard disk

**I/O**
Twin RS232 serial ports, single parallel port

**SOFTWARE**

**Operating system**
Apple's own (supports CP/M and Xenix)

**Languages**
Optional Basic, Pascal, Cobol, assembler

**Applications**
Word processing, list processing, spreadsheet, project management, graphics handling

**USE**
Business

**PRICE**
Approx. £6500

**APPLE ADD-ONS**

**Like all major computer manufacturers, Apple produces a wide range of peripheral equipment to enable users to expand and upgrade their systems to allow greater efficiency of data handling and more complex applications. The products described below are examples of hardware add-ons that can be linked to Apple micros to provide extra storage and hard copy.**

- **Unifile**
This is a double-density 5½ in single disk drive (there is also a dual version) recently produced for the Apple III. This provides 850 k of storage and is designed to provide back-up files for information on hard disk.

- **Profile**
Profile is a 5 megabyte hard disk drive for use with the Apple III and the Lisa. It uses a 5½ in winchester disk. Up to four of these drives can be linked to one Apple III.

- **Dot matrix printer**
Apple produces several different types of printer for use with its computers. This dot matrix printer can be linked up to any Apple machine.
The world's largest computer and office-equipment manufacturer, IBM, began its corporate life in 1911 as the Computing-Tabulating-Recording Co. This corporation brought together companies which had variously performed statistical work for the US census office, produced computing scales, and manufactured mechanical time recorders. In 1924 the name International Business Machines Corporation was adopted.

By the 1930s the company was producing machines that could go beyond addition and subtraction to perform full-scale accounting tasks. IBM continued to expand during the Depression and in 1936 was able to provide the machines and services for "the biggest accounting operation of all time"—the Social Security Program. World War II brought further involvement with government programs. Washington used more than 5000 of IBM's accounting machines to keep track of troops and equipment, while other IBM machines were used in mobile units overseas.

In 1944, after six years' cooperation with Harvard University, the first general-purpose digital computer was completed. Four years later came the Selective Sequence Electronic Calculator, and in 1953, IBM's first production computer, the 701. The development of electronics and data processing technology has produced continual improvement, particularly in computers' rate of operation. While the 701 performed 21,000 calculations per second, IBM's most powerful computers now handle millions of instructions per second.

It was inevitable that IBM, like many of the other mainframe manufacturers, should eventually produce a microcomputer. What was surprising, however, was the skill with which IBM made its debut in this field. The IBM Personal Computer, launched in late 1981, enjoyed enormous success. It was a true micro, whereas many other manufacturers of large computers only produced adapted versions of minicomputers running operating systems and software that were not standard in the micro world. Like the pioneering microcomputers of the 1970s, the Personal Computer spawned, with IBM's full approval, a huge sub-industry of software and hardware production companies.
The IBM Personal Computer is an advanced 16-bit microcomputer which, since its launch in 1981, has become the industry standard for sophisticated business micros. Large quantities of software have been created for the IBM, making it a very versatile machine, able to handle all usual applications.

The basic IBM is made up of four units. It has a good typewriter-style separate keyboard with a numeric keypad and programmable function keys. A "system unit" contains a high-speed 16-bit microprocessor and 64 k of RAM. It also houses a speaker and up to two 5¼ in disk drives. The monitor has a monochrome display which is clear and well laid-out. Color, however, is not standard, but can be added with a color monitor.

**TECHNICAL SPECIFICATION**

**HARDWARE**
- CPU 8088
- RAM 64 k, expandable to 512 k
- ROM 40 k, including automatic fault diagnosis that operates on power-up
- Display 25 lines x 80 characters; optional color/graphics display card (color monitor required) to give 16 foreground and 8 background colors
- Keyboard 83 keys including numeric pad, cursor control and 10 programmable keys
- Disks Single 5¼ in, capacity 160 k; optional second 160 k disk, optional single or twin 320 k disks
- I/O RS232 port; additional RS232 port optional

**SOFTWARE**
- Operating system MS-DOS, UCSD p-System
- Languages Basic; other languages optional
- Applications Large range available, including word processing, database, spreadsheet and general business

**USE**
- Business, particularly large companies

**PRICE**
- Approx. £2700

**IBM PCjr**

New to IBM is the PCjr, known unofficially as the "Peanut". This is a 64 k home machine available in either a basic (cassette-based) or enhanced (single floppy) version. It has slots for cartridges, a good keyboard, and uses an external monitor or TV. The keyboard is, in fact, cordless, using infrared signals to communicate with the processor.
The microcomputer industry has experienced several revolutions. One of them was the advent in 1980 of the Sinclair ZX80, the first proper computer to be offered at below the significant figure of £100. It represented the debut in the field of micros of Sinclair Research, a company founded in July 1979 by Clive Sinclair, the British entrepreneur and consumer electronics expert.

The tiny ZX80, measuring only 9 ins × 7 ins × 2 ins and weighing a mere 12 oz, was greeted with immense enthusiasm. Over 100,000 machines were sold, with more than 60 per cent of these being exported. The computer was provided with Basic in ROM and all that was required to set it up was to plug it into a domestic television set.

The keyboard was of the membrane type, with the characters and numerals marked on it, but it was features like this that cut production costs and allowed Sinclair to sell the machine so inexpensively. In any case, a whole new market of home users was prepared to overlook such inconveniences in its enthusiasm for low-cost computing.

By March 1981, when the more advanced successor to the ZX80 was launched, Clive Sinclair's prediction that the earlier machine would dramatically change the face of personal computing had been proved right. The new computer, known as the ZX81 in the UK and as the TS1000 in the USA, was similarly available at a budget price and by the end of August 1982 had sold over 500,000 units in more than 30 countries. Sales have continued healthily, so that Sinclair has become, in terms of production volume, the world's foremost personal computer company.

The ZX81 once again advanced home computing. The modest 2 k RAM can be expanded to 16 k with the Sinclair pack and to a very versatile 64 k with modules from independent manufacturers. Another breakthrough was the introduction of a low-cost thermal printer for use with the ZX81. Software was never a problem with this machine: independent suppliers had acknowledged the importance of the ZX80 and were well prepared to provide software for the new model.

Sinclair's third and most recent computer, known as the ZX Spectrum in the UK, was launched in April 1982 to sell concurrently with the ZX81. Although it is an improved version of the earlier machine, it was not intended to replace it. It has Basic in 16 k of ROM and 16 k of RAM, expandable to 48 k. (A 48 k model is also available.) Movable keys, and color, graphics, and sound capabilities are further refinements. The versatility of the Spectrum makes it suitable for home, business, and educational use. UK sales are well established and the machine reached the USA early in 1983.

Sinclair Research is not the manufacturer of any of its machines. All production is sub-contracted, which has left the company free to concentrate on the conception, development, and marketing of products. It invests the majority of its profits in research and development, but personal computers are not the sole beneficiaries of these funds: a screen pocket television is soon to be launched.

In February 1982 Sinclair Research reached a licensing agreement with Timex Corporation whereby the latter manufactures and markets Sinclair personal computers in North America. The contract, covering present and future personal computer product development by Sinclair as well as Timex's own developments of Sinclair's technology, gives Sinclair Research a royalty on all sales.
ZX81

The ZX81 is probably the best-selling computer in the world. Although other companies are producing home computers at ever-lower prices, the ZX81 still remains the cheapest machine on the market.

The ZX81 has a touch-sensitive keyboard which, although adequate for simple home computing, is not suitable for serious use. There is 2 k of RAM, which can be expanded up to 64 k. The 8 k ROM contains a Basic interpreter. Storage is limited to cassettes. No monitor is provided, as at this level it is assumed the user will connect up to a domestic television set. Possible hardware add-ons include a Sinclair printer, a typewriter-style keyboard and various sophisticated interfaces.

There has been a lot of software produced for the ZX81, consisting mainly of games of widely varying quality, but there are also some simple business packages.

**TECHNICAL SPECIFICATION**

<table>
<thead>
<tr>
<th>HARDWARE</th>
<th>SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong> Z80A</td>
<td><strong>Operating system</strong> Sinclair’s own in ROM</td>
</tr>
<tr>
<td><strong>RAM</strong> 2 k, expandable to 16 k with Sinclair pack or 64 k with independently produced modules</td>
<td><strong>Languages</strong> Sinclair Basic in ROM</td>
</tr>
<tr>
<td><strong>ROM</strong> 8 k containing Basic interpreter</td>
<td><strong>Applications</strong> Large quantity of software, mostly games of varying quality and complexity; some products intended for business use</td>
</tr>
<tr>
<td><strong>Display</strong> 24 lines × 32 characters; 64 × 44 graphics</td>
<td><strong>USE</strong></td>
</tr>
<tr>
<td><strong>Keyboard</strong> 40 touch-sensitive keys; single-key entry of Basic keywords</td>
<td><strong>Home</strong></td>
</tr>
<tr>
<td><strong>Cassette Interface</strong> to domestic cassette recorder</td>
<td><strong>PRICE</strong> Approx. £40</td>
</tr>
<tr>
<td><strong>I/O Connector</strong> for add-on interfaces</td>
<td></td>
</tr>
</tbody>
</table>

ZX Spectrum

The ZX Spectrum is a more sophisticated machine than the ZX81, although it is still very cheap.

It has a typewriter-style keyboard which, although easier to use than that of the ZX81, employs an often confusing single-key entry of Basic keywords through the complex use of the shift keys. The standard Spectrum has 16 k of RAM, but you can obtain a 48 k version. The 16 k ROM contains the operating system and an enhanced version of Sinclair Basic, which allows you to run ZX81 programs without alteration in most cases. This Basic allows full floating point arithmetic, with string handling, numerical and string arrays. The Spectrum also offers sound, color, and graphics. A Sinclair printer can be added. Memory storage is currently mainly by domestic cassette recorder. However, Sinclair have recently launched their own Microdrive that will accommodate 85 k of storage.

**TECHNICAL SPECIFICATION**

<table>
<thead>
<tr>
<th>HARDWARE</th>
<th>SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong> Z80A</td>
<td><strong>Operating system</strong> Sinclair’s own in ROM</td>
</tr>
<tr>
<td><strong>RAM</strong> 16 k (expandable to 48 k); 48 k model also available</td>
<td><strong>Languages</strong> Enhanced Sinclair Basic in ROM</td>
</tr>
<tr>
<td><strong>ROM</strong> 16 k including operating system and Basic</td>
<td><strong>Applications</strong> Wide choice of games and some business programs</td>
</tr>
<tr>
<td><strong>Display</strong> 24 lines × 32 characters (user-reconfigurable to 40-column Teletext standard), 8 colors for foreground, background, and borders; 256 × 176 graphics, 16 graphics characters, 21 user-defined graphics characters</td>
<td><strong>USE</strong></td>
</tr>
<tr>
<td><strong>Keyboard</strong> 40 keys, auto-repeat on all, single-key entry of Basic keywords and graphics commands</td>
<td><strong>Home</strong></td>
</tr>
<tr>
<td><strong>Cassette Interface</strong> to standard cassette recorder</td>
<td><strong>PRICE</strong> Approx. £100 (16 k)</td>
</tr>
<tr>
<td><strong>I/O Connector</strong> for add-on interfaces</td>
<td>Approx. £130 (48 k)</td>
</tr>
</tbody>
</table>
With Apple and Tandy Corporation, Commodore Business Machines was one of the big three companies which founded the microcomputer industry. Commodore was originally a producer of hand-held calculators and was the first company to mass-market an electronic calculator for less than $100.

Commodore prospered in the 1970s, taking over the semiconductor research and manufacturing company MOS Technology in 1976. This move proved very fruitful since it produced the 6502 microprocessor on which the best-selling Commodore PET was to be based. The first PET was shown to the public in June 1977 and received rapturous acclaim. Commodore established a dealer network in early 1978.

The PET was the first fully-integrated micro, with the screen, keyboard, and cassette unit enclosed in the same case. It developed into a series of business and home machines, complete with peripherals, which established Commodore as a market leader. Over the years the VIC-20 has achieved huge sales worldwide, to become one of the most popular of all computers, but is increasingly sharing the honors with the Commodore 64, which has substantially more memory.

The 8000 Series computers are designed for both business and educational use, and the 4000 Series machines for the latter only, while the P Series is intended for scientific and business use and the B Series computers specifically for business.

Commodore PET
The rather unwieldy name Personal Electronic Transactor soon gave way to the neat acronym PET. Originally intended for home use, this innovative machine, for which a huge range of software was produced, was rapidly adopted by the business community. The PET which is shown right probably dates from 1978.

VIC-20
The VIC-20 has been a best-selling micro for some years; Commodore claim worldwide sales of over one million units. Its appeal is mainly to first-time computer owners and games enthusiasts, as the small RAM size of 5 k is too limiting for any serious applications.

The VIC-20 is software-compatible with the now obsolete PET range, and users can therefore draw on the large pool of software that was created for those early micros and adapt the programs to take advantage of VIC’s improved capabilities, such as color and high-resolution graphics.

The VIC-20 consists of a single keyboard unit containing the CPU and memory chips. The operating system and a Basic interpreter are contained in ROM. Input and output facilities include a connection for a domestic television or monitor, plug-in cartridge slot, modem socket, and a games paddle or joystick port. Most users store their programs on cassette, but you can also link up to a 5½ in disk drive.

One of the main attractions of the VIC-20 is that it is widely available in computer shops for direct sale to the public. This means that you can try it out before buying.

**TECHNICAL SPECIFICATION**

**HARDWARE**
- CPU 6502
- RAM 5 k, expandable to 29 k
- ROM Operating system and Basic interpreter in ROM
- Display 25 lines x 22 characters with block graphics, 8 border colors, 16 screen colors, 8 character colors; optional high-resolution graphics cartridge gives 176 x 158 graphics
- Keyboard 66 keys including cursor control and four programmable keys
- I/O Cassette, joystick and modem interfaces; VHF television
- Expansion connector for external RAM and plug-in cartridges; printer and disk drive port

**SOFTWARE**
- Operating system: Commodore’s own in ROM
- Language Basic in ROM
- Applications: Wide range of games and general home software; some “lightweight” business packages

**USE**
- Home

**PRICE**
- Approx. £140
Choosing your computer · Module 44

44

COMMODORE 64

The Commodore 64 is one of the most interesting personal computers currently available and is likely to become very popular among home, education, and business users.

Externally it looks very similar to the VIC-20, but it is much more powerful. It offers 64 k of RAM as standard (but remember, if you are using the Basic interpreter only 38 k of this is actually available). It has a much better display and improved graphics, but these are still limited by the capabilities of the domestic television set.

Commodore has produced a large range of add-ons for the VIC-20, including printers, disk drives, games controllers and sophisticated interfaces. The 64 can use all of these and is also capable of some degree of networking. The Commodore is software-compatible with the VIC and the PET and is also likely to be supplied with a range of specially written software. In addition, you can add a Z80 processor in the form of a plug-in cartridge to give you access to the vast store of CP/M programs.

Commodore has recently launched a portable version of the 64, the Executive 64.

4000 SERIES

The Commodore 4000 series computers are the direct descendants of the original PET. These machines are mainly suitable for the education market, as more modern micros with improved capabilities have superseded the 4000 in the home entertainment and business markets. However, the 4000 has been a very popular micro and many are still in use in all areas of computing.

The 4000 series have several points in their favor. They have direct access to the large pool of software of all types that has been developed for the PET since 1978. They are simple to use and of sturdy construction (an important consideration for educational buyers). Commodore 4000 systems are also fairly easily expanded, allowing you to add on disk drives and printers, and other peripherals. There is also extensive documentation.

There are two different versions of the 4000 series, the 4016 with 16 k of RAM and the 4032 with 32 k of RAM. The standard configuration of both versions comprises a monochrome monitor mounted on a type-writer-style keyboard unit.

---

TECHNICAL SPECIFICATION

HARDWARE
CPU 6510 (enhanced version of the 6502)
RAM 64 k
ROM 20 k containing operating system and Basic interpreter
Display 25 lines x 40 characters, 320 x 200 graphics, 62 predefined graphics symbols, 16 text colors, 255 screen and border combinations
Keyboard 66 keys, including cursor control and 4 programmable keys
I/O Cassette, RS232 interface, serial port for printer and disk drive, joystick and plug-in cartridge ports, VHF television and video outputs

SOFTWARE
Operating system
Commodore's own; CP/M optional with Z80 cartridge
Languages Basic in ROM; other languages available under CP/M
Applications Numerous games and much other home software; some business packages; large range of software under CP/M

USE
Mostly home although some business and education

PRICE
Approx. £230

TECHNICAL SPECIFICATION

HARDWARE
CPU 6502
RAM 16 k (Model 4016), or 32 k (Model 4032)
ROM 18 k, containing Basic and operating system
Display 25 lines x 40 characters with pre-programmed graphics characters
Keyboard 74 keys including numeric pads and cursor control
Disks Optional single 5½ in on Model 4016, twin 5½ in on Model 4032
I/O Parallel port, IEEE-488

SOFTWARE
Operating system Commodore's own in ROM
Languages Basic in ROM; other languages available
Applications Large range of educational software and some business software

USE
Mainly educational, some home and business

PRICE
Approx. £570
8000 SERIES

The 8000 Series micros, the 8032 and the 8096, are enhanced versions of the 4000 Series, which are in turn descended from the now obsolete PET. The main improvements made to the 8000 Series are the enlarged RAM, 32 k (Model 8032) and 96 k (Model 8096), and the extended 80-character display.

There are no particularly exciting features but, as well-established machines that have been on the market for some time, the 8000 Series have a lot to recommend them in terms of reliability and availability of good quality software. They are also comparatively cheap, making them a good choice for small businesses and for educational applications.

The 8000 Series, like other Commodore micros, have a built-in IEEE-488 interface. This allows you to connect up to laboratory apparatus and other equipment, a particularly attractive feature for all scientific users.

Both the 8032 and the 8096 are due to be re-styled in 1983. They will be fitted into more up-to-date cases similar to those of the P and B Series, but the inner workings of the computers will remain largely the same.

TECHNICAL SPECIFICATION

HARDWARE
CPU 6502
RAM 32 k (Model 8032), 96 k (Model 8096)
ROM 18 k, containing Basic interpreter and operating system
Display 25 lines x 80 characters, 128 pre-programmed graphics characters
Keyboard 73 keys including numeric pad and cursor control
I/O Parallel port, IEEE-488 port, twin cassette ports, expansion connector for disk drives

SOFTWARE
Operating system Commodore's own in ROM
Languages Basic in ROM
Applications Huge range of applications software
USE Primarily a small business system but also useful in education and science
PRICE Approx. £780

B AND P SERIES

Commodore's latest machines, the B (700) Series and the P (500) Series were announced in April 1982, but the B was not available to buyers until the end of that year, and the P was planned to reach the stores in mid 1983.

Although these machines incorporate many up-to-date features, they are out of step with many current micros in retaining an 8-bit processor when most others are using 16-bit CPUs. Most users will not see much difference in performance compared with a 16-bit machine, but as more sophisticated software becomes available allowing full access to large capacity memory, it is possible that 8-bit users will be at a disadvantage. Commodore has tried to compensate by planning several add-ons including a Z80 card to enable you to use CP/M software and a 16-bit processor board to give access to 16-bit programs.

The more powerful B, to which the technical specification below relates, is aimed primarily at the business sector and can handle all standard business applications including word processing, stock control, and accounting packages.

The P is being marketed as a personal computer specifically for scientific and professional applications.

TECHNICAL SPECIFICATION

HARDWARE
CPU 6509 (6502 software-compatible)
RAM 64 k, expandable to 256 k
ROM 28 k, containing operating system and Basic interpreter
Display 25 lines x 80 characters; optional integral green phosphor monitor
Keyboard 94-key type writer-style, including numeric pad and cursor control; 10 programmable keys
Disks Twin integral 5½ in floppy disk drives

I/O IEEE-488 and RS232 ports, cassette port, user port, slot for plug-in processor card

SOFTWARE
Operating system Commodore's own in ROM
Languages Basic in ROM; others optional with CP/M and 16-bit processor add-ons
Applications Wide range of business software available

USE General business
PRICE Approx. £1150/£700 (700/500)
Originally a manufacturer of leather goods, Tandy took the unusual step of diversifying into electronics when it merged in 1963 with the Radio Shack chain of electronics hobbyists stores.

After Apple and Commodore, Tandy Corporation (trading as Radio Shack in the USA and many other countries) was the third of the founders of the microcomputer industry. Like its predecessors, it had a huge success with its first machine. The TRS-80 Model I was a modular computer which had a separate keyboard, screen, and expansion box. The latter component, besides holding extra memory, provided input for a printer, disk drives, and other peripherals. A further similarity with the hugely popular Apple II and Commodore PET was that the TRS-80 gave rise to a plethora of companies producing software and hardware that was compatible with the machine.

The Model II, a far larger business machine, was much less successful than its precursor, mainly because Tandy chose to write its own operating system instead of using CP/M, the industry standard. The rival machines from Apple and Commodore were also unsuitable for CP/M, being based around the 6502 processor chip, but Tandy did not take advantage of this fact.

**TRS-80 MC-10**

Tandy's new TRS-80 MC-10 (also known as the Micro Color Computer) has a large name for a little computer. Its measurements are, in fact, 2 x 7 x 8.5 ins, and at first glance it is not unlike the Sinclair ZX Spectrum, albeit with an improved keyboard.

The MC-10 allows two forms of keyword entry - either by typing words in full or by holding down the control key to allow access to the vast majority of available statements and commands.

The MC-10's on-board 4 k of RAM can be increased to a reasonable 20 k, but this is about the limit of expansion it is capable of. The MC-10 has an RS232C interface, which enables it to be connected to any Tandy printer with a serial interface. In fact, any suitable printer can also be used as long as it line feeds automatically with every carriage return.

**TRS-80 COLOR**

The TRS-80 Color computer is at the bottom of Radio Shack's range of home computers. It is principally a games machine, but word processing and home management packages are also available.

The TRS-80 Color has 16 k of RAM, expandable to 32 k. A Basic interpreter is installed in the 16 k ROM. It links up to a domestic television set and has sound and color graphics as standard.

The TRS-80 Color computer is a good choice as a family entertainment machine or as a first computer for anyone who wants to learn about programming.

**TECHNICAL SPECIFICATION**

**HARDWARE**

CPU 6809
RAM 16 k, expandable to 32 k
ROM 16 k
Display 16 lines x 32 characters; 192 x 256 one-color and background color
Keyboard 53 keys including cursor control
Cassette 1500 baud
I/O RS232 serial port, two joystick connectors

**SOFTWARE**

Operating system Radio Shack's own in ROM
Languages Basic; optional extended basic
Applications Games and some home management packages

**USE**

Home

**PRICE**

Approx. £180

---

**TECHNICAL SPECIFICATION**

**HARDWARE**

CPU 6809
RAM 16 k, expandable to 32 k
ROM 16 k containing Basic
Display 16 lines x 32 characters; graphics 16 x 32 eight colors, 192 x 256 one-color and background color
Keyboard 53 keys including cursor control
Cassette 1500 baud
Disks Optional, up to four 5¼ in floppy, capacity 165 k

**SOFTWARE**

Operating system Machine code in ROM
Languages Microsoft Basic, assembler
Applications Games, some educational software

**USE**

Home

**PRICE**

Approx. £80
TRS-80 MODEL 100

The Model 100 represents Tandy's first foray into the hotly competitive field of portable computers. It is a strong rival for, above all, the Epson HX-20 and Texas Instruments' CC-40.

Designed to fit neatly in a briefcase, the Model 100 measures 11 inches x 8 1/2 inches and weighs only 3 1/4 lb. Either AC power or four internal NiCad batteries can be used to drive the machine.

The keyboard is of the proper typewriter kind and has smaller additional keys for special functions. The display is very legible and strong rival for, above all, the Epson HX-20 and Texas Instruments' CC-40.

The Model 4 from Tandy has now effectively superseded the Model III although remaining completely compatible with all Model III software, using as it does the TRSDOS 6.0 operating system. More importantly, from the business point of view, the Model 4 is equally at home with CP/M, allowing the user access to the 3000 plus programs written for this widely used operating system.

The TRS-80 Model 4 is available in three configurations: 24 lines, single disk with 64 k RAM; 24 lines, twin disks, 64 k RAM; and a starter model with no disk drives, 16 lines, and running Model III Basic. Bear in mind, though, if you are thinking of economy, that CP/M requires a minimum of 64 k. The disk-drive versions can be upgraded to 128 k using a 64 k RAM pack. An 80 character text line should be sufficient for most business applications.

In appearance the Model 4 is not much different to the Model III. Gone, however, is the spray-paint finish of the Model III and instead the Model 4 incorporates a molded, and therefore permanent, cream finish. Additional keys on the nondetachable keyboard are CTRL, used with other keys to provide a set of control commands, CAPS, to lock the display into either upper or lower case, and three special-function keys found on the numeric keypad.

Included only in the price is the Model 4 TRSDOS operating system and Basic interpreter.

TECHNICAL SPECIFICATION

SOFTWARE

Operating system Tandy's own
Languages Microsoft Basic
Applications Telecommunications, personal time and file management

USE

Personal computing, portable word processing, file keeping, and telecommunications

PRICE

Approx. £500 (with 8k of RAM)
Approx. £650 (with 24k of RAM)

SOFTWARE

Operating system LDOS, CP/M Plus, TRSDOS
Languages Fortran, Cobol, Pascal, Forth, Microsoft Basic
Applications Word processing, financial planning, data management, accounts systems

USE

Home, small business

PRICE

Approx. £1500

TECHNICAL SPECIFICATION

HARDWARE

CPU 8085 (8-bit CMOS version)
RAM 8k or 24k, expandable to non-volatile 32k
ROM 32k (containing applications software)
Display 8 lines x 40 characters LCD; also usable as 240 x 60 dot matrix
Keyboard 76 keys including eight programmable-function keys, four command and four cursor-control keys
I/O Built-in direct-connect modem; interfaces for standard cassette recorder, parallel printer, bar-code reader, and RS232 devices
TRS-80 MODEL 12

The TRS-80 Model 12 is a powerful business machine similar in appearance to the more expensive Model 16. Its price, however, places it in the same area of the market as the established Model II. Furthermore, it is designed to run all the software originally created for the latter. The Model 12 does not break new ground in the business machines field, but it has the benefit of the large and reliable network of Tandy (Tandy Radio Shack in the USA) dealers behind it.

The detachable keyboard has 82 keys, including a numeric pad for arithmetical applications and eight special-function keys. Tandy launched a new 12 megabyte hard disk drive at the same time as it announced the Model 12. The system is also adaptable for use with the Model 16 and the Model II. The additional storage capacity, along with an extra 16 k of RAM, make the Model 12 a very good proposition for all business tasks involving large information files and complicated transactions. Such operations as inventory control, large spreadsheet planning, and medical record keeping are well within the capabilities of the enhanced machine.

TECHNICAL SPECIFICATION

HARDWARE
CPU Z80A
RAM 80 k
Display 24 lines × 40 or 80 characters, upper and lower case; 32 graphics characters
Keyboard 82 keys including numeric keypad and eight special-function keys
Disks One or two 8 in floppy disks, capacity up to 1.25 megabytes; double-density and double- or single-sided optional 12 megabyte hard disk drive available
I/O Optional card cage includes six expansion slots, a standard parallel port; two RS232 serial and one external disk bay port are also included

SOFTWARE
Operating system Tandy's own operating system (TRS-DOS 4.2) will run all Model II software
Languages Basic
Applications Word processing, spreadsheet, general business packages
USE
Business

PRICE
Approx. £2400 (with one 1.25 megabyte disk drive)

TRS-80 MODEL 16

The T RS Model 16 is Radio Shack's most powerful business machine. Based on the latest 16-bit processor technology, it also incorporates a Z80 chip allowing access to existing Radio Shack 8-bit software. This is a particularly useful feature as it may be some time before there is a comparable amount of 16-bit software available.

The Model 16 can be used on its own as a single-user computer, or converted by installing a Xenix operating system to create a multi-user network. Although the 16-bit processor is powerful enough to overcome the response-time disadvantages of 8-bit multi-user systems, Xenix can be difficult to use for non-experts.

However, as a single-user micro, the Model 16 is an attractive choice. It comes with a high-resolution 12 in monitor and one or two built-in 8 in disk drives. The typewriter-style keyboard has a numeric keypad and two programmable function keys. If offers good expansion possibilities for printers, graphics add-ons and hard disk drives. The standard 128 k RAM can be expanded up to 512 k.

The Model 16 is competitive in terms of price with similar machines and would make a particularly good buy for a company in need of extra computing power and already equipped with Model II micros.

TECHNICAL SPECIFICATION

HARDWARE
CPU 68000 (16-bit) and Z80 (8-bit)
RAM 128 k, expandable to 512 k
Display 24 lines × 40 or 80 characters; 32 business graphics characters
Keyboard 76 keys including numeric pad, cursor control and two programmable keys
Disks Single 8 in floppy, capacity 1.25 megabytes; optional up to three extra 8 in drives, up to four hard disks, capacity 8.4 megabytes

I/O Parallel printer port, twin RS232 serial ports

SOFTWARE
Operating system Radio Shack's own; optional Xenix
Languages Basic
Applications Large range of business software including word processing and spreadsheet; all Model II software is compatible

USE
Business

PRICE
Approx. £3800
In 1972 a young computer engineer, working in his garage, developed the first coin-operated video game to achieve widespread success. His business expanded rapidly and four years after this breakthrough his company, Atari, became part of Warner Communications Incorporated, achieving sales of $75 million by 1980.

Atari’s success in producing coin-operated video games encouraged it to go into production, in 1977, of its video computer system (VCS) for home use. This unit, used with a domestic television set, takes adapted versions of the games played in arcades and bars on Atari’s coin-operated machines, as well as original games. VCS sales are numbered in millions and those of the cartridges which contain the games in tens of millions. Atari frequently negotiates with film companies in order to offer the users of its arcade machines and VCS units exciting games based on the latest big-screen science-fiction hits.

While one division of the company – Coin Operated Games – produces arcade machines, and another – Consumer Electronics – makes VCS units and cartridges for them, a third division produces the home computers that have made Atari’s name significant in the world of micros. The first machines from the Home Computers division were launched in 1979 – the 400 and the 800, powerful and versatile machines for home use.

The great advantage of these machines over other microcomputers is that they are designed for the complete newcomer to computing. The user needs no knowledge whatsoever of programming in order to take advantage of Atari’s pre-written programs. The software gives easy access to subjects as diverse as stock analysis, conversational French, and chess. With business, education, and entertainment as the guiding principles, Atari have produced a wide range of programs specifically for the home user. The automation in Atari’s home machines does not prevent the more ambitious, however, from learning to program their own computer.

Atari home computers have gained a reputation for their excellent color, graphics, sound, and animation capabilities. These features, which benefit many types of program, but particularly the more spectacular games, are found in their most advanced form in Atari’s new XL Series machines.

---

### ATARI 400

The Atari 400 is designed principally as a games machine for the home, and therefore has few of the more sophisticated features of a personal computer for serious use. Its main advantage is that it offers access to the wide variety of games that Atari produce, although there are some home finance packages available as well. While Atari games are some of the best on the market, the graphics quality available on the 400 is not as good as that of the arcade machines because of the technical limitations of the domestic television set.

The Atari 400 has a touch-sensitive QWERTY keyboard designed to stand up to fairly rough family treatment. Most games packages are on cartridges which plug directly into the computer but programs can also be stored on a domestic cassette recorder. Various add-ons are available, including joysticks and games paddles, and an interface module to enable you to link up to a wide range of other accessories.

---

### TECHNICAL SPECIFICATION

**HARDWARE**
- CPU 6502
- RAM 16k
- ROM 10k, containing operating system
- Display 24 lines x 40 characters; 320 x 192 graphics; 16 colors, each with 16 intensities
- Keyboard 57 touch-sensitive keys including cursor control and four programmable keys
- Cassette Single cassette interface

**SOFTWARE**
- Operating system: Atari’s own in ROM
- Languages: Optional Basic, Pilot, assembler
- Applications: Large range of games/home software; some “business” software available

**USE**
- Home

**PRICE**
- Approx. £150
The Atari 800 is the upgraded version of the 400 and is similar to that machine in many ways. It is also principally a games machine but has several improved features that enable it to handle more complex software. It has 48 k of built-in RAM and a disk interface allowing you to link up to a maximum of four Atari 810 disk drives for more efficient program storage. An Atari Basic cartridge is included in the standard store-bought package.

The Atari 800 has a proper typewriter-style moving keyboard with 57 keys, including cursor control and four programmable function keys. In addition to the games paddle and joystick jacks, the machine has a video output socket to allow you to connect up to a monitor in order to improve the quality of the graphics display. Other possible peripherals for the 800 include a printer and a multiple interface module.

**ATARI 800**

The Atari 800 is the upgraded version of the 400 and is similar to that machine in many ways. It is also principally a games machine but has several improved features that enable it to handle more complex software. It has 48 k of built-in RAM and a disk interface allowing you to link up to a maximum of four Atari 810 disk drives for more efficient program storage. An Atari Basic cartridge is included in the standard store-bought package.

The Atari 800 has a proper typewriter-style moving keyboard with 57 keys, including cursor control and four programmable function keys. In addition to the games paddle and joystick jacks, the machine has a video output socket to allow you to connect up to a monitor in order to improve the quality of the graphics display. Other possible peripherals for the 800 include a printer and a multiple interface module.

**TECHNICAL SPECIFICATION**

**HARDWARE**
- CPU 6502C
- RAM 16 k (600XL); 64 k (800XL)
- ROM 24 k

**Display**
- 11 graphics modes, 5 text modes; 256 colors; 24 lines x 40 characters

**Keyboard**
- 62 keys, including four special-function keys, “Help” key, 29 graphics keys

**Cassette** Optional special Atari recorder

**Disks** Optional disks with up to 4 x 88 k or 4 x 127 k capacity

**I/O** Software cartridge slot, parallel bus, TV output, two controller ports, serial I/O connector

**SOFTWARE**
- Operating system: Atari's own in ROM

**Languages**
- Basic; optional Pilot, assembler

**Applications**
- Large range of games and home software; some home-business packages available

**USE**
- Home

**PRICE**
- Approx. £160 (600XL)
- Approx. £250 (800XL)

Atari has launched a series of new computers to replace its proven 400 and 800 games machines. The series includes the 600XL and 800XL. These machines use the same chips as the earlier models—a 6502C plus several of Atari's own creation. An improvement over the 400 and 800 lies in their display capabilities: they have 16 modes as opposed to 12.

The keyboards on both machines are easy to use and have a row of general-purpose buttons in addition to the standard QWERTY keys. Three memory storage systems—cartridge, cassette, and disk—are available. The maximum capacity is four disks holding 127 k each. For the 600XL, a 64 k RAM module brings the machine into line with its recently launched competitors.

**ATARI 600XL and 800XL**

Atari has launched a series of new computers to replace its proven 400 and 800 games machines. The series includes the 600XL and 800XL. These machines use the same chips as the earlier models—a 6502C plus several of Atari's own creation. An improvement over the 400 and 800 lies in their display capabilities: they have 16 modes as opposed to 12.

The keyboards on both machines are easy to use and have a row of general-purpose buttons in addition to the standard QWERTY keys. Three memory storage systems—cartridge, cassette, and disk—are available. The maximum capacity is four disks holding 127 k each. For the 600XL, a 64 k RAM module brings the machine into line with its recently launched competitors.

**TECHNICAL SPECIFICATION**

**HARDWARE**
- CPU 6502C
- RAM 16 k (600XL); 64 k (800XL)
- ROM 24 k

**Display**
- 11 graphics modes, 5 text modes; 256 colors; 24 lines x 40 characters

**Keyboard**
- 62 keys, including four special-function keys, “Help” key, 29 graphics keys

**Cassette** Optional special Atari recorder

**Disks** Optional disks with up to 4 x 88 k or 4 x 127 k capacity

**I/O** Software cartridge slot, parallel bus, TV output, two controller ports, serial I/O connector

**SOFTWARE**
- Operating system: Atari's own in ROM

**Languages**
- Basic; optional Pilot, assembler

**Applications**
- Large range of games, educational and semi-business software

**USE**
- Home

**PRICE**
- Approx. £160 (600XL)
- Approx. £250 (800XL)

Logo

**Applications**
- Large range of games, educational and semi-business software

**USE**
- Home

**PRICE**
- Approx. £160 (600XL)
- Approx. £250 (800XL)
Choosing your computer · Module 47

A GUIDE TO OTHER MACHINES

ACT APRICOT

The all-British Apricot is a powerful 16-bit business machine that has the advantage of software-compatibility with the Sirius 1 and the IBM Personal Computer.

The machine uses the powerful 8086 processor and can also take a fast math processing chip, the 8087. It benefits from the latest versions of the foremost 16-bit operating systems — MS-DOS II and Concurrent CP/M-86. The latter allows you to perform more than one task at a time.

A novel feature of the Apricot is a small LCD screen on the keyboard. Apart from assisting the user by presenting simple options while running applications, it acts as a calculator, a calendar, a clock, and a window into the monitor screen.

AQUARIUS

The Aquarius is based on the tried-and-tested Z80A microprocessor and has 8 k of ROM and a modest, although expandable, 4 k of RAM. Its language is built-in Microsoft Basic. The keyboard has 49 moving keys and frequently used Basic instructions can be entered by depressing a control key at the left of the keyboard at the same time as the appropriate instruction key.

The display has 16 colors, 24 lines x 40 characters of text format, and graphics resolution of 320 x 192. Including the full ASCII set, there are 256 characters. By using the mini-expander the one-channel sound of the basic machine can be increased to three channels. This is not the main function of the mini-expander, however. It is designed to take two memory expansion cartridges.

Memory can also be expanded by 4 k or 16 k increments to a maximum of 52 k without using the mini-expander. In this case the memory expansion cartridges are inserted directly into ports on the machine.

Other hardware, including a printer that operates at 80 characters a second and a modem, are planned. Games software is also promised.

In common with other recent micros, the Aquarius is compact and light. It measures approximately 13 x 6 x 2 in and weighs 3 lb.

SOFTWARE
Operating system Concurrent CP/M-86 and MS-DOS II;
optional UCSD p-System, BOS
Languages Two Basic interpreters; optional Pascal, Fortran, Cobol; other languages available under CP/M-86 and MS-DOS
Applications Database, communications, graphics, utilities, optional electronic mail, word processing, financial, and very large range of other general business packages

USE Business
PRICE Approx. £1500

TECHNICAL SPECIFICATION
HARDWARE
CPU 8086; optional 8087 math processor
RAM 256 k, expandable to 768 k
Display 25 lines x 80 characters and 50 lines x 132 characters; 800 x 400 graphics
Keyboard 96 keys including cursor control, 8 pre-set function keys with LCD labeling, numeric pad, editing keys
Disks One single-sided 3½ in microfloppy, capacity 315 k, optional twin single-sided microfloppies, or 1 or 2 double-sided microfloppies, capacity 720 k per disk
I/O Centronics printer port, RS232 serial port, optional modem board with auto-dial

SOFTWARE
Operating system Aquarius’s own
Languages Microsoft Basic
Applications Games and home

USE Home
PRICE Approx. £80

Cassette Optional data recorder
I/O Memory expansion cartridge ports, RS232-style printer interface
### BBC MICROCOMPUTER

Designed and manufactured by Acorn Computers Limited, the BBC Microcomputer offers a combination of power and expandability with a language that can be initially confusing. This language, BBC Basic, is a development of Acorn's version of Basic. Program listings in this version of Basic appear non-standard because of the extensions added to facilitate the use of the machine's graphics and sound and by the facility for abbreviating keywords to a single letter.

Despite these features, the machine's power and its impressive graphics and sound capabilities combine to create one of the best-selling computers in the UK.

There are in fact two BBC machines - Models A and B. Both use the 6502 processor chip, but using an interface known as the Tube you can add a second chip, even a 16-bit one like the 68000.

Model A has 16 k of RAM, but for an extra £100 Model B provides twice that capacity and better graphics.

### DECRAINBOW

This computer has the DEC-developed CP/M-86/80 operating system, and so allows the use of both CP/M-80 and CP/M-86 software.

The Rainbow uses the same 16-bit processor as the IBM and Victor microcomputers. The development of software for 16-bit machines is still lagging behind that for 8-bit computers. But the combination of the available 16-bit software and 8-bit CP/M software provides a fair degree of versatility with a machine like the Rainbow, which has both a 16-bit processor, the 8088, and an 8-bit processor, the extensively used Z80. The machine itself decides which processor is needed for the program offered to it and then puts it into operation with the appropriate operating system.

The disk drive system offered by the Rainbow is unique in that a single drive can hold two disks.

### TECHNICAL SPECIFICATION

#### HARDWARE

- **CPU**: 6502
- **RAM**: Model A 16 k, Model B 32 k
- **ROM**: 32 k, containing Basic and operating system

#### Display

Eight display modes (not all on Model A) for a range of color combinations, text capabilities (including Teletext-compatible and 32 lines of 30 characters) and graphics up to $640 \times 256$ resolution; color monitor required for high-resolution graphics and text.

- **Keyboard**: 74 keys, including cursor control and 10 programmable keys
- **Cassette**: 300 and 1200 baud
- **Disks**: Optional, single or twin 5½ in floppy

#### I/O

- **RS232 serial (compatible with RS232)**, Centronics parallel, 8-bit parallel general-purpose interface, extension interface for Prestel/Viewdata, the Tube interface for additional processors, optional network interface (Acorn Econet)

#### SOFTWARE

- **Operating system**: Acorn's own in ROM; CP/M promised
- **Languages**: Basic in ROM; others promised
- **Applications**: Range of games and general home software; educational packages

#### USE

Home and education

#### PRICE

Approx. £299 (A)  
Approx. £399 (B)
The Dragon 32 is manufactured in Wales with all-British components. It is compact and lightweight but it is of a sturdy construction. The Dragon's combination of a substantial RAM of 32 k, a proper keyboard, plus sound and color capabilities, has made it a very popular computer.

Particularly impressive are the Dragon's graphics. There are nine colors and the Basic interpreter incorporates commands which, if you want to write your own games programs, will allow you to benefit to the full from the versatile graphics.

The RAM is constituted in such a way that you can, at any one time, have eight "pages" of display in use, moving between them at will. You can even display several pages at once. However, the more pages that are in use, whether on the screen simultaneously or not, the lower the graphics resolution and the fewer the number of colors you can have on each page.

Sound is output through your television set, which makes volume adjustment easier than it is with home computers with integral loudspeakers. Note: The new Dragon 64 is now available with an extra serial interface.

---

The British-built Elan Enterprise is a recently introduced machine with a wide range of attractions for the home user. The computer's features, which include a word processor, very high resolution graphics, 256 colors, and 50 × 80-character text handling, compete with those of machines priced at around £1000. Yet the Elan Enterprise is offered at a very reasonable £200. At present software is in short supply, but, using the optional disk drives, the machine will run the CP/M operating system, for which there exists a wide selection of software, including many programming languages. Furthermore, the outstanding capabilities of the machine should soon lead to widespread adoption by software producers.

The Elan Enterprise has American National Standard Basic in ROM and also allows you to run programs in the widely used Microsoft Basic language with little or no modification.

The machine's styling is unusual in incorporating a joystick instead of keys to control the cursor. Another unusual, if not unique, feature is the text editor that is built into the operating system. This allows word processing without the need for additional software. However, to benefit from the extra-wide display mode (used in word processing) a high-resolution monitor is essential.

---

**DRAGON 32**

**THE DRAGON 32**

The Dragon 32 is manufactured in Wales with all-British components. It is compact and lightweight but it is of a sturdy construction. The Dragon's combination of a substantial RAM of 32 k, a proper keyboard, plus sound and color capabilities, has made it a very popular computer.

Particularly impressive are the Dragon's graphics. There are nine colors and the Basic interpreter incorporates commands which, if you want to write your own games programs, will allow you to benefit to the full from the versatile graphics.

The RAM is constituted in such a way that you can, at any one time, have eight "pages" of display in use, moving between them at will. You can even display several pages at once. However, the more pages that are in use, whether on the screen simultaneously or not, the lower the graphics resolution and the fewer the number of colors you can have on each page.

Sound is output through your television set, which makes volume adjustment easier than it is with home computers with integral loudspeakers. Note: The new Dragon 64 is now available with an extra serial interface.

---

**ELAN ENTERPRISE**

**THE ELAN ENTERPRISE**

The British-built Elan Enterprise is a recently introduced machine with a wide range of attractions for the home user. The computer's features, which include a word processor, very high resolution graphics, 256 colors, and 50 × 80-character text handling, compete with those of machines priced at around £1000. Yet the Elan Enterprise is offered at a very reasonable £200. At present software is in short supply, but, using the optional disk drives, the machine will run the CP/M operating system, for which there exists a wide selection of software, including many programming languages. Furthermore, the outstanding capabilities of the machine should soon lead to widespread adoption by software producers.

The Elan Enterprise has American National Standard Basic in ROM and also allows you to run programs in the widely used Microsoft Basic language with little or no modification.

The machine's styling is unusual in incorporating a joystick instead of keys to control the cursor. Another unusual, if not unique, feature is the text editor that is built into the operating system. This allows word processing without the need for additional software. However, to benefit from the extra-wide display mode (used in word processing) a high-resolution monitor is essential.

---

**TECHNICAL SPECIFICATION**

**HARDWARE**

CPU 68000
RAM 32 k
ROM 16 k, containing Basic interpreter and operating system

**DISPLAY**

32 lines x 16 characters; graphics resolutions from 64 x 32 to 256 x 192 (nine colors); not all colors available at higher resolutions

**KEYBOARD**

53 keys, including cursor control

**CASSETTE**

Automatic motor on/off under program control

**I/O**

Centronics parallel, expansion port for plug-in ROM packs, joystick ports

**SOFTWARE**

Operating system: Dragon's own in ROM
Language: Basic in ROM
Applications: Good selection of games and general home software on cassette and in ROM packs

**USE**

Home

**PRICE**

Approx. £175

---

**TECHNICAL SPECIFICATION**

**HARDWARE**

CPU Z80
RAM 64 k
ROM 32 k, containing Basic and operating system

**DISPLAY**

25 lines x 40 characters; optional other modes up to 50 lines x 80 characters (with memory expansion and monitor)

**KEYBOARD**

69 keys including programmable-function and editing keys; cursor control via built-in joystick

**CASSETTE**

1200 baud

**DISKS**

Optional 3½ in micro-floppy, capacity 350 k per disk

**I/O**

Parallel, RS232 serial, network, cassette, RGB monitor, UHFTV and cartridge ports

**SOFTWARE**

Operating system: EROS (Elan ROM Operating System); optional CP/M

Languages: Basic with extensions to include Microsoft compatibility; optional Forth and Lisp

Applications: Built-in word processor in ROM, programming tutorial; optional games and home software

**USE**

Home, educational, small business

**PRICE**

Approx. £200
ELECTRON

The Electron is a long-awaited machine from the creators of the Atom and the BBC Micro. Comparison with the BBC Model B is inevitable, yet at half the price the Electron is very good value. On the other hand, the cheaper machine lacks the BBC's teletext mode, provision for extra ROMs, a second cassette speed, and various interfaces. It also operates as slowly as one-third of the speed of the BBC Micro, depending on the job in hand.

Like the BBC machine, the Electron offers 32 k of RAM and 32 k of ROM. Its internal layout, however, is much simplified, being based around a 6502A central processor, three other processors, and four RAM chips. The Electron’s huge 68-pin logic array takes care of the tasks of most of the BBC Micro's numerous chips.

The housing and keyboard are of a solid design. The latter is of the typewriter kind and most of the keys have a pre-defined function printed on their front. This means that Basic keywords—the Electron uses Acorn Basic, which is virtually identical to BBC Basic—can be entered at single keystroke. An additional bonus is that number keys double as user-definable function keys.

Software is no problem owing to the Electron’s compatibility with the BBC Micro, which is very well provided for. Acorn promise interfaces and ports for printers, floppy disks, and other peripherals.

EPSON HX-20

Until recently the name Epson signified printers for the microcomputer market. Epson’s output of dot matrix printers is still greater than those of all its rivals put together, but the company has now become a manufacturer of microcomputers too.

The HX-20 is the first of its machines to become available in the West. It is a compact computer with a raised keyboard. By comparison with various other machines described as portable, the HX-20 is truly worthy of the name. But in order to keep the machine compact the designers have produced a display that is too small. With only four lines of 20 characters each it is not large enough for word processing or many other serious functions. However, the HX-20’s wide range of facilities should be set against this shortcoming. Apart from the Hewlett-Packard HP75C, which is far more expensive, the HX-20 offers more facilities than any other true portable. Apart from the proper keyboard—a pleasure to find in a portable machine—the HX-20 has a RAM memory of a realistic size (both RAM and ROM are expandable, the former from 16 k to 32 k, and the latter from 32 k to 64 k), serial interfaces, and the options of a diminutive built-in printer and a micro-cassette unit. A disk-drive unit is to become available, but it will need a main power supply. Epson’s separate dot matrix printers can be used with the HX-20, as can a domestic television and an acoustic coupler.

TECHNICAL SPECIFICATION

SOFTWARE
Operating system Acorn’s own
Languages Acorn Basic in ROM; Lisp, Forth
Applications Games and general home software; educational programs planned
USE Home
PRICE Approx. £200

HARDWARE
CPU 6800-compatible
RAM 16 k, expandable to 32 k
ROM 32 k, expandable to 64 k
Display 4 lines x 20 characters, built-in LCD; 120 x 32 dots graphics
Keyboard 68 keys, including cursor control and 5 programmable keys
Cassette Optional built-in microcassette unit which operates under full computer control
I/O 1 twin RS232 ports; connectors for optional built-in micro printer unit

SOFTWARE
Operating system Epson’s own in ROM
Languages Basic in ROM
Applications General business packages available as plug-in ROMs and on tape
USE Business
PRICE Approx. £420

TECHNICAL SPECIFICATION

HARDWARE
CPU 6800-compatible
RAM 16 k, expandable to 32 k
ROM 32 k, expandable to 64 k
Display 4 lines x 20 characters, built-in LCD; 120 x 32 dots graphics
Keyboard 68 keys, including cursor control and 5 programmable keys
Cassette Optional built-in microcassette unit which operates under full computer control
I/O 1 twin RS232 ports; connectors for optional built-in micro printer unit

SOFTWARE
Operating system Epson’s own in ROM
Languages Basic in ROM
Applications General business packages available as plug-in ROMs and on tape
USE Business
PRICE Approx. £420
The world's largest computer printer manufacturer, Epson entered the microcomputer market with its portable HX-20. It has now added the normal-sized QX-10 to its range of products. This machine is an 8-bit model, launched at a time when most microcomputer companies are busy producing 16-bit machines. However, its 8-bit processor does not prevent it from giving access to a huge range of tried-and-tested software, and it is certainly equal to a wide variety of business tasks.

The keyboard is of good quality, with 103 keys, including 14 programmable keys. The display is excellent and has 16 built-in character sets. The graphics capabilities include the extremely rare facility of a 16:1 zoom-in. A further refinement allows you to display several type-styles simultaneously.

The amount of RAM that can be addressed by the CPU is an impressive 256 k, whereas a maximum of 64 k is common in other 8-bit machines.

This is a small but deceptively powerful machine, being based on the Motorola 68000 CPU. This processor gives it the power of many of the smaller minicomputers currently available.

Many computers with the 68000 CPU use the operating system known as Unix. Bell Laboratories created this as a software development system, with the aim of easing the burden of the computer programmer. But ease of use for the programmer can lead to the opposite for the user. Fortune adapted Unix for its users, so that a "menu" of programs and data files appears on the screen. The user then selects the required function. Yet this modified Unix still allows you to switch from, say, word processing to looking for an item in a database without all the problems associated with a less sophisticated operating system and a less powerful CPU.

Fortune's version of Unix also allows multi-use of the CPU. While this facility is well-established in the world of big computers, it is not so great an advantage with micros, where the CPU is a relatively cheap component. The alternative to multi-use of one CPU is to link several machines in a network in which there is a CPU for each user but hard disks and printers are shared.

---

### TECHNICAL SPECIFICATION

**HARDWARE**
- **CPU**: Z80
- **RAM**: 192 k, expandable to 256 k
- **Display**: 25 lines x 80 characters, 640 x 400 graphics
- **Keyboard**: 103 keys, including numeric pad, cursor control, and 14 special function keys
- **Disks**: Twin 5¼ in floppy, capacity 640 k
- **I/O**: Centronics parallel, RS232; optional networking and "universal" interfaces

**SOFTWARE**
- **Operating system**: CP/M-compatible
- **Languages**: Basic; others optional under CP/M
- **Applications**: Wide range under CP/M

**USE**
- Business and education

**PRICE**
- Approx. £1740

### TECHNICAL SPECIFICATION

**HARDWARE**
- **CPU**: Motorola 68000
- **RAM**: 128 k, expandable to 1 megabyte
- **ROM**: 4-16 k, depending on which hardware options are chosen
- **Display**: 25 lines x 80 characters; 13 special characters for word processing; 15 graphics symbols; color monitor optional
- **Keyboard**: 99 keys, including 9 cursor-control, 15 numeric and 16 programmable keys
- **Disks**: 1-4 5½ in floppy disks, capacity 680 k per disk; 1-4 hard disks optional

**I/O**: RS232 serial port; optional RS232 and parallel ports, communications controller, Ethernet network interface

**SOFTWARE**
- **Operating system**: Unix-compatible, with Fortune's own "shell"
- **Languages**: Basic; optional Cobol, Fortran, Pascal, C
- **Applications**: Word processing, spreadsheet, integrated accounts, database

**USE**
- Business

**PRICE**
- Approx. £5500
FUTURE COMPUTERS FX20

The FX20 is the first machine from the new British company Future Computers. It is intended to be a tool for the "electronic office" and accordingly has built-in networking capabilities. Several of these business machines can be linked together to pass information between one another and to share the expensive facilities.

The FX20 uses the 8088 processor, a 16-bit component used also by the IBM Personal Computer and the Sirius 1. This arrangement provides the machine with a growing software base requiring little or no modification. Further versatility is available through the same plug-in cards - for additional RAM and color graphics - as used by the IBM machine.

Promised facilities include a high-resolution graphics board with a stunning 1280 × 500 display, a Telex interface, expanded memory, and additional I/O.

Unfortunately, even the largest companies fail to meet deadlines, having promised the availability of equipment on a particular date. While this observation does not imply a specific criticism of Future Computers, it is a good principle not to make a commitment to buying equipment that is not yet available, whatever the company involved.

TECHNICAL SPECIFICATION

HARDWARE
CPU 8088; optional maths processor and Z80
RAM 128 k, expandable to 1 megabyte
ROM 1 k
Display 25 lines × 80 characters; planned optional medium resolution 640 × 250 graphics; planned optional high-resolution 1280 × 500 graphics with high-resolution display unit
Keyboard 108 keys, including numeric pad, cursor control and 20 programmable keys
Disks Twin 5½ in floppy, capacity 800 k per disk
I/O RS232 serial printer port, network interface, RS232 communications port; optional 6 RS232 ports, IBM-compatible expansion bus, modem and Telex interfaces

SOFTWARE
Operating system CP/M-86; optional MS-DOS, Concurrent CP/M
Languages Optional Basic, Pascal, assembler; others from independent suppliers
Applications Word processing; optional database, spreadsheet, general business and maintenance control packages

USE
Business

PRICE
Approx. £1900

HEWLETT-PACKARD HP75C

In 1982 the truly portable computer made its appearance on the market. Like Epson's HX-20, the Hewlett-Packard HP75C can be described as a true portable. Yet even these machines lack a flat screen that gives the standard display of 24 lines of 80 characters, and micro-floppy disk drives are as yet not widely available. Until these refinements are standard the Epson and the Hewlett-Packard machines remain the frontrunners in portable computers.

Like other Hewlett-Packard products, the HP75C is very well constructed. It is a machine for the serious scientist or engineer, and is certainly not intended as an executive toy or for children.

The storage medium of the HP75C is unusual in that it comprises thin magnetic cards, which each hold 1.3 kbytes. These cards are read from and written on when pulled through a small slot in the computer. The system is effective in its simplicity, but the capacity is rather limited.

The language used in this machine is a powerful and user-friendly Basic. The CPU uses an 8-bit processor designed and manufactured by Hewlett-Packard. Also designed by the company is the HP-ll interface. This networking system allows the use of a digital cassette recorder, a video, a plotter, and a printer, among other peripherals.

TECHNICAL SPECIFICATION

HARDWARE
CPU 8-bit (designed and manufactured by Hewlett-Packard)
RAM 16 k, expandable to 24 k
ROM 48 k, containing Basic and operating system
Display Single-line liquid crystal, 32 characters
Keyboard 64 keys, including cursor control
Storage Magnetic cards with 1.3 k capacity per card
I/O HP-II interface

SOFTWARE
Operating system Hewlett-Packard's own
Languages Basic in ROM
Applications Owner's Pack containing address book, financial calculations, and a game
USE
Mainly scientific and engineering

PRICE
Approx. £730

185
The HP86A is an inexpensive machine by Hewlett-Packard standards but, as is usual with HP products, the engineering is of very high quality.

Hewlett-Packard’s prestige has been built on machines aimed at the scientific and engineering spheres. This situation grew out of the fact the company was already heavily involved in producing equipment for the laboratory.

To meet the needs of scientist and engineer, the machines have interfaces for connection to other Hewlett-Packard equipment, software designed to handle numerical data, and graphics that make it easy to display data in graph form.

Hewlett-Packard machines are very useful for business purposes too. The HP86A, especially if equipped with the CP/M module, can tackle most conventional business problems in addition to a range of scientific and engineering tasks.

There is a drawback, however. Adding the CP/M module precludes the use of what are probably the most attractive of the machine’s facilities: its powerful Basic and its graphics.

### TECHNICAL SPECIFICATION

**HARDWARE**
- CPU 8-bit (designed and manufactured by Hewlett-Packard)
- RAM 30 k, expandable to 576 k
- ROM 48 k, containing Basic interpreter and operating system

**Display**
- 16 or 24 lines × 80 characters; the display acts as a window into an area of up to 204 lines × 80 characters; various graphics modes up to 544 × 240 resolution

**Keyboard**
- 91 keys, including numeric pad, cursor control and 7 programmable keys

**Disks**
- Optional single or twin 5½ in floppy, 270 k capacity per disk

**I/O**
- Centronics parallel port, interface port for up to two disk drives; optional IEEE-488, RS232, HP-1L and GP-10 interfaces

**SOFTWARE**
- Operating system: Hewlett-Packard’s own in ROM; CP/M available with optional plug-in module

**Languages**
- Extended scientific Basic with graphics capability in ROM

**Applications**
- Extensive ROM-based library for scientific, engineering and statistical work; word processing and spreadsheet available under CP/M

**USE**
- Scientific, engineering and general business

**PRICE**
- Approx. £1200

---

**LYNX**

The LYNX is the first offering from the Cambridge-based company Computors. The machine has several impressive features, foremost among these being a memory which can be expanded from 48 k to a stupendous 192 k. The high-resolution graphics are also a boon and there is the added advantage that you can define the color of each pixel individually. The keyboard is typewriter-style, with 37 keys.

A CP/M operating system can be used with the Z80A processor chip and, together with the good range of I/O facilities, this makes the LYNX very good for business use. (CP/M is established as the industry standard operating system for 8-bit business machines.)

The LYNX at its most basic, on the other hand, is a competent home/hobbyist computer along the lines of the popular Atari 400. It will support a color as well as a monochrome monitor and has a Basic interpreter in ROM. At present Computors rely on other companies for games software and more is needed.

Single-sided 5½ in floppy disks are expected to be available around mid 1983, along with a disk controller.

### TECHNICAL SPECIFICATION

**HARDWARE**
- CPU Z80A
- RAM 48 k, expandable to 192 k
- ROM 16 k, containing Basic interpreter and monitor

**Display**
- 24 lines × 40 characters, 248 × 256 graphics, 8 colors; expandable, with extra RAM, to 24 lines × 80 characters and 248 × 512 graphics (color monitor required for this display)

**Keyboard**
- 57 keys, including cursor control

**Languages**
- Basic in ROM; Pascal, Forth, and Comal planned

**Applications**
- Games and home software and business programs (the latter for use under CP/M, available from independent suppliers)

**USE**
- Home/hobbyist and business

**PRICE**
- Approx. £230
NEC APC

The APC (Advanced Personal Computer) is the latest machine from NEC, the makers of the PC8000 series. It is a business machine with sufficient flexibility to be an asset to companies of widely varying size.

The keyboard is easy to use and has 109 keys, including a full numeric pad for arithmetical work and 22 programmable-function keys which, when used with the Shift key, provide a further 22 user-definable functions. It is detachable from the housing which also holds the main circuitry, the monitor, and the disk drives.

The APC is supplied with one 8 in floppy disk, holding up to 1 megabyte. A second 8 in disk is optional, and a 5½ in winchester-style hard disk is also available. This gives a formatted storage capacity of 9.27 megabytes.

NEC produces a large quantity of software for the APC and there is a wide range from other suppliers. Most business applications are possible. In fact, since the APC supports both the CP/M-86 and the MS-DOS operating systems, the range of business software is extremely wide.

TECHNICAL SPECIFICATION

HARDWARE
Cpu Z80-compatable
RAM 32 k, expandable to 64 k with external expansion unit
ROM 24 k, containing Basic, expandable to 32 k
Disk 20 lines x 80, 72, 40, or 36 characters, 160 x 100 graphics, 8 colors
Keyboard 82 keys, including numeric keypad, and 5 programmable keys
Cassette 600 baud
Disks Optional twin 5½ in single-sided, single-density floppy disks, capacity 143 k per disk
I/O Centronics parallel printer port, serial interface; optional expansion unit gives twin RS232 and single IEEE-488 ports
SOFTWARE
Operating system NEC's own; CP/M optional
Languages Basic; others available with CP/M option
Applications Word processing and other business software available; wide range with CP/M option
USE Small business
PRICE Approx. £1200

NEC PC8000

This computer is the top-selling microcomputer in Japan, where its price has made it a very attractive proposition for the home user. The PC8000 was an advanced machine when it first came on the market. It was one of the very few business microcomputers to offer color as a standard facility. Color - in this case there is a choice of eight colors - represents an additional information channel. This is very useful for displaying graphs and pie charts in a visually attractive and easily assimilable way. The PC8000 offers the user the added benefit of a color word processor.

There is an extensive range of business software for the PC8000, including provision for sales, purchase and general ledgers, invoicing, stock control, and payroll. These can be used independently or as part of an integrated accounting system.

The machine is not without its drawbacks, however. While many other manufacturers offer 64 k of RAM at competitive prices, the PC8000 has only 32 k as the basic provision. To increase this to 64 k, an external expansion unit is required.

TECHNICAL SPECIFICATION

HARDWARE
Cpu 8086
RAM 128 k, expandable to 256 k
ROM 8 k; optional 32-bit
Arithmetic Processing Unit
Display 25 lines x 80 characters, 640 x 475 graphics resolution in 8 colors; 224 predefined characters and 256 user-definable characters
Keyboard 109 keys including numeric keypad and 22 function keys (the Shift key allows 22 further user-definable functions)
Disks One 8 in floppy is standard, a second is optional; the double-sided, double-density disk allows 1 megabyte of mass storage; 5½ in hard disk giving 9.27 megabytes also available
SOFTWARE
Operating system CP/M-86, MS-DOS
Languages Basic, Cobol, Fortran, Pascal, and most other business languages
Applications General accounting, word processing, database management, and other business software available
USE Business
PRICE Approx. £2000

PRIVATE

Choosing your computer · Module 47
NORTH STAR
ADVANTAGE

One of the first companies to play an important part in the microcomputer revolution was North Star. The Horizon, a machine intended for both business and scientific use, was its first model. This is still on the market, but the Advantage is a more up-to-date computer.

The Advantage is an impressive machine, its business-graphics capabilities being particularly notable. In order that the best use may be made of the graphics hardware, North Star has developed not only its own version of the CP/M operating system but an operating system of its own creation, G-DOS. Both of these systems work with the Advantage's Graphics Basic. The graphics resolution is moderate - 640 x 240 - but adequate for business.

Another feature of the Advantage is the provision for additional I/O controller cards. These are plugged in via six internal slots.

In two ways the Advantage goes against current trends. First, unlike some of the latest 16-bit machines, it does not have color. Secondly, the screen, keyboard, disks and electronics are housed in a single unit, whereas the "three-box" system is increasingly popular with other manufacturers. This configuration prevents adjustment of the screen and keyboard in relation to each other, but it does increase the machine's portability.

North Star has promised a hardware math processor board which will speed up math-based operations.

OLIVETTI M20

Olivetti, among the world's largest office equipment suppliers, entered the microcomputer market in 1982 with the M20.

Olivetti's choice of chip for the CPU is surprising. The Zilog Z8001 is a powerful 16-bit chip which performs particularly well in mathematical work, but no standard operating system has been created for it. Therefore Olivetti wrote its own: PCOS. Being a big enough company to provide a wide range of suitable software, Olivetti has avoided the problem of many manufacturers who choose an uncommon operating system for their machines: a shortage of software. In addition to Olivetti's own software, packages are available for the M20 from other manufacturers.

Three machines comprise the M20 range, the basic model having 128 k of RAM and a single floppy disk drive. An optional colour graphics system is available for all these models.

TECHNICAL SPECIFICATION

HARDWARE
CPU Z80; optional plug-in math processor promised
RAM 64 k, with additional 20 k for display
ROM 2 k
Display 24 lines x 80 characters, 640 x 240 graphics
Keyboard 87 keys, including cursor control, numeric pad and 15 programmable keys
Disks Twin 5½ in floppy, capacity 360 k per disk; optional hard disk, capacity 5 megabytes, to replace one floppy
I/O Optional plug-in RS232 serial and parallel ports

SOFTWARE
Operating system CP/M with graphics extension, G-DOS (North Star's own system for graphics)
Languages North Star Graphics Basic (for use with G-DOS); optional Basic, Fortran and Cobol for CP/M, plus others from independent suppliers
Applications Business graphics package; optional word processing, mailing list, database, accounting
USE Business
PRICE Approx. £2000

OLIVETTI M20

OLIVETTI M20

Olivetti, among the world's largest office equipment suppliers, entered the microcomputer market in 1982 with the M20.

Olivetti's choice of chip for the CPU is surprising. The Zilog Z8001 is a powerful 16-bit chip which performs particularly well in mathematical work, but no standard operating system has been created for it. Therefore Olivetti wrote its own: PCOS. Being a big enough company to provide a wide range of suitable software, Olivetti has avoided the problem of many manufacturers who choose an uncommon operating system for their machines: a shortage of software. In addition to Olivetti's own software, packages are available for the M20 from other manufacturers.

Three machines comprise the M20 range, the basic model having 128 k of RAM and a single floppy disk drive. An optional colour graphics system is available for all these models.

TECHNICAL SPECIFICATION

HARDWARE
CPU Z8001
RAM 160 k, expandable to 244 k
ROM 2 k, expandable to 8 k
Display 16 lines x 64 characters or 25 lines x 80 characters; 512 x 256 or 480 x 256 graphics; optional color display giving 8 colors
Keyboard 72 keys, including numeric pad
I/O RS222 serial, Centronics parallel; optional twin RS232 and single IEEE-488
Disks Twin 5½ in floppy disks, capacity 320 k per disk

SOFTWARE
Operating system PCOS (Professional Computer Operating System) Olivetti's own
Languages Basic
Applications Word processing, spreadsheet, communications, general business packages plus specialized ones for lawyers, independent schools, surveyors, builders, electricians, scientists, engineers, and statisticians
USE Business and scientific
PRICE Approx. £2500
ORIC-1

Like the Sinclair ZX Spectrum and several recently launched machines, Tangerine Computers' Oric-1 is aimed at the budget end (under £200) of the micro market. Like the Sinclair machine, the Oric-1 comes in two versions - 16 k and 48 k.

The keyboard is well designed, with keys that are a cross between typewriter keys and the rubber keys of the Spectrum. Use of each key is accompanied by a "bleep", but this facility can be switched off. The angle of the keys makes for easy use.

The Oric-1's display is similar to the format used by Prestel, Ceefax, and Oracle. Eight colors, including black and white, are available, and you can display them in either the foreground or background mode. The high-resolution graphics capability is 200 lines x 240 cells. A sound synthesizer, to add effects to games, is also incorporated. It is best to output the sound through your hi-fi rather than through the internal speaker.

I/O facilities are good: the Centronics parallel printer port allows you to use printers ranging from simple to sophisticated, while all the system signals are available through the expansion port, so that you can add whatever peripherals you want. A modem, to allow connection to Prestel, is planned.

By comparison with the Spectrum, the Oric-1 is poorly provided for in terms of software, and as the machines' Basics are different, Spectrum software will not run on the Oric-1. However, existing software - mostly games - is already being translated into Oric Basic.

SHARP PC-1500

The PC-1500 has the appearance, at first sight, of an overgrown calculator with far too many keys. It can, of course, be used as a calculator, but it is a true computer, programmed in Basic. This use of Basic as a programming language gives the machine much greater versatility than any calculator, even including the programmable models.

The PC-1500 uses special chips that consume much less power than normal chips. These CMOS (Complementary Metal Oxide Silicon) chips allow the machine to run for 50 hours on its minute batteries, according to Sharp's figures.

There is a drawback, however, in that the CMOS chips do not perform as rapidly as others, so that the PC-1500 is a rather slow machine in operation.

The model is very compact and the design is good. In particular, the display is very clear and is large enough to show graphics. Six user-definable function keys are situated below the display and the whole keyboard comprises 65 keys, including a numeric pad.

A four-color plotter is available as a useful add-on. This produces a wide variety of graphics symbols and text in a range of sizes. Four very small ball-point pens are set in a barrel similar to that of a revolver. Under program control the barrel rotates to allow selection of the different colors to be made.

TECHNICAL SPECIFICATION

HARDWARE
CPU 8-bit CMOS
RAM 2.6 k expandable with plug-in modules to either 4 or 8 k
ROM 16 k, containing Basic interpreter
Display Single-line liquid crystal, 26 characters; 7 x 156 dots graphics
Keyboard 65 keys, including numeric pad and 6 programmable keys
I/O Twin cassette interfaces for program/data storage; optional four-color plotter

SOFTWARE
Language Basic interpreter in ROM

USE
Scientific, engineering, business

PRICE
Approx. £170

ORIC-1

Like the Sinclair ZX Spectrum and several recently launched machines, Tangerine Computers' Oric-1 is aimed at the budget end (under £200) of the micro market. Like the Sinclair machine, the Oric-1 comes in two versions - 16 k and 48 k.

The keyboard is well designed, with keys that are a cross between typewriter keys and the rubber keys of the Spectrum. Use of each key is accompanied by a "bleep", but this facility can be switched off. The angle of the keys makes for easy use.

The Oric-1's display is similar to the format used by Prestel, Ceefax, and Oracle. Eight colors, including black and white, are available, and you can display them in either the foreground or background mode. The high-resolution graphics capability is 200 lines x 240 cells. A sound synthesizer, to add effects to games, is also incorporated. It is best to output the sound through your hi-fi rather than through the internal speaker.

I/O facilities are good: the Centronics parallel printer port allows you to use printers ranging from simple to sophisticated, while all the system signals are available through the expansion port, so that you can add whatever peripherals you want. A modem, to allow connection to Prestel, is planned.

By comparison with the Spectrum, the Oric-1 is poorly provided for in terms of software, and as the machines' Basics are different, Spectrum software will not run on the Oric-1. However, existing software - mostly games - is already being translated into Oric Basic.

SHARP PC-1500

The PC-1500 has the appearance, at first sight, of an overgrown calculator with far too many keys. It can, of course, be used as a calculator, but it is a true computer, programmed in Basic. This use of Basic as a programming language gives the machine much greater versatility than any calculator, even including the programmable models.

The PC-1500 uses special chips that consume much less power than normal chips. These CMOS (Complementary Metal Oxide Silicon) chips allow the machine to run for 50 hours on its minute batteries, according to Sharp's figures.

There is a drawback, however, in that the CMOS chips do not perform as rapidly as others, so that the PC-1500 is a rather slow machine in operation.

The model is very compact and the design is good. In particular, the display is very clear and is large enough to show graphics. Six user-definable function keys are situated below the display and the whole keyboard comprises 65 keys, including a numeric pad.

A four-color plotter is available as a useful add-on. This produces a wide variety of graphics symbols and text in a range of sizes. Four very small ball-point pens are set in a barrel similar to that of a revolver. Under program control the barrel rotates to allow selection of the different colors to be made.

TECHNICAL SPECIFICATION

HARDWARE
CPU 8-bit CMOS
RAM 2.6 k expandable with plug-in modules to either 4 or 8 k
ROM 16 k, containing Basic interpreter
Display Single-line liquid crystal, 26 characters; 7 x 156 dots graphics
Keyboard 65 keys, including numeric pad and 6 programmable keys
I/O Twin cassette interfaces for program/data storage; optional four-color plotter

SOFTWARE
Language Basic interpreter in ROM

USE
Scientific, engineering, business

PRICE
Approx. £170
SHARP MZ-700

Sharp was among the first Japanese companies to make inroads into the UK market, enjoying moderate success with its MZ-80K and its update the MZ-80A. The company has not been strong in business machines in this market, but the MZ-700, which is available in several versions, could provide a breakthrough.

The basic computer offers eight colors, high-quality graphics and an optional four-color printer/plotter for both text and graphics is available. As with the printer/plotter, the optional cassette recorder fits into a space behind the keyboard.

The machine's S-Basic is provided on tape, as with its predecessors. Loading each time the machine is used is tedious but the tape system means that other languages, including Fortran and Pascal, can easily be employed. Machine-code programs can also be loaded from tape.

The MZ-700 range lacks the built-in monitor of the earlier Sharp machines and so is more compact. The keyboard is typewriter-style, with 69 keys, including cursor control and five programmable-function keys. The latter, when used with Shift, boost the total to ten.

The MZ-700 is the first Sharp home micro to have color graphics, the predefined graphics characters being very good. The basic format is 25 lines x 40 characters, but 50 lines x 80 characters is possible by using quarter blocks.

SIRIUS 1

This machine was the first of the new generation of 16-bit micros to arrive in the UK. Imported into this country by ACT, it is virtually the same machine as that which goes by the name of the Victor 9000 in the USA. The Sirius 1 was designed and manufactured by Chuck Peddle, the creator of the original Commodore PET.

When the Sirius first appeared, its price — similar to that of the many less powerful machines — caused a storm. For the price you might have paid for a 64 k, 8-bit system, the Sirius offered a 16-bit processor (the 8086) and 128 k of RAM, expandable to 696 k.

Other benefits of the Sirius include a good keyboard, with 95 keys including numeric pad, cursor and editing controls, an 800 x 400 graphics display, and very fast 5½ inch disks, giving 600 k per disk.

The display's high quality makes it very effective for work processing. All the programmable function keys can be set up to perform word-processing operations, and you can program certain other keys to produce at a single keystroke words and phrases that you use frequently. Both of these facilities derive from the fact that you can reconfigure the whole keyboard under software control.

TECHNICAL SPECIFICATION

<table>
<thead>
<tr>
<th>HARDWARE</th>
<th>SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Z80</td>
<td>Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System</td>
</tr>
<tr>
<td>RAM 64 k user RAM, 4k video RAM; 4k character generation</td>
<td>Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran</td>
</tr>
<tr>
<td>ROM 4k machine code monitor, 4k character expansion connector</td>
<td>Applications Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available</td>
</tr>
<tr>
<td>Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display</td>
<td>USE Business</td>
</tr>
<tr>
<td>Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys</td>
<td>PRICE Approx. £2200</td>
</tr>
</tbody>
</table>

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

I/O Cassette, TV, video, joystick ports, Sharp expansion connector

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200

TECHNICAL SPECIFICATION

CPU 8088 (16-bit)

RAM 128 k, expandable to 896 k

Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display

Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200

TECHNICAL SPECIFICATION

CPU 8088 (16-bit)

RAM 128 k, expandable to 896 k

Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display

Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200

TECHNICAL SPECIFICATION

CPU 8088 (16-bit)

RAM 128 k, expandable to 896 k

Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display

Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200

TECHNICAL SPECIFICATION

CPU 8088 (16-bit)

RAM 128 k, expandable to 896 k

Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display

Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200

TECHNICAL SPECIFICATION

CPU 8088 (16-bit)

RAM 128 k, expandable to 896 k

Display 25 lines x 80 characters, switchable to 50 lines x 128 characters; 800 x 400 dots graphics display

Keyboard 95 keys, including numeric pad, cursor and editing controls, brightness and contrast controls, seven programmable keys

Disks: Twin 5½ inch in single-sided, capacity 600 k per disk; optional double-sided drives, capacity 1200 k per disk; optional 5 and 10 megabyte hard disks

I/O Twin RS232 ports, Centronics parallel printer port, IEEE 488 port; internal expansion bus for plug-in cards

SOFTWARE

Operating system CP/M-86 and MS-DOS; optional CP/M-80 (with plug-in Z80 card), UCSD p-System

Languages Basic (interpreted and compiled), Pascal, C, Forth, Fortran

Applications: Pulser range of business software from ACT; word processing, spreadsheet, communications, graphics; business packages also available

USE: Business

PRICE: Approx. £2200
SORD/CGL M5

One of the first true home computers from Japan, the M5, which is available direct from Sord or through the UK distributors Computer Games Limited (CGL), is deceptively capable beneath its downmarket exterior. The 3 k RAM seems, at first inspection, poor value for money, but further examination reveals that a separate 16 k RAM block is used for display. This means that the sophisticated graphics capabilities of the M5 do not consume precious program space.

The keyboard has rubber calculator-style keys which register a confirmatory click on the monitor. The 55-key layout includes keys that give each of the main Basic commands at a single stroke, although the keyword function can be turned off and the commands typed in full if preferred.

An unusual feature of the M5 is that it does not have Basic contained internally in ROM. Integer-only Basic-I is provided by a plug-in ROM cartridge. Similarly, to use the machine's impressive graphics or floating-point Basics, you have to insert Basic-G or Basic-F cartridges.

User-definable groups of graphics characters known as "sprites" are an interesting feature of the Basic-G ROM. These function in 31 planes, so that complicated games can be invented. Used in conjunction with the machine's powerful sound synthesizer, this capability makes the M5 a very good games machine.

SORD M23

The M23 possesses surprising power for its size: its RAM capacity is 128 k. Although it cannot be considered a true portable, the machine is very compact.

Sord has taken an unconventional step in producing its own software. At the heart of its range is PIPS (Pan Information Processing System), which is a combination of operating system, general-purpose business package, and spreadsheet.

TECHNICAL SPECIFICATION

HARDWARE
CPU Z80
RAM 3k user RAM (expandable to 7k with Basic-G cartridge), 16k video RAM
ROM 8k, expandable to 16k
Display With optional Basic-G 24 lines x 32 or 40 characters, 40 lines x 64 characters pixel graphics, 32 sprites, 16 colors
Keyboard 55 keys including cursor control
I/O Cassette, Centronics printer, games controllers, TV and monitor interfaces

SOFTWARE
Operating system Contained in ROM cartridges
Languages Basic-I (Integer Basic), optional Basic-G and Basic-F, all in ROM cartridges
Applications Games

USE
Home

PRICE
Approx. £150

I/O Twin RS232 serial ports, Centronics parallel printer port; optional IEEE-488 parallel port

SOFTWARE
Operating system Sord's own Languages Optional Basic, Fortran, Pascal, assembler
Applications PIPS business package, word processing, data entry (all available as plug-in ROM cartridges)

USE
Business

PRICE
Approx. £1950

TECHNICAL SPECIFICATION

HARDWARE
CPU Z80
RAM 128 k
ROM 4k
Display 24 lines x 40 or 80 characters; optional liquid crystal display giving 8 lines x 80 characters; optional graphics board giving 640 x 200 resolution; optional color display
Keyboard 92 keys, including numeric keypad, cursor control, and 7 programmable keys
Disks Optional twin 5.25 in floppy, capacity 330 k per disk; optional 8 in disks also available
TEXAS INSTRUMENTS
CC-40

TI's CC-40 or, to give it its full title, Compact Computer 40 is a hand-held, battery-operated machine intended for the professional user.

Apart from its portability, the greatest attraction of the model is the plug-in applications software produced by TI. At the moment, eight solid-state plug-in cartridges are available. Fourteen additional software packages are available on Watertape digital tape which is intended to be used with a special drive.

The TI software is currently of interest to business and technical users only, but the range of applications is expected to expand, with TI encapsulating other manufacturers' software in ROM.

The CC-40, like two other popular portables, the Epson HX-20 and Radio Shack's TRS-80 Model 100, has a built-in liquid crystal display. It is of a reasonable size for the professional user.

When you use a Hex-bus peripheral interface with this machine it is possible to take advantage of the inexpensive peripherals which are produced for use with Texas Instruments' 99/2.

TEXAS INSTRUMENTS
99/4A

Texas Instruments is among the world's largest manufacturers of electronic components. The company claims to have invented the component at the heart of the micro-computer revolution: the silicon chip. That TI should add home computers to its product list — calculators, educational aids, and scientific instruments had already established the name — was inevitable.

The TI 99/4A is a well-designed and versatile machine. It is the only home computer with a 16-bit microprocessor in the CPU. However, since this chip is of TI's creation it is not compatible with software from other manufacturers. TI has tackled the problem by producing its own extensive range of software, including the usual range of games and home packages. The company also produces a variety of hardware add-ons, among them a speech synthesizer. (Speech synthesis is a field in which TI is heavily involved, with a huge research department dedicated to its advancement.)

The 99/4A comes with Basic as standard, although an optional module is available that extends the Basic interpreter's powers substantially. This adds a wide range of additional commands to the machine's facilities, including powerful graphics capabilities.

TECHNICAL SPECIFICATION
HARDWARE
CPU 9900 (TI's own)
RAM 16 k, expandable to 48 k
ROM 34 k
Display 31 characters dot matrix (5 x 8); 18 indicators, of which 6 are programmable; adjustable contrast
Keyboard 65 keys, including numeric keypad
I/O Hex-bus peripheral interface required for various I/O functions; solid-state plug-in cartridges
SOFTWARE
Operating system TI's own
Languages Extended TI Basic
USE
Personal portable computing
PRICE
Approx. £170

TECHNICAL SPECIFICATION
I/O card giving twin RS232 and single Centronics printer parts
SOFTWARE
Operating system TI's own
Languages TI Basic in ROM; optional extended Basic, Pascal, Logo, assembler
Applications Good range of games and home software, plus some scientific/engineering and program development packages
USE
Home
PRICE
Approx. £100
A

Access The process of retrieving data from a disk file or memory location.

Acoustic coupler A device used in conjunction with a telephone handset that allows computers to communicate. The computer usually connects to the coupler through an RS232 interface. The coupler converts digital signals to audio tones and vice versa.

Ada A high-level programming language used mostly by the US Department of Defense but likely to gain wider acceptance.

ADC Analog-to-digital conversion.

Address The number (usually hexadecimal) of a specific memory location.

AI See Artificial intelligence.

Algori Algorithmic Oriented Language, a high-level programming language used mainly in scientific work.

Algorithm A description of the method used to solve a particular problem. Rather than write a solution in one specific programming language, it is considered more useful to produce it as an algorithm, which programmers can then convert into code using whichever language is most suitable.

Alphanumeric A term used to describe all the letters of the alphabet, numbers and, sometimes, punctuation marks as well.

ALU Arithmetic and Logic Unit. The part of the CPU that carries out all the computing functions.

AND A software or hardware logic function that produces a logic “1” if both inputs are “1”, otherwise a “0” result.

ANSI Stands for American National Standards Institute. This organization has been trying to standardize the Basic language. Although its draft standard has been available for some time, it has not been generally accepted by manufacturers.

APL Stands for A Programming Language. It is a powerful general-purpose language developed by IBM and used mostly on mainframes and minicomputers. Instead of using English-like commands, APL relies heavily on special symbols that ordinary keyboards, monitors, and printers cannot handle.

ASCII Acronym for American Standard Code for Information Interchange. This is a standard binary code for representing letters, numbers, punctuation marks, and other symbols, plus a few extra codes for control purposes. Computers use ASCII to store information in memory and on floppy disk, and to send it to other computers and to peripherals.

Assembly software This is a piece of software that performs some real-world job, as opposed to systems software, which is concerned with internal computer functions. A word processor or an accounting system are examples of applications software.

Architecture A term used to describe the internal structure of a computer's hardware as it is seen by the software.

Artificial intelligence A branch of computer science concerned with the development of highly advanced software. A practical development of AI is the expert system, which uses a defined set of rules to access a “bank” of information in order to suggest solutions to problems.

Auto dialer A feature provided on some modems that will automatically dial a specific number, either one preprogrammed into it or one sent in ASCII form by the computer.
is a feature of a multi-tasking operating system.

Backing up The process of making a security copy of everything you keep on disk. Reliable though most modern micros are, they are still prone to failure from time to time. Probably the weakest part of the system is the floppy disk: disk drive mechanisms wear out and develop faults and the disks themselves, unless very carefully treated, can easily be damaged. You should, therefore, as a matter of routine, make a back-up copy and keep this in a separate place. How often you make back-ups of your data depends on how much work you can afford to lose should something go wrong.

Basic The most widely used programming language. Basic was originally designed as an easy-to-learn language that would allow beginners to obtain results with the minimum of effort. The name itself stands for Beginner’s All-purpose Symbolic Instruction Code. A Basic interpreter is available on virtually every make of microcomputer and the language can be used for a wide range of applications, although it is rather slow for some jobs unless a Basic compiler is used. The major problem with Basic is that its features vary (sometimes considerably) from computer to computer. This can make it difficult to use a program written for one machine on another, different machine.

Batch processing A system of data processing in which small pieces of information are recorded on magnetic tape until a sufficiently large batch exists to be loaded into the computer and processed together. This is a more efficient method of utilizing the time of large computers. Batch processing is not used to any significant extent with microcomputers.

Baudot code A simple alphanumeric code used in telex machines and teleprinters and comprised of five bits per character. It is also known as the Murray code in some countries.

Baud rate The speed at which information is transmitted through a serial I/O interface. Generally, the baud rate divided by ten gives the number of characters per second. 300 baud, for example, is approximately 30 characters a second.

BCD See Binary coded decimal.

Benchmark One or more programs designed to test certain characteristics of a computer or CPU, usually resulting in a series of timings so that machines can be compared and some measure of their power or efficiency obtained.

Bidirectional printing Most modern dot matrix and daisywheel printers are designed so that a line of text can be printed out when the print head is traveling in either direction. Having printed a line moving from left to right, the machine moves to the next line and prints it from right to left. This saves time as there is no need to wait while the print head travels back to the start of the line.

Binary A method of representing numbers using 2 as a base instead of the more familiar 10 (decimal) system. This is essential for computers as information can be represented either by the presence or absence of an electrical current.

Binary coded decimal A method of storing numbers within a computer. Each byte is regarded as two four-bit groups or nibbles and, in turn, each group is used to store one decimal digit in binary. Although this takes more memory space and makes arithmetic operations slower, it gives very accurate results.

Bit Contraction of binary digit. A bit is the basic unit of computer information. Usually eight bits are required to form the codes representing any character, number, punctuation mark or symbol found on a computer keyboard. Eight bits are called a byte.

Bit-slice processor A CPU made of several chips, each of which is, in effect, a vertical slice through an ALU. Usually bit-slice processor chips are four bits wide and can be linked in parallel to form a processor 8, 16, 32, or 64 bits wide. The result is a very fast processor, but bit-slice machines are much more difficult and expensive to build than those based on single-chip CPUs. Bit-slice technology is used in minicomputers and mainframes.

Boolean algebra A branch of mathematics dealing with logical statements and propositions that form the basis of the logic used in computer software and hardware.

Buffer An area of memory reserved for holding data temporarily. Most computers have, for example, a keyboard buffer — as you type in a command, each letter is simply placed in the buffer. The computer then checks each character to see if you have typed the RETURN key to indicate that the command is complete. Only when you type RETURN does the computer fetch the command from the buffer and take whatever action is appropriate. Certain hardware circuits are also called buffers — their most common use is to boost signals that are too weak to drive
the number of chips to which they are being distributed, or to form the interface between two different parts of systems or complete systems.

**Bug** A malfunction of some kind, either in the computer’s hardware or software. Traditionally, the term is attributed to computing pioneer Grace Hopper who, on investigating a fault on an early computer, traced its cause to a moth trapped between two relay contacts.

**Byte** A grouping of eight bits. Because, generally speaking, there is not a lot you can do with a single bit, they are grouped into units capable of representing more useful quantities. The basic unit in microcomputing is the byte. Many low-cost micros use 8-bit processors, which handle information and instructions a byte at a time; more powerful micros have 16-bit processors which work in units of two bytes at a time.

**C** A programming language that combines the versatility of assembler language with the speed of writing and ease of use of a high-level language such as Pascal. Originally developed by Bell Laboratories in the USA for use with the Unix operating system, C is now gaining in popularity in the micro world and C compilers are available for many micros.

**CAL** Stands for Computer-Aided Learning.

**Cambridge ring** A method of connecting up computers and peripherals to form a local area network.

**Centronics interface** Centronics is a company that manufactured dot matrix printers. As the micro industry began, Centronics had the market for small, low-cost printers virtually to itself and thus the parallel I/O method used on Centronics printers became (and remains) the industry standard.

**Character generator** A circuit that turns ASCII codes into the dot matrix matrix patterns required to generate the appropriate character on a monitor screen.

**Chip** Colloquial name for an integrated circuit, in which large numbers of transistors, diodes, and other components are built up on a single piece of silicon.

**Clock** An electronic circuit that generates a series of precise pulses, used by the rest of the computer to synchronize operations. Not to be confused with a clock/calendar chip, which keeps track of the time and date.

**CMOS** Stands for Complementary Metal Oxide Silicon, a special form of chip construction that results in a component using very little power but operating rather slower than the other types of chip technology in use. CMOS is normally used in calculators, watches, and pocket computers.

**Cobol** Stands for Common Business Orientated Language. A high-level programming language designed for and used in commercial or business applications. It was originally designed for mainframes but is now appearing on micros, too, to allow the micro user to take advantage of the massive amount of Cobol software available.

**Compact disk** A 4.7 in plastic disk on which digitized sound has been recorded and which can be read with a special laser-based playing unit. Although in its infancy, the technology has much potential as a method for distributing computer software.

**Compatibility** The ability to use a peripheral or piece of software on several different machines. In hardware terms, this means a common, standardized interface such as the Centronics interface. For software, it is vital: any commercially produced package needs to be able to run on as many different computers as possible. The CP/M family of operating systems achieves this to a high degree, while high-level languages are another method of transferring software. With some languages, such as Basic, compatibility hardly exists.

**Compiler** A piece of software that converts a program written in a high-level language into the binary code that computers can understand. A program developed with a compiler will nearly always work more quickly than one that requires an interpreter, but will not necessarily be either as quick or as short as an assembler language program.

**CP/M** The most widely available microcomputer operating system. Developed in the USA, CP/M gained rapid acceptance as a sensible and reasonably easy way to ensure that programs written on one make of computer would work properly on another. It provides a standard “connection” between the software and hardware. Several versions of CP/M are now available: CP/M-60 for 8-bit machines, CP/M-86 for 16-bit micros, MP/M (a multi-user version), and CP/NET, a network version. CP/M can be justifiedly criticized as rather crude and not very user friendly, but it is easy to use once you are used to it.

**CPU** Stands for Central Processing Unit, the “brains” of a computer where all the actual computing takes place. In a microcomputer, this is contained on a single component called a microprocessor, while on larger computers it comprises anything from several printed circuit boards to a closet-sized cabinet crammed with boards.

**CRT** Stands for Cathode Ray Tube and, strictly speaking, refers to the television-like display found on modern computers. It is, however, frequently used to refer to the display plus keyboard which, in some machines, are housed together in a separate unit.

**Cursor** The small symbol, line, or character that indicates the point on the screen at which the next character to typed will appear.

**Daisywheel printer** A high-quality printer which,
Glossary

instead of using type bars as found in conventional typewriters, has a small plastic or metal wheel containing the characters on spokes. The wheel is rotated by an electric motor until the appropriate character is uppermost and then an electromagnetic hammer knocks it against an inked ribbon. A low-cost daisywheel printer prints about 8 characters per second; better-quality machines can print at between 45 and 60 characters per second.

Data Anything stored in or produced by a computer which is not a program. Data processing is simply the carrying out of some sort of operation (sorting, collating, calculating, for example) on data in a computer.

Database A store of data or information arranged in an ordered way to allow it to be retrieved easily. The storage of and rapid searching and sorting through massive amounts of information in a database is one of the applications for which computers are ideally suited.

Debugging The process of searching for and eliminating the faults on a piece of software or hardware. Sometimes, in the micro world, this is carried out by the manufacturer; often, it is the customer that ends up doing it.

Dial-up system A computer system that the public can access. Very popular in the USA and becoming more so in Europe, the service is sometimes free, sometimes for a charge. Customers use their own computers and a modem or acoustic coupler. Facilities provided vary from simple “bulletin board” message handling to sophisticated databases on many different subjects.

Digitizing tablet An alternative name for a graphics tablet.

Directory An area on a disk reserved for information describing the files—programs and data—that are held on the disk and where each is on the disk. The former information is usually available to the user in response to a command such as CATALOG or DIR, while the latter is used by the computer itself to find information stored on the disk.

Disk drive The mechanism that holds and rotates a disk, either floppy or hard, on which computer programs and data are stored for later retrieval.

Documentation The manuals that accompany a new computer or piece of software, telling you how to use it.

Dot matrix printer A low-cost, faster alternative to the daisywheel printer. Characters are printed by firing a series of needles at the ribbon and paper to form the appropriate pattern of dots. The quality of the resulting print is not as good as that produced by a daisywheel printer, but some dot matrix printers are capable of high-quality output and can print up to speeds of several hundred characters per second.

Down Slang term indicating that a computer is not working because of some sort of fault.

E

EAPROM Stands for Erasable Programmable ROM. This is a type of PROM whose contents can be erased by exposure to strong ultraviolet light. It can then be reprogrammed using a special programming device.

Ethernet A type of network system that allows audio and video information to be carried as well as computer data. Although currently rather expensive, the Ethernet hardware is dropping in price and the system is expected to become widely used.

Expert system Software that uses the principles of artificial intelligence to solve problems by questioning the user for information and arriving at a possible answer by matching this information with data held in its “knowledge bank.”

Field A category of information held inside the computer. Information in, say, a database must be organized in a particular structure to allow the computer to sort through it. Each record is, in turn, divided into fields. An electronic address book, for example, would have one entry per record and each record might be subdivided into several fields— one for the person’s name, one or more for the address, and one for the telephone number.

File Generally used to describe a collection of data...
stored on a disk or cassette. Some files will be programs that the computer can use, while others will be information such as databases or text produced by word processing.

**Firmware** One or more programs permanently fixed in a memory chip and thus forming part of the computer system. Generally, firmware is used to hold monitors and, in home computers, games and other software in interchangeable ROM packs.

**Flip-flop** A logic circuit that can be made to switch to and hold either a binary 1 or 0; RAM cells are, basically, flip-flops.

**Floppy disk** A storage medium for data and programs. Because the contents of a computer’s memory vanish when the power is switched off, it is necessary to have a permanent way of storing information. Floppy disks are the most popular way of achieving this on business computers, and increasingly so on home machines, too. The disk consists of a thin plastic base material coated with a magnetic oxide similar to that used in audio cassettes, but of far higher quality. The disk is housed permanently in a protective envelope with holes in it to allow a read/write head in the disk drive to come into contact with the disk’s surface. As the disk is rotated in the drive, the head can be moved across its surface under computer control, to allow the computer to access information stored anywhere on its surface.

**Flowchart** The visual representation, with special symbols, of the logic of a program or algorithm.

**Formatting** The recording of special information on a disk by a computer to identify the tracks and sectors on the disk surface. Most makes of computer seem to have their own particular format, making it difficult to use a disk formatted for one make of machine in a computer of another make.

**Fortran** A high-level programming language, similar in many ways to Basic. As its name (FORMula TRANslator) suggests, its main use is in scientific and engineering applications. Like Cobol, it was developed and remains, mostly a mainframe computer language, although Fortran compilers are available on micros, allowing access to the wide range of software written in this language.

**G-H**

**Gate** A logic circuit in a computer.

**Graphics** The appearance of pictures or diagrams on the screen as opposed to letters and numbers. Most home computers and many business machines now offer graphics, either as standard or as an optional extra. As well as producing increasingly detailed displays for games, graphics are gaining widespread use in business computing as a way of presenting information in a more easily assimilable form. The use of color adds an extra information channel to graphics displays, and you can expect to see more business machines offering color graphics as standard.

**Graphics tablet** A peripheral input device. You draw on a special surface with a stylus and as you do so your drawing appears on the computer’s screen and can be saved on disk or printed out on a graphics printer or plotter.

**Handshaking** An interchange of special signals between a computer and a peripheral such as a printer. The printer, for example, might send out a signal to indicate that it is ready to receive characters from the computer; the computer might then send a signal signifying that a character is coming. When the printer receives the character, it prints it and sends out the “ready” signal again, and so on.

**Hard copy** The output (perhaps text or a program listing) printed out by the computer on to paper.

**Hard disk** A magnetic disk for holding programs and data in much the same way as a floppy disk. Although hard disks are expensive to buy, they hold far more than a floppy disk and work more quickly. The most popular type for microcomputers is the Winchester disk.

**Hardware** The physical components of the computer system.

**Hexadecimal** A numbering system using the base of 16, as opposed to our normal decimal system that uses base 10 and binary that uses base 2. Hexadecimal is mainly used as a convenient way to represent binary quantities. As its base is 16, it needs an extra five symbols to represent the decimal quantities 10-15, and this is done with the letters A-F.

**High-level language** A programming language in which commands are as near as possible to English words and in which details of the computer’s hardware need not concern the programmer.

**IEEE-488** An interface standard mainly used to connect laboratory instruments and other scientific equipment to computers, either for control purposes or to allow the computer to collect data.

**Information technology** The merging of computing and communications and high-speed digital communications links carrying data as well as digitized sound and video.

**Ink jet printer** A printer in which tiny drops of ink are fired at the paper to build up characters dot by dot. The advantages are speed, quietness, and an output that looks much superior to that of a dot matrix printer but slightly inferior to that of a daisy-wheel printer.

**Interactive** Generally, this term refers to the facility for the user to input information on demand from the computer and for the computer to respond
Glossary

Interface The connection between two systems. As far as the computer is concerned, this means a variety of electrical, physical, and software standards which enable various types of equipment to be connected together. Thus, a computer with a Centronics interface can be used with any printer equipped with a Centronics interface. Other commonly found interfaces are the RS232 and the IEEE-488.

Interpreter A piece of software that translates a program written in a high-level programming language into the binary instructions the computer understands. Unlike a compiler, though, an interpreter does this while the program is being used. An interpreted program is therefore much slower than a compiled one or one written in assembler language, but is much more easily altered and corrected during the development stage. Basic is one of the most popular interpreted languages and Basic interpreters are found on nearly every microcomputer.

I/O Stands for input/output. I/O ports are provided on most computers to allow them to communicate with other devices. Generally, these are built to one or more of the popular interface standards.

Joystick An add-on device popular with home computer hobbyists. It allows you to, say, move an object about on the screen by moving a lever or knob instead of typing codes or instructions at the keyboard. Also widely used (in a more sophisticated form) for computer-aided design.

Kbyte Short for kilobyte - 1024 bytes of data or program. Also commonly abbreviated to k.

Keyboard The principal way to communicate with the computer. A computer keyboard is much the same as a typewriter keyboard, with the addition of a few extra keys. These extra keys, sometimes, include ones that can be set up by the user to perform whole strings of instructions at a single keystroke. These are called “programmable function keys”.

Keyword A specially marked word in a database record or file that can be searched for by the computer or which the computer can use to sort the information into some order.

Kilobaud A measure of the speed at which serial data is transmitted, meaning 1000 baud.

LED Stands for Light Emitting Diode, usually found as an indicator lamp on a computer, especially on disk drives to show when a drive is in use.

Light pen A special stylus that can be interfaced with a computer and pressed against the screen to select an item from a menu or to “draw” on the screen.

Lisp A high-level language used in artificial intelligence research. Stands for LISt Processing.

Local area network A network system that links computers within the same room, building, or site, as opposed to networks that use telephone cables or satellites.

Loop A programming term for a section of a program that is executed repeatedly. If the program is properly written, the number of iterations will be controlled in some way. A program that contains an “infinite loop” is one that will never finish executing until the machine is physically switched off.

Machine code A computer language consisting of a series of binary codes, usually produced by an assembler or compiler. As its name implies, machine code is the language understood directly by the computer.

Machine language See Machine code.

Mainframe A large, powerful computer capable of performing many tasks simultaneously and of servicing many users simultaneously.

Megabyte A million bytes.

Memory One of the major components of a computer system. Memory is where the computer holds its programs and the data those programs require. Users are mostly concerned with the high-speed random access memory (RAM), which forms part of the main system. This, however, loses its contents when the power is switched off; so more permanent types of memory are also provided—ROM. Any information in ROM is stored permanently.

Memory map A diagram of a computer’s memory system, showing which parts are used for specific purposes. Some areas, for example, might be reserved for the operating system while other parts will be reserved for the user’s programs.

Menu A main selection of possible tasks presented to the user on the screen while a program is running. The user selects from the menu, usually, by typing the number corresponding to the action or operation desired.

Microchip An integrated circuit. Consists of a piece of silicon on which an electronic circuit has been constructed. Also known colloquially as a silicon chip. A microprocessor is one type of silicon chip.

Microcode The control instructions built into the CPU which interpret the binary machine code instructions into signals for controlling various parts of the CPU.

Microcomputer A small computer based around a microprocessor chip. The term microcomputer is synonymous with “personal computer”, meaning a computer whose resources are available to one user
only. Personal computers range from the very basic to systems of microcomputers able to service many users.

Microelectronics The branch of electronics concerned with making and using integrated circuits.

Micro floppy A new type of floppy disk measuring either 3 or 3½ in across instead of the more normal 5¼ or 6 in.

Microprocessor A computer's central processing unit (CPU) squeezed on to a single component instead of being divided into several components or even several boards of components. A microprocessor forms the basis of a microcomputer system.

Minicomputer A computer that is not big enough to be called a mainframe but is too big and powerful to be a microcomputer. Minicomputers are becoming more and more powerful so that the distinction between a large minicomputer and a small mainframe is becoming increasingly blurred.

Mnemonic In computer terms, a way of expressing an assembler language instruction so that it is easy to remember and use.

Modem A device that allows computers to communicate through the normal telephone system. Digital signals are converted into audio tones suitable for transmission along telephone lines to another modem, which reconverts them into digital signals that can be understood by the computer at the receiving end. Modems are wired directly into the phone system, unlike acoustic couplers. Usually, however, special permission is required before a modem can be installed.

Monitor A word with two widely different meanings. In one sense, it is a type of cathode ray tube (CRT) or visual display unit (VDU) on which the computer displays text and graphics. In the other sense, it is a piece of software in machine code that performs simple instructions such as accepting characters typed in at the keyboard and displaying them on the screen. Increasingly, monitors are placed in firmware and do nothing but load the operating system from disk and then hand over control of the system to that.

Motherboard The board into which additional boards containing extra memory or I/O interfaces can be plugged. The S100 system is one in which the motherboard contains 100 connections for each socket.

Mouse A small box, mounted on rollers, and connected to the computer by a cable. As the user moves the mouse around on a flat surface such as a desk top, a pointer on the screen moves accordingly. It is possible to select an operation by moving the pointer (via the mouse) to a picture or text line describing the operation and then pressing a button on the mouse to inform the computer that this is the selected operation.

MS-DOS A 16-bit operating system marketed by Microsoft Inc., for use on microcomputers with the 8086 or 8088 processors. It is the operating system chosen by IBM for its Personal Computer and its main rival is the 16-bit versions of CP/M—CP/M-86.

MTBF Technical abbreviation of "mean time between failures". It is a measure of the reliability of a piece of equipment and it is usually expressed in hours.

Multi-tasking The process of carrying out more than one operation simultaneously.

Multi-user A system in which a sophisticated operating system allows several users to share the same computer. In order to perform this task, the CPU spends a little time working for one user, and then some time for the next user, and so on. To the users, the result should appear as if they each had a CPU of their own. It is a technique inherited from the world of big computers but really does not work too well on microcomputers, where the microprocessor often cannot cope with the load.

NAND A logic operation that works in precisely the opposite way to the logic operation AND.

Network A method of connecting several computers together to allow them to exchange data and to share costly peripherals such as printers and hard disks. Currently two types of networks are in contention for supremacy in the micro world—Ethernet and the Cambridge ring. While neither has yet gained absolute recognition, other manufacturers keep introducing new systems, causing a considerable degree of confusion within the industry.

NTSC The color television standard used in America and some other countries. It stands for National Television System Committee (although television engineers know it as "Never Twice the Same Color").

Octal A numbering system to base eight.

OEM Stands for Original Equipment Manufacturer. Something of a misnomer really, as OEM is usually a company that buys ready built computer modules—sometimes entire computers—from other manufacturers, puts its own labels on them, and markets them under its own name.

Off-line A piece of equipment that is not ready to receive data from or transmit data to a computer, either because it is switched off or because it is somehow otherwise disconnected.

On-line The opposite of off-line.

Opcode An assembler language instruction.

Operating system A program that contains a set of small standardized programs that a programmer can use to carry out various operations such as storing data on disk and retrieving it. The most popular operating system is CP/M. A computer can do nothing without a program—a list of instructions to tell it what to do. This includes the basic things such as ac-
Glossary

ceping a character typed in at the keyboard and displaying it on the screen. Some computers contain a monitor to perform these functions but in a more complex machine, especially one containing disk drives, a correspondingly more complex program is required.

Optical disk A clear, plastic disk containing an aluminum inner core on which binary data has been engraved and which is read by a low-powered laser beam. This is the technology used in compact and video disks.

Optical fiber A thread of highly transparent glass which is pulsed very rapidly to carry a stream of binary signals. As well as carrying a high volume of data, optical fibers are immune to the electrical interference that can plague conventional cables. It is rapidly becoming the standard way for computers to communicate.

P-Q

PAL Stands for Phase Alternation Line and is the color television system used in most European and many other countries.

Parallel I/O A system of communicating between a computer and peripheral by sending data a byte at a time along eight wires, with one or more wires for the ground and one for handshaking signals.

Parallel processing Carrying out two or more tasks simultaneously by having more than one CPU built into the system and devoting one to each task.

Pascal A high-level programming language widely used in microcomputers but by no means as popular as Basic. It is a compiled language, which makes it slightly more awkward to use but, unlike Basic, it stays more or less the same regardless of which computer it is used on. This makes the process of transferring programs from one computer to another very easy.

PCB Stands for Printed Circuit Board, and it is the base on which most computer circuits are constructed. It consists of a plastic or plexiglass board coated in copper, which, in turn, is coated with a photosensitive material. The connecting lines between the components are photographically projected on to the coating, which hardens wherever light falls on it. Dipping the board in acid then removes all the copper covered by unhardened coating, leaving the pattern of connecting lines.

PEEK An instruction in Basic and some other programming languages, which allows the user to obtain the contents of a specific memory location.

Peripherals Extra pieces of equipment that do not form part of the main computer system. These items include devices such as printers, modems, and acoustic couplers.

Plotter Peripheral output device that allows the production of graphics such as graphs, charts, or technical drawings on paper.

POKE The opposite of PEEK. POKE allows you to place a value directly into a specific memory location.

Port In computer terms, a circuit that forms an interface between the system and a peripheral to allow I/O operations to take place.

Printed circuit board See PCB.

Printer A device for producing hard copy from a computer. Printers come in two general types—dot matrix and daisywheel.

Printwheel The element in a daisywheel printer that contains the raised type for producing characters.

Processor The central part of a computer that performs most of the actual computing functions. Also known as the CPU.

Program A list of instructions that tell the computer what to do and in what order to do it. There are two main types of programs—systems programs (such as monitor and operating systems) which are concerned with organizing the internal workings of the computer system, and applications programs that perform some useful task for the user (such as accounts or word processing). Systems programs are usually written in a low-level programming language such as assembler language, while applications programs, which may need to be altered more frequently, are usually written in a more user-friendly high-level language.

Programming language Computers can only understand programs in binary notation. Unfortunately, it is particularly difficult to understand, let alone think in, binary, so programs are instead written in a near-English (usually) notation. This is then translated into binary patterns by a special program called a compiler or an interpreter. Programming languages such as assembler, because they are very close to the machine code of the computer, are referred to as low-level languages, while others such as Basic, which are fairly close to English, are referred to as high-level languages.

PROM A type of ROM chip, the contents of which can be defined by the user but which cannot be erased and reprogrammed in the same way as an EPROM.

p-System A microcomputer operating system that is rapidly gaining in popularity although it is doubtful whether it will ever gain as widespread acceptance as CP/M. One of its principal advantages is that programs written for it will work on a very wide variety of machines, whereas CP/M programs will only work on machines that contain certain types of CPUs.

QWERTY The name given to the normal typewriter layout for keyboards, named after the first five letters on the top row of the alpha section. Although designed originally to slow down typing (because early mechanical typewriters simply could not keep up with typists), the QWERTY layout has become so standardized that it is unlikely to be replaced for a long time.
**Glossary**

**RAM** Stands for Random Access Memory. Rather a misnomer as there is nothing random about the way it is used. This is where the computer keeps the programs it is currently using and where any information required for or produced by the program is stored until the computer either needs it (in the case of input) or is instructed to move it elsewhere (in the case of output), to a printer, for example. The advantage of having as much RAM as possible is that larger programs and more data can be kept there, where it is more quickly accessible than it would be if it was kept on disk. However, RAM loses its contents when the power is switched off, so disks or cassette tapes are still necessary.

**Read/write head** The component in a disk drive that turns electrical pulses into magnetic ones to record data on to a disk. It also performs the exact opposite function to read data from a disk.

**Register** A memory location within a computer's CPU in which data can be held while the computer is operating on it. Keeping frequently used data within the CPU's registers allows the CPU to work much faster than if it had to move the data continually to and from locations in the system's main memory.

**Resolution** A measure of the quality of a graphics device (screen, printer, or plotter), usually in the form of the number of horizontal and vertical dots making up the image. The finer the resolution (the higher the number of dots), the better the quality of the resulting image.

**ROM** A type of memory component in which the contents are permanently fixed, usually during the manufacturing stage. Some types of ROM, though, notably PROMs and EPROMs, can be programmed after they have been manufactured. The main use for ROMs is to hold programs such monitors in firmware.

**RS232** A standard method of interfacing a peripheral to a computer, in which data is sent serially, one bit at a time. Also known as V24. The speed at which the data is transferred is known as the baud rate and — usually — this must be preset at each end of the link before successful transfer can take place.

**S100** A system — not now much used — of constructing a microcomputer around a motherboard containing interconnected sockets, each with 100 connections. Each element of the system — CPU, memory, I/O interfaces, for example — resides on a separate board. Although rather more expensive than the now normal method of putting everything on one PCB, the system does allow great flexibility and expansion potential.

**Screen** The normal method by which the computer communicates with its user. Most home computers use a domestic television for this purpose, while business machines usually use a monitor, capable of a higher quality display than a television.

**Scroll** The movement of a display on a screen, usually upward or downward. Scrolling is most frequently encountered in word processing, to allow the user to read through a document a screenful at a time.

**Sector** The surface of a floppy disk is divided electronically into a number of concentric tracks, each of which is in turn divided into a number of sectors. Each sector usually holds either 128, 256, or 512 bytes. This system allows a computer to find any piece of information on a disk by moving the read/write head to the appropriate track and reading the sector containing the information.

**Serial I/O** A method for transmitting information to and from a computer one bit at a time, unlike parallel I/O in which information is transmitted by a byte at a time. The main serial I/O standard is the RS232 interface.

**Smalltalk** A very advanced operating system developed by the Xerox Corporation and designed to make computer use as easy as possible for the layperson. It presents possible choices of operation in the form of pictures or "icons" on the screen. The user selects one by moving a pointer with the aid of a mouse to the appropriate position and then presses a button on top of the mouse to inform the computer of the choice.

**Software** An alternative name for computer programs.

**Spooler** A peripheral device or piece of software that allows a computer to produce hard copy on a printer while doing something else.

**Speech recognition** The ability of a computer to recognize spoken commands. Just as computers can be made to synthesize speech (of sorts), so they can accept instructions and data in spoken form through a speech recognition peripheral. Like speech synthesis, though, it is a complex process which is still in its infancy and has not, as yet, produced a computer capable of understanding more than a few words.

**String** A group of characters representing a word or sentence, held in a computer's memory. In Basic, a string variable — one to which alphanumeric text can be stored — is denoted by a $ sign after the variable's name.

**Stringy floppy** A small cartridge containing a continuous tape loop, which

**Source code** The name for a computer program written with a text editor or word processor, before it has been compiled.
provides the computer with a random access capability similar to, but still slower than, a floppy disk at a price comparable to that of a cassette.

Synchronous communications A method of exchanging data at very high speeds between computers, involving careful timing and special control codes. Used mostly with mainframe and minicomputers.

Syntax error A mistyped command or instruction (PRIMT instead of PRINT, for example) resulting in the computer failing to understand what you mean. A syntax error message will come up on the screen when this happens (the exact form varies from computer to computer) and you will see it quite often when you are learning to program.

Systems analysis The process of deciding whether, say, an office could benefit from computerization and, if so, in what ways.

Thermal printer A type of dot-matrix printer that uses special, heat-sensitive paper and forms characters by applying heated dots to the paper's surface. Although they are fairly fast and very quiet, thermal printers have never become popular because of the cost of the paper.

Touch-sensitive screen A special type of screen covered in a fine wire mesh that is connected to the computer through an interface. Rather than select an operation by typing a number or by moving a pointer with a mouse, the user simply touches the part of the screen where the description of the desired operation is displayed. The computer senses where the screen has been touched and, because it knows what is displayed at that location, initiates the selected choice.

Truth table A chart, AND for example, showing the results from all possible input combinations of a logic operation.

UART Stands for Universal Asynchronous Receiver/Transmitter. A device that converts parallel data into serial form for transmission along a serial interface, and vice versa.

ULSI Stands for Ultra Large-Scale Integration, and refers to the latest breed of silicon chips containing the equivalent of 100,000 or more components on one chip.

Unix A minicomputer operating system that is now available for some microcomputers. Although it has some useful features, it was originally designed to make life easier for programmers and is scarcely the friendliest operating system for commercial work.

Upward compatible Term used to describe software, such as an operating system or some types of hardware, to which extra facilities have been added. As an example, the new version of an operating system might be upward compatible with the old version, meaning that programs which work with the old version will still work with the new one, but that programs written for the new version will not work with the old one.

User friendly A piece of jargon to denote a computer or a piece of software that is easy to use by somebody who has no background in computer sciences. As low-cost computing power—in the form of microcomputers—becomes increasingly available to non-specialist users, user friendliness is becoming increasingly important.

Word processing The use of a computer, together with special software, to store and manipulate text. This allows an author to make easy alterations, produce perfect copies, or to combine, for example, a standard letter with a list of names and addresses to produce apparently personalized letters.

Winchester disks A particular type of hard disk in which the actual disks, together with their read/write heads, are housed in a sealed chamber. The result is a compact unit that can be integrated with a microcomputer and which can hold large amounts of information—typically five or ten million bytes. The disadvantage is that you cannot remove the disks and this makes it difficult to keep back-up or security copies of the information stored on them. Often, winchesters are sold with special tape back-up units on to which the information can be safely dumped in case something goes wrong with the disk unit.

Z80 The name of one of the most popular 8-bit microprocessor chips used as a base for microcomputers. Made by an American company called Zilog Inc., it forms the heart of most of today's CP/M-80 machines.
Comptometer 26
Compuserve 149
Computer
applications 22
documentation 101
games 119
inside 18
keyboards 76
logic 31, 54
magazines 42
problems 48
screens 39
setting up 34
system schematic 63
typical business system 15
typical home system 14
Computer design
chips 70
miniaturization 11, 32, 154
Computer history 24-32
Computer language see Languages
Computer logic see Logic
Computers systems
explanation 14-17
interface 16
Concurrent CP/M-86 136
CONTROL 76
Control bus 64, 68
Control Data Institute 160
Copies
with CP/M 135
digital disk 153
floppy disk 87
hard copies 88-91, 92, 94-5
tape back-up for hard
disk 86
tape programs 83
using two-disk drive 15
CP/M operating system
assembler programming
124-6
for business micros 98
CP/M Plus 136
CP/M-80 136
CP/M-86 136
CP/NET 136
and CPU 136
disk directory 84
family 136
function 134-7
memory 135
programming 100, 135
software portability 123
CPU
8-bit 15
16-bit 15
address bus 75
bit mapping 78
and CP/M 136
definition 14-15
function 66-9
home computer 18-20
memory system 61, 64, 74-5
multi-user 17
registers 64
schematic 68
Crashing 43, 47-8
Cromemco
disk operating system
see CDOS
Cursor control keys 77
Customer mailing 138-9, 162

D
DAD 150, 153
Daisywheel printer
Diablo 630 91
function 88-90
in system 15, 16
Data
and ALU 54
bus 64, 68, 74
entering 37
error 48
storage 54-7, 82-7
See also Applications
software, Cartridges,
Cassettes, Disks,
Floppy disk, Storage
peripherals
Database management
systems 23, 138-9
Data statements 110-11
dBASE II 139
Debugging 43, 117-18
DELETE 37
Diablo 630 printer 91
Dialects see Basic
Difference engine 27-8
Differential analyzer 29
Digital audio disk 150, 153
Digital computers
Babbage 29
Colossus 31
IBM 30
Zuse 30
Digital Research 128, 134
See also CP/M
Digital tapes 82
Digital-to-analog
converter 96
DIR 84
Disk drive
advantage over
cartridge/cassettes 42
function 84-5
twin drive 15
Disks
audio 152-3
compact 152-3
digital audio 150, 153
directory 84, 87
floppy see Floppy disk
format 84
hard see Hard disk
intelligent controller 64
and spreadsheet 141
video 150-3
winchester 87
Display see Monitor
Dot matrix printer
Epson MX-80 91
function 88-9
graphics capability 95
in system 13, 15

E-G
8-bit operating systems
134-6, 154
Eckert, J. Presper 31
Edison, Thomas 152-3
Editing keys 77
EDSAC 31
Educational applications
132-3, 159-60
END 107
ENIAC 31
ENTER 38, 77
EPROM 70, 75
Epson
HX-20 13
MX-80 printer 91
QX-10 21
ESCAPE 40, 76
Ethernet 158
Expert systems 146-7
Faults
debugging 43, 117-18
finder 47-8
logic error 117-18
syntax error 117-18
tuning 36
Felt, Dorr E. 26
Filing systems 45-6
FILL 116
Flags register 123-4
Flip-flops 60-2
Floppy disk
capacity 87
in computer system
15-16
construction 42-3, 84-5
storing subroutines 45
wear 87
See also Disk drive
Formatting 84, 135, 142
Forth 38, 99, 130-1
Fortran 127
Full adder circuit 59
Games
development of 119-20
paddle 16, 20
and TV set 148
GOSUB 45, 113, 117
GOTO 117-18
Grant, George Barnard 26
Graphics
applications 23
bit-mapped 79, 122
color 78
conversion 116
familiarization 38
image formation 79
joystick 94
memory 122
peripherals 94-5
plotter 16

H
Handshaking 81
Hard disk 16, 86-7
Hardware
assessing needs 162-3
cassettes/cartridges 82-3
chips 70-3
CPU 66-9
disks 84-7
exotica 92-6
input/output 76-81
memory 74-5
printers 88-91
word processing 144
See also individual
tenaries for above
Hardware and software
specifications 164-92
Hewlett-Packard
General Purpose Inter
face Bus 81
postfix notation 131
Hexadecimal system
converting opcodes into
126

204
explained 50-1
Home computers
choosing 162-3
components 18-19
games 23, 119-20, 148
graphics 121
hardware facilities 105
information banks 148-9, 152
system 14
and television 149-50
and video 150-2
See also 164-92 passim

IBM
company profile 168
Harvard Mark I 30
origins 29-30
Personal Computer (PC) 10, 21
winchester disk (3030) 87
700 31-2
1130 130
IEEE-488
function 12
and parallel interface 81
IF ... THEN 108
Information banks 148-9, 152
Initializing 84
INPUT 107-8
Input/output see I/O
INSERT 37
Intel
8080 134, 137
8086 69
8088 69, 137
Intelligent controller
chips 63, 64
Interfaces
Apple IIe 20-1
cassette 40-1
Centronics 64, 81
IEEE-488 12
monitor 12
parallel 81
printer 80
RS232 12, 64, 80-1
serial 80-1
storage system 34
within computer 16
Interpreter 100
Inverer 62
I/O
and C 129
connections 21
definitions 14, 16
functions 14, 76, 78, 134
handshaking 81
memory-mapped
system 80
parallel 81
ports 64, 80
serial 78
setting up 35
systems 76-81
See also Keyboard, Monitor
Jacquard, Joseph 29
Joystick 16, 94
Jupiter Ace 38
Laser technology 150-3
Learning techniques/ systems 132-3, 159-60
Leibniz, Gottfried 26
LET 108
Light pen 16
Line numbers 106
LIST 41
Logic
chips 19
circuits 56-7, 60-2
errors 117-18
functions 54-5
gates 56-9, 64, 68
shift 55
Logo 99, 132-3, 160
Loops 110-13, 132
faults 47
home system 14
IBM PC 10
image resolution 121
modulator 34, 36
Osborne 12
output function 76-9
tuning 36
TV screen 14, 34, 77
types 39
word processor 144
See also 164-92 passim
Moore, Charles 130
Morland, Sir Samuel 26
MOS Technology 69
Motorola 68000 137
Mosaic 142
MOS technology 150-3

K
Kbytes 52
Kelvin, Lord 29
Keyboards
Apple IIe 20
data input 16, 76-7
Epson HX-20 13
familiarization 37
IBM PC 10
Maltron 77
numeric keypad 77
Osborne 112
programmable function keys 76-7
QWERTY 76-7
Sinclair ZX81 78
Sinclair Spectrum 19, 37, 78
single keystroke 37
touch sensitive 37, 77
word processing 144
See also 164-92 passim
Krick, V. 139
Kanadian 132

M
Machine code 48, 123
Mainframe computer
CPU 15
definition 9-11
Maltron keyboard 77
Manchester Mark I 31
Math processor 64
Mauchly, John W. 31
Medical applications 146-7, 159
Memory
altering 62
bytes 50-2
capacity 52, 154-5
and CPU 74-5
definition 14-16
development 31
graphics 122
home computer 18-20
-integrated I/O systems 80
memory 77-8
systems 60-3
variable 67
See also RAM, ROM
Mach 142
McKeown, Janis 139

L
Languages
Algo 131
and algorithms 104
assembler language 123
Basic See separate entry
C 99, 128-30
Cobol 99, 128-30
development of 32
dialects 37-8
Forth 38, 99, 130-1
functions 98-101
high-level 123, 128
Logo 99, 132-3, 160
Pascal 99, 128, 131-2
Laser technology 150-3
Learning techniques/ systems 132-3, 159-60
Leibniz, Gottfried 26
LET 108
Light pen 16
Line numbers 106
LIST 41
Logic
chips 19
circuits 56-7, 60-2
ers 117-18
functions 54-5
gates 56-9, 64, 68
shift 55
Logo 99, 132-3, 160
Loops 110-13, 132
faults 47
home system 14
IBM PC 10
image resolution 121
modulator 34, 36
Osborne 112
output function 76-9
tuning 36
TV screen 14, 34, 77
types 39
word processor 144
See also 164-92 passim

N-O
NAND 54-5
NAND gate 56-9, 60
Napier, John 25
Networks
CPU 17
development of 32
information banks 148-9
interfaces 93-2
Project MAC 32
telephone system 64
von Neumann, John 31
NEW 41
Nibble 52
Number bases 50
Numeric keypad 77
Numeric variable 107
Opcode 124, 126
Operand 124
Operating systems
CDOS 136
CP/M 134-6
CP/M Plus 136
CP/M-80 136
CP/M-86 136
CP/NET 136
MP/M 136
MS-DOS 137
UCSD p 137
Unix 137
OR 56, 128
Osborne 112
Oughtred, William 25
Index

P-Q

PAINT 116
PAL 18
Papert, Seymour 132
Pascal 99, 128, 131-2
Pascal, Blaise 25
PEEK 116
Peripherals
  analog-to-digital
  converter 96
  digital-to-analog
  converter 96
  graphics 94-5
  linking systems 92
  speech recognizer 96
  speech synthesizer 96
Plato 160
Plotter 95
Pointers 129
POKE 48, 116
Portable computers 12-13
  See also 164-92 passim
Postfix notation 131
Power supply 34, 37
Prestel 148-9
PRINT 37
Printers
  connections 64
  daisywheel 15, 16
  dot matrix 13, 88-9, 91
  electrostatic 90-1
  filing system 45-6
  inkjet 90
  interface 81
  I/O system 80-1
  laser 90
  thermal 90-1
Program counter 68
Programmable function keys 77
Programs (and programming)
  algorithms 44, 102, 104, 107-8
  alterations 108-9
  and ALU 54
  analyzing 106
  arrays and loops 110-13
  assembler 123-6
  averages 109-10
  in Basic 105-16
  compiling 100
  conversion 116
  debugging 117-18
  development 31
  expansion 110
  flowcharts 44, 102-3
  function 13, 68
  games 119-20
  idiosyncracies 46
  languages See separate entry
  learning 42-3
  line numbers 106
  planning 44, 103-4
  portability 123
  roulette 113
  standardization 46
  subroutines 45
  writing down 44
  Project MAC 32
  Prospector 44
  PUSH 124
  QWERTY 76-7
  RX
Radio Shack (Tandy)
  company profile 175
  PC 11
  RAM
    definition 15-16, 18-20
    function 75
    memory mapping 77-8
    programming 100
  Random Access Memory
    see RAM
  READ 110
  Read Only Memory
    see ROM
  Read/write head 85, 86
  Registers
    in ALU 68
    in CPU 62, 68
    function 123-4
    in Z80 66
  REM 44-5
  REPEAT 76, 132, 133
  RESTORE 110
  RETURN 38, 45, 76, 113, 117
  RF modulator 34, 35, 36
  Ring network 17
  ROM
    cartridges 83
    definition 15-16, 18-20
    function 75
    memory mapping 78
    programming 100
  Roulette program 113-16
  RS232 12, 64, 80-1, 93
  RUN 105, 110
  16-bit systems 137, 154-5
  SAVE 48
  Saving programs 40, 42
  Scheutz, George 28
  Schickard, Wilhelm 25
  Screen see Monitor
  Search and replace 142
  Search function 22
  Serial interfacing 80-1
  Shannon, Claude 29
  Shockley, William 32
  Silicon chip see Chip
  Sinclair Research
    company profile 170
    Spectrum 18-21, 35-6, 37
    ZX Printer 90, 91
  Software
    algorithms 102-4
    assembler programming
      123-6
    assessing needs 162-3
    Basic programming
      105-16
    databases 138-9
    debugging 117-18
    games 119-20
    graphics 121-2
    hierarchy 98
    program planning
      101-3
    spreadsheets 140-1
    systems 134-7
    word processing 142-4
  Sorting routines 46
  Sound
    adjusting 38-9
    conversion 116
    See also Speakers
  Source, The 149
  SPACE 38
  Space Invaders 119, 120
  Speakers 19, 21, 38-9
  Speech recognizer 96
  Speech synthesizer 96
  Speed
    cartridge data loading 83
    cassette data loading 82
    computer 22-3
    disk drive 43
    interfacing 81
    printers 89, 90
    program loading 40, 100
    using Forth 130-1
    Sperry Rand 31
    Spreadsheet 23, 140-1
  Stack 124, 130
  Stack pointer 124
  Star network 17
  Start bits 80
  Stibitz, George 29-30
  STOP 40, 48
  Storage peripherals
    assessing needs 162
    binary notation 50-7
    function 15-16, 34, 82-7
    See also Cartridges,
      Cassettes, Data, Disks
  Strings 129
  String variable 107
  Stringy floppy 82-3
  Subroutines
    in Basic 113
    falling into 117
    filing 45
    using 45, 124, 126
  Syntax error 48, 105, 117
  System clock 64
  System crash 43, 47-8
  System hierarchy 98-9
  System prompt 134
  Systems software 134-7
  T
  32-bit operating systems 154-5
  Tandy see Radio Shack
  Tape back-up 86
  Text
    display 79
    editor 123
    handling 142-4
    indexing 143
  Thomas de Colmar, Charles X. 26
  Thomas, James 29
  Time Gate 120
  Timex see Sinclair Research
  TRACE 118
  Transistor 32
  TROFF 118
  TRON 118
  Truth tables 54-5, 57
  Tubes 30, 32
  Turing, Alan 30
  Turtle 132-3, 160
  TV screen
    and computer 148-53
    tuning 36, 38, 47
    See also Monitor
  TX-O computer 32
INDEX TO MAJOR MACHINE MODELS

<table>
<thead>
<tr>
<th>ACT Apricot 180</th>
<th>DEC Rainbow 181</th>
<th>Lynx 186</th>
<th>Sord CGL M5 191, M23 191</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Ile 165</td>
<td>Dragon 32 182</td>
<td>NEC APC 187, PC8000 187</td>
<td></td>
</tr>
<tr>
<td>II Plus 164</td>
<td>Elan Enterprise 182</td>
<td>North Star Advantage 188</td>
<td></td>
</tr>
<tr>
<td>III 165</td>
<td>Electron 183</td>
<td>Olivetti M20 188</td>
<td></td>
</tr>
<tr>
<td>Lisa 166-7</td>
<td>Epson HX-20 183</td>
<td>Oric 1 189</td>
<td></td>
</tr>
<tr>
<td>Aquarius 180</td>
<td>QX-10 184</td>
<td>Sharp MZ-700 190</td>
<td></td>
</tr>
<tr>
<td>Atari 400 178</td>
<td>Fortune 32:16 184</td>
<td>PC-1500 189</td>
<td></td>
</tr>
<tr>
<td>800 179</td>
<td>Future Computers FX20 185</td>
<td></td>
<td></td>
</tr>
<tr>
<td>600XL 179</td>
<td>Hewlett-Packard HP75C 185</td>
<td></td>
<td></td>
</tr>
<tr>
<td>800XL 179</td>
<td>HP86A 186</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BBC Microcomputer 181</td>
<td>IBM PC 168-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oric-1 189</td>
<td>IBM PCjr 169</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commodore B and P Series 174</td>
<td>IBM 8000 Series 174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIC-20 172</td>
<td>Sinclair Spectrum 171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 173</td>
<td>Z881 171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 Series 173</td>
<td>Sirius 1 190</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000 Series 174</td>
<td>Tandy TRS-80 Color 175</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRS-80 MC-10 175</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRS-80 Model 4 176</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRS-80 Model 12 177</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRS-80 Model 16 177</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRS-80 Model 100 176</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Texas Instruments CC-40 192</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>99/4A 192</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U-V
UART 64, 80-1
UCSD p-system 131-2, 137
UNIVAC-1 31
Universal Asynchronous Receiver/Transmitter see UART
Unix 128
User-friendly systems 134-6, 155
User manuals 34, 101, 116
Valves see Tubes
Variables 46, 107-8, 118
Video controller 122

W
Wilkes, Maurice 31
 Winchester disk 87
 Wiring faults 47-8
 installing 34, 36

X-Z
Xerox Corporation 158
XOR in C 128
 electronic logic function 56

Zilog compatibility of Z80 and Z8000 CPUs 137
Z001 CPU 67
Z80 66, 12
Z80 CPU 67, 69, 134
Z80 assembler language programming 123
Z80 internal architecture 123
Z8000 66
Z8002 CPU 67
Zuse, Konrad 30
ACKNOWLEDGMENTS

AUTHOR'S ACKNOWLEDGMENTS

I would like to thank the staff of Dorling Kindersley for their patience and good humour in dealing with my erratic observance of their deadlines, and Maggie Burton for her help and support during the writing of this book.

Peter Rodwell

Dorling Kindersley would like to thank the following individuals for their help in producing this book: Ivan Berti and Douglas Taylor for technical consultation and advice; Tim Shackleton, Cathy Meeus, Anne Cannon, Judith More, Heather Shackleton, Phil Wilkinson and Thomas Graham for additional editorial assistance; Martin Dohrn, Tim Woodcock, Paul Brierley, Tim Stephens, and Lynne Standford Taylor (photographic section Science Museum, London) for photographic assistance. Thanks also go to Polly Dawes for design assistance and to the following organizations for supplying information and equipment: Suren Patel (Digitus Ltd), Denise Iwata (Zilog Inc.), Lisa Dreske (Intel Corporation), Commodore Computer Systems Group, Robocom Ltd (London), Rod Holland (Currah Computer Components), Geoff Green (Wells O'Brien and Company, Manchester), Paul Young (Young Evett & Young, London), Mike Rudd (Sony), Michael Burman and Bob Green (F. E. Burman Ltd), and Brian Retter (Airedale Graphics).

Our very special thanks go to Vic Chambers and Tony Wallace (Chambers Wallace Ltd) for film setting and make-up services.

PICTURE CREDITS

Key

1 = top, 2 = bottom, c = center, l = left, r = right

Apple Corp 164 (b)
Paul Brierley 70-1
Commodore 69 (t)
Karin Craddock 94-5 (t), 94 (br), 95 (bl)
Martin Dohrn 20-1, 22, 35 (t & tr), 39, 57, 74-5, 78, 82-3, 84 (t), 88-9, 91, 93, 120 (l), 121, 122, 143
Philip Dowell 11, 18-19
Ferranti 72-9
Nicholas Harris 35 (b), 42-3, 119
Jonathan Hilton 41
Intel 69 (b) (courtesy Intel Corporation)
Rick Kemp 120 (r)
Peter Rodwell 23 (r), 32, 76, 79, 138, 139, 140-1
Science Museum, London
Tim Stephens 6-7
Texas Instruments 58
Tim Woodcock 10, 12, 14-15, Front jacket (UK and Australian editions)
John Walsh (Science Photo Library) Title page, 69 (t)
Xerox 156-7
Zilog 66-7 (“Reproduced by permission (1983) Zilog, Inc. This material shall not be reproduced without the written consent of Zilog, Inc.”)

ILLUSTRATORS

Kuo Kang Chen
David Worth (Line and Line)
John Bishop
Nicholas Harris
Mark Richards

PHOTOGRAPHIC SERVICES

Negs
Paulo Colour
W Photo
Software · Module 32

p-System (see Module 33).

A Pascal program comprises a main module and a number of subroutines or "procedures", which are called from the main module or from other procedures. Unlike C, Pascal imposes a strict rule that no procedure can be used until it has first been defined. This effectively means that you start your program with the "lowest" procedure and build up until you can define the main module right up at the end of the listing. This "bottom-up" approach has much to recommend it as it imposes a neat, orderly way of arranging the program, although there is an argument that putting the main module at the top of the listing makes it easier to decide what the program does when you initially come across pages and pages of program listing.

All variables must be defined before they can be used and their type given: REAL (numbers with decimal points) and INTEGER (whole numbers). The declaration of variables takes place at the start of the module:

```
VAR NUMBER, ERIC, TOTAL: REAL;
```

This line defines the variables NUMBER, ERIC, and TOTAL as real numbers.

Pascal has a good collection of operators. There are the usual +, -, / (division), and * (multiplication) plus an integer division operator, "div", which always chops off any decimals resulting from a division and returns an integer result (for example, 12/8=1.5 but 12 div 8=1). You can obtain the remainder of a "div" with the modulus operator, "mod". Transcendental functions (sine, tan, log, for example) are provided with a few unusual operators like "odd" (which tests whether a number is odd or even) and "round" (which rounds a real number to its nearest integer).

Pascal has three loop structures, the first of which closely resembles Basic's FOR . . . NEXT loop:

```
FOR X:=1 TO 10 DO
BEGIN
  Y:= Y+6;
END
```

The loop must be surrounded by BEGIN and END to indicate its limits. The TO can be replaced by DOWN TO to decrement the control variable:

```
FOR X:= 10 DOWNTO 1 DO
```

The WHILE . . . DO loop below tests its controlling position before the body of the loop is executed:

```
WHILE X<>Y DO
BEGIN
  Y:= Y+6;
END
```

while the REPEAT . . . UNTIL loop does the test at the end of the loop:

```
REPEAT
  Y:= Y+6;
UNTIL X = Y;
```

Input and output are handled flexibly, with the main commands being "writeln" (to display text) and "readln" (to accept data from the keyboard).

Pascal has found favor outside of the academic world with general business applications programming. It does produce good, efficient code and is easy both to learn and to use, despite its insistence on rather fussy points of syntax (a good example of a language designed for ease of compilation by the computer rather than for ease of use by the programmer). It is also suitable for general scientific work, but although some systems programming is carried out in Pascal, it is not a particularly useful language for this type of work.

Logo

Logo was pioneered by Seymour Papert (author of the best-seller Mindstorms) in the 1960s primarily as a teaching language for children. With its roots firmly in psychology and artificial intelligence (see Module 37), Logo is not, however, concerned solely with teaching about computers. It aims to give children a general introduction to problem solving with a computer, but its approach is to present itself in much broader terms so that a child, through using Logo, develops a systematic attitude to all general problem solving.

Logo allows the child to "teach" the computer, rather than the other way round, and in so doing allows him/her to develop an appreciation of the structuring of procedures and, by extension, an idea of how to express the solution to problems.

In the classroom, learning with Logo is made more enjoyable by the use of the "turtle". As originally conceived, the turtle is a robot animal consisting of a plexiglass dome on wheels. At its center is a retractable pen capable of drawing a line on the
sembler code into your new definitions, should speed be important.

Forth is probably not the ideal first language to learn, as its efficient use demands an appreciation of certain computing concepts. Central to these is the stack (see Module 31), which plays a far more major role in Forth than in most other languages. In a computer, the stack is a special, reserved area of memory. Data can be “pushed” on to the stack and “popped” off it in a “last on, first off” sequence. Forth seems to use the stack for just about every operation in which something is to be done to one or more items of data. Take, for example, the simple task of adding two numbers together. In Basic you could type something like this:

```
PRINT 3+4
```

and get the answer 7. Forth, however, uses a different sequence of mathematic operations, called Reverse Polish or postfix notation. (The type of notation used in Basic and many other languages is called infix notation.) The same operation in Forth would be typed:

```
3 4 +
```

in order to produce the answer 7. What happens is that Forth accepts each number, 3 and 4, and pushes them on to the stack. The “+” operator tells it to perform an arithmetic addition on the two top numbers on the stack, so it pops them off, adds them, and prints out the result. You can see this at work in the BOX definition above. Typing “42 EMIT” causes the number 42 to be pushed on to the stack; “EMIT” then grabs the number at the top of the stack, treats it as an ASCII code, and displays that character. Likewise, SPACES pops the first number off the stack and uses that as a loop counter to print that number of spaces on the screen.

Working in postfix notation is at first rather confusing (unless you have a Hewlett-Packard calculator, which also uses postfix notation), but it is a lot easier if you take the time to understand how the stack works and what happens when you ask Forth to carry out an operation using the stack.

The stack is also used in Forth (and in many other languages) for passing values from one operation to another. Sometimes, an operation will have pushed values on to the stack in an order that is different to the order required by another operation. Anticipating this, Forth provides words to manipulate the contents of the stack. You can make a copy of something on the stack without disturbing the stack’s contents, or you can jump down the stack, ignoring what is on top, in order to access something further down. You can also reorder the contents of the stack. Naturally, it takes practice to carry out these operations, and unbelievable chaos can result if you get it wrong.

Many users of Forth are, in fact, enthusiasts and hobbyists, and many Forth implementations are on microcomputers and small minis. But the language now also has a large professional following and has been used to implement some interesting applications, including database management, word processing, controlling film cameras, and even operating an entire observatory.

**Pascal**

Various attempts have been made to produce a programming language that would be all things to all programmers. Nobody yet has succeeded, but Pascal probably comes closer than most others – at least in the micro world – to being a genuine all-purpose language. Pascal was designed by its originator, Nicholas Wirth, following his first-hand experience in defining an “all things” language called Algol 68. Algol 68 was defined by a committee, of which Wirth was a member, as the general-purpose language, and the result was a vast and complex language which has not gained particularly wide acceptance.

After this experience, Wirth felt improvements could be made, and so set about designing an improved language incorporating many major Algol 68 features. The result of this is a language with considerable power, which can also be easily implemented on small computers – including micros.

Algol, as its name implies, was intended to allow the programmer to develop an algorithm and then translate this easily into a program. Pascal continues this approach and imposes a very strict structure on the resulting program to ensure the programmer adheres to this principle. As a result, Pascal has gained wide favor in academic circles as a teaching language: students learn good programming habits simply because Pascal forces them to develop and use a particular approach. In addition, Wirth tried to make sure that the language was as portable as possible, although others, subsequently, have developed the language and extended various of its facilities. This has, somewhat, eroded its portability. Currently, its most widely used implementation in the micro world is that developed by the University of California at San Diego, known as UCSD Pascal – there is even an operating system based around this implementation called the UCSD
Dorling Kindersley
Computer Books

The Personal Computer Handbook heralds a series of quality computer books from Dorling Kindersley. With the emphasis on superb graphics and clear, integrated text, these books will form a new benchmark in computer publishing.

Watch for:

• A complete course of screen-shot programming manuals for the Sinclair Spectrum, the BBC Micro, the Commodore 64, the Acorn Electron, and the other best-selling personal computers
• Peter McWilliams – "the Dr Spock of computers" – phenomenally successful US author, now published in book form for the first time in the UK

AND

Dorling Kindersley Software

A list of personal computer programs offering strong editorial content, excellent graphic presentation, and great value for money.

Watch for:

• High-quality arcade games
• Big-name adventure games
• Classic strategy – with a difference
• Educational software opening up exciting new avenues

DORLING KINDERSLEY LTD
9 Henrietta Street
London WC2E 8PS
THE PERSONAL COMPUTER HANDBOOK

THE FULLY ILLUSTRATED, FULLY COMPREHENSIVE GUIDE FOR EVERY MICRO USER

How computers work - from RAMs and ROMs to bits and bytes

A jargon-free guide to computer software - for games, business and education

An inside look at the hardware - what goes on beneath the lid

Contents include: What computers can do ★ How computers think ★ Setting up and getting going ★ Running programs ★ Fault-finding ★ The hardware and the software ★ Writing programs in Basic ★ Debugging ★ Games and graphics ★ Word processing ★ Buyer’s guide to choosing a computer ★ and much, much more

“It is unlikely that any of this year’s micro intro books will outsell The Personal Computer Handbook”

Micro Business, October 1983

DORLING KINDERSLEY