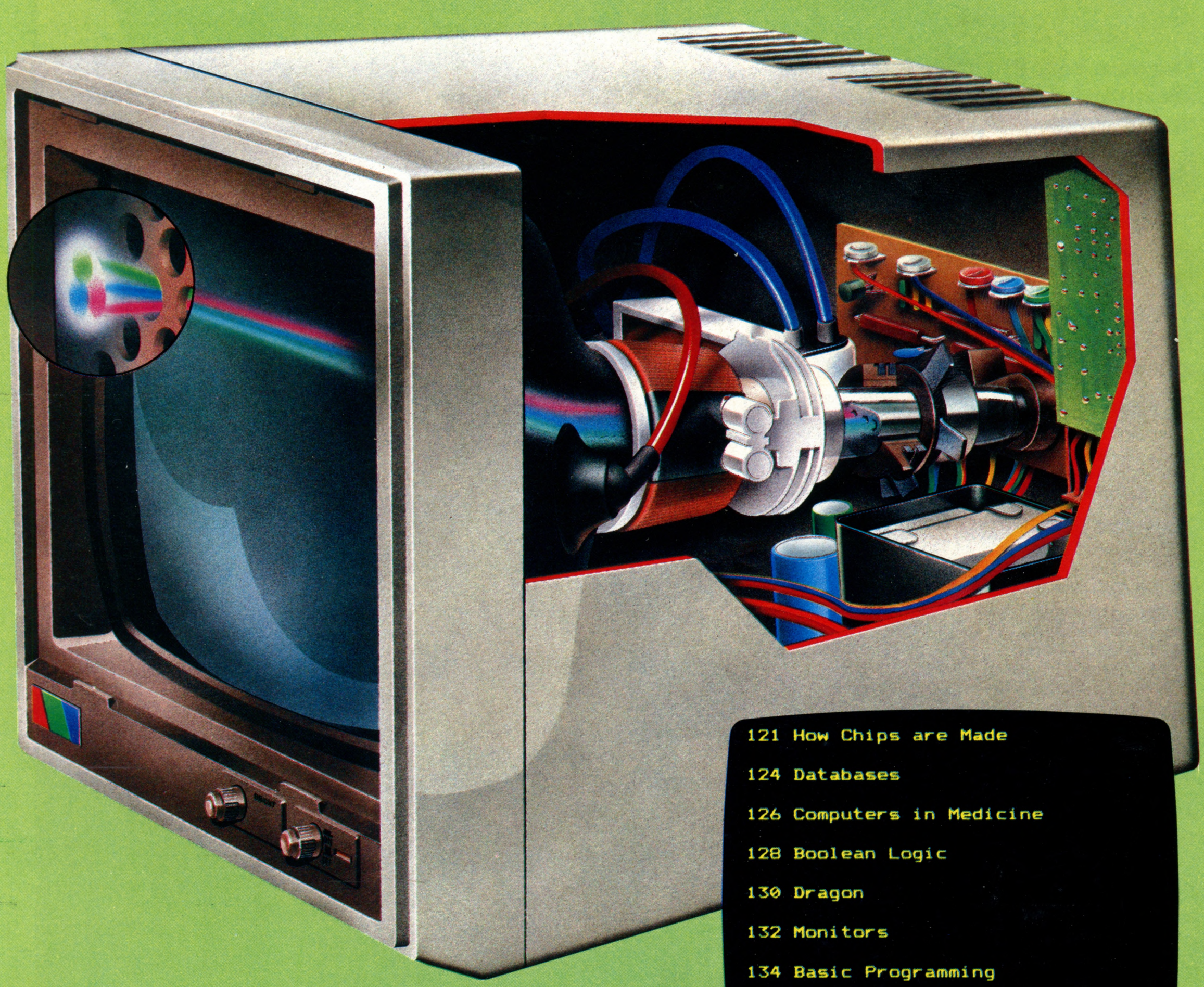
ISSN 0265-2919

R 80p

THE HOME COMPUTER COURSE 7

MASTERING YOUR HOME COMPUTER IN 24 WEEKS



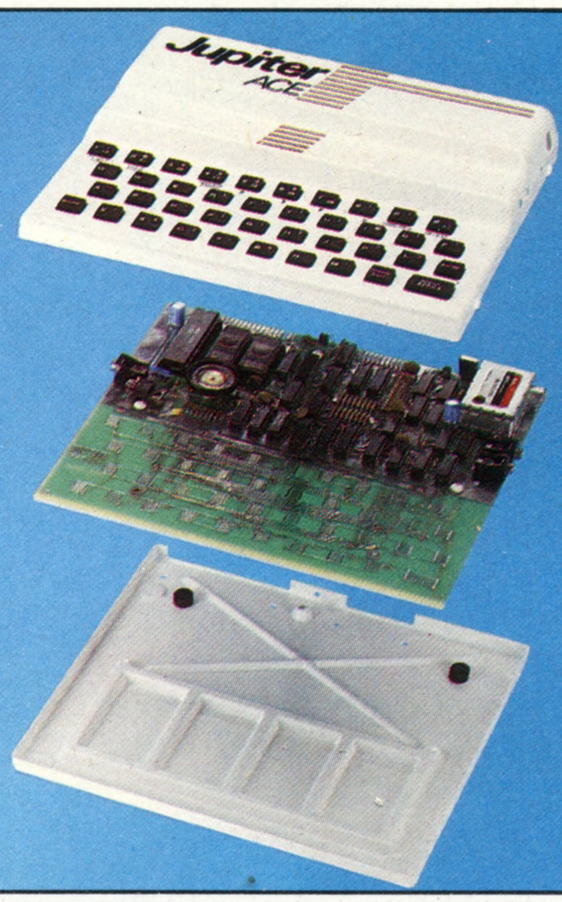
An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- We review the Jupiter ACE, a low-cost computer and the only machine to use FORTH as its standard programming language
- We take a look at how computers are infiltrating the world of music, and what home computers have in common with professional synthesisers
- Sprites are the most convenient method for creating moving objects on a computer screen — we explain how they work



Editor Gareth Jefferson; **Art Editor** David Whelan; **Production Editor** Catherine Cardwell; **Picture Editor** Claudia Zeff; **Designer** Hazel Bennington; **Art Assistants** Steve Broadhurst, Liz Dixon; **Sub Editor** Teresa Bushell; **Researcher** Melanie Davis; **Consultant Editor** David Tebbutt; **Project Manager** Jimmy Egerton; **Contributors** Tim Heath, Roger Ford, Richard King, Ray Hammond; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; © 1983 by Orbis Publishing Ltd: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95
How to obtain your copies of HOME COMPUTER COURSE – Copies are obtainable by placing a regular order at your newsagent.
Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.
How to obtain binders for HOME COMPUTER COURSE – UK and Eire: Details of how to obtain your binders (and of our special offer) are in issues 4, 5, and 8. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.
Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

Miniature Engineering

Scientists discovered that the microchip could be developed by using one of the earth's most abundant commodities

Silicon occurs naturally all over the surface of this planet in greater abundance than any other element except oxygen, with which it combines to form silica. Many of us spend most of our holidays lying on it and making our castles out of it. The whole of the microelectronics revolution is built on sand!

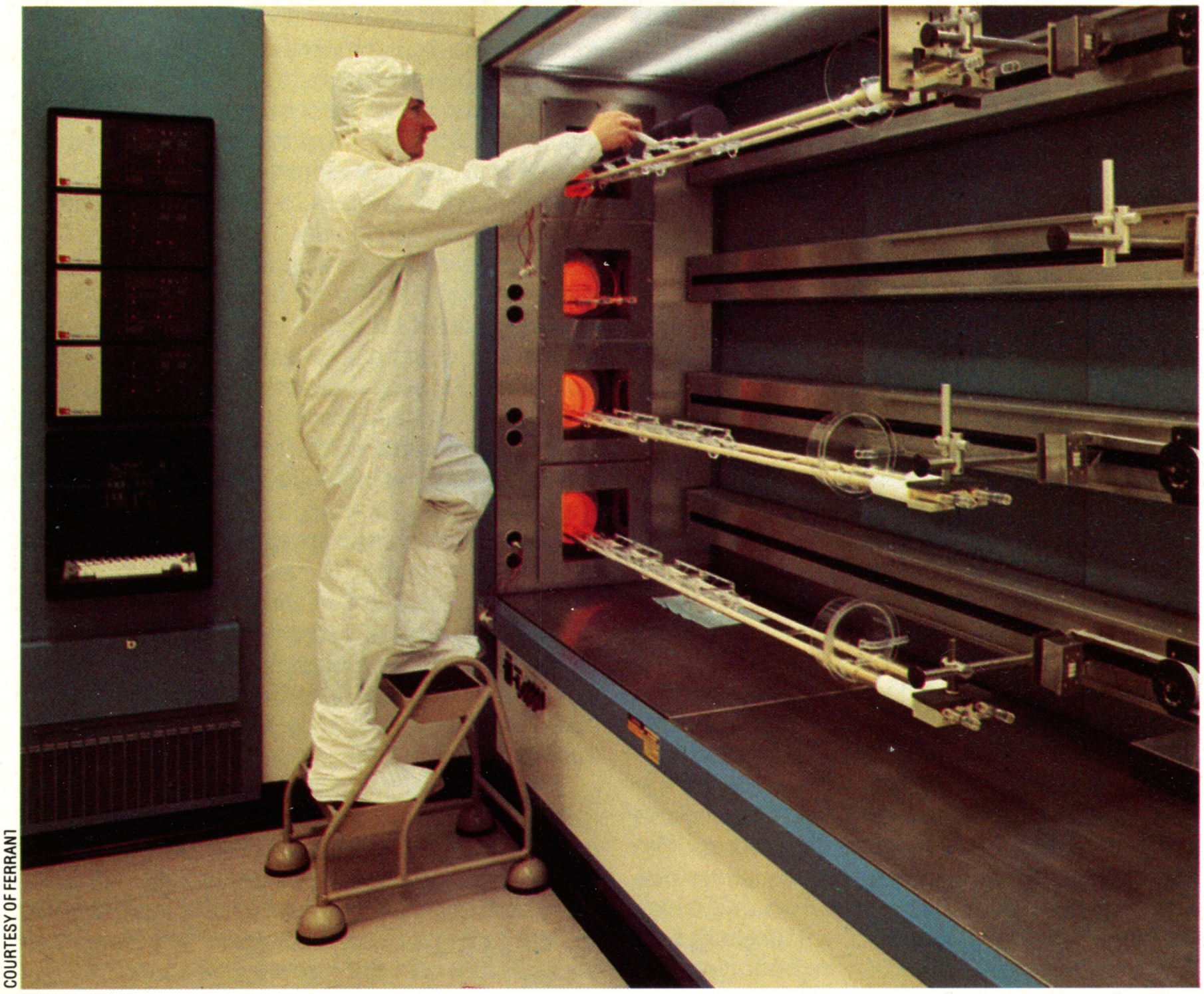
The importance of silicon to the microelectronics industry lies in its physical structure. In its pure form silicon is an extremely poor conductor of electricity. However, when controlled amounts of certain impurities are introduced, silicon becomes a semi-conductor.

The conduction of electricity through a substance is determined by the number of electrons in each of its atoms and the way they are bonded together. In metals, electrical current is carried by electrons without any firm bond. These electrons are free to wander about inside the atomic structure, transferring their allegiance with their electrical charge from one atom to another. In an insulator all the electrons are firmly bonded together; therefore a current cannot pass from one point to another.

The manufacture of pure silicon is a simple process. First, the raw oxide is refined chemically until it is 99.99 per cent pure. It is then placed into a crucible and heated to melting point, which is 1410°C (2570°F), in an atmosphere of purified inert gas, to keep out unwanted elements.

The process of introducing controlled amounts of impurities (known as 'doping') calls for the pure silicon to be combined with phosphorus, which produces 'n-type' silicon (so known because it carries the negative charge), or boron, which forms 'p-type' or positively chargeable silicon.

A large crystal is grown by introducing a perfect 'seed' crystal into the melt, and slowly withdrawing it, turning it at the same time.



COURTESY OF FERRANTI

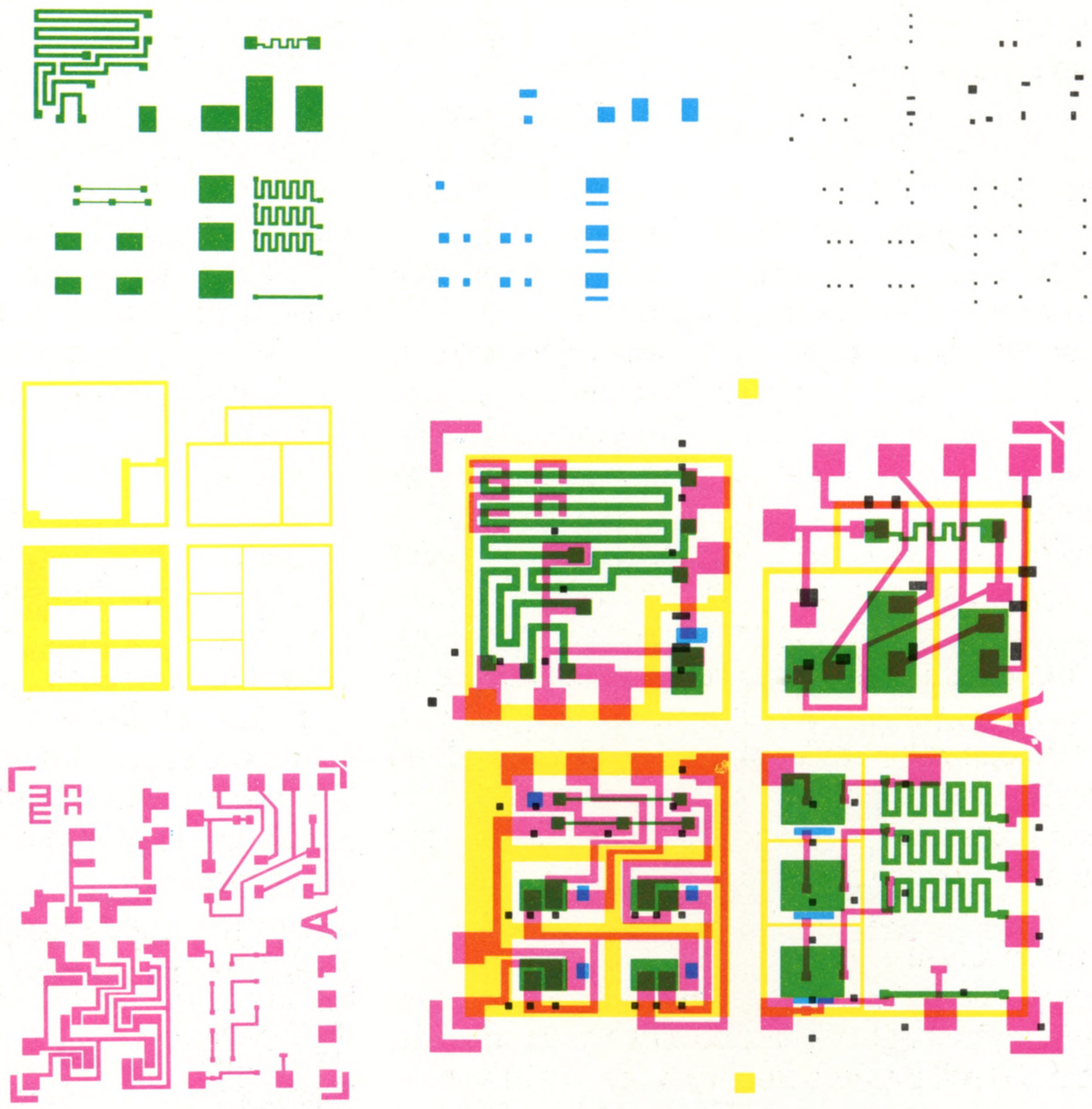
The Chip Furnace
The photograph shows a silicon chip furnace at work. Once the silicon has been refined, sliced, polished, masked and etched, it must be given a surface coating of silicon dioxide. This is done by heating the wafers to 1050°C (1920°F) then passing a stream of very pure oxygen or superheated steam over the wafers. While they are in the furnace, inside the 'boat', which is made of fused quartz, a layer of silicon dioxide forms over the surface of the wafer.

This layer is then selectively removed by the next etching process, and the cycle is repeated for each layer of the chip

The Sum Of The Parts

The integrated circuit fabrication process calls for each layer of the circuit to be deposited separately onto the surface of the silicon wafer. This technique is necessarily very similar to the way we print HOME COMPUTER COURSE's colour illustrations. Each photograph is first separated into three colours and black. These four are then printed, one on top of the other, in very precise registration to achieve the effect you see on these pages.

The chip-making process uses deposited layers of doped silicon and other materials instead of ink, but the printing process is much simpler than the masking/depositing/etching process. As you can see from the photograph the individual masks are combined in sequence in order to build up the microcircuit in this case, a very simple transistorised device



Crystals seven to ten centimetres (three to four inches) in diameter, and 60 centimetres (two feet) or more long can be produced in this way. These are then ground to a standard diameter, usually either 76 millimetres (three inches) or 100 millimetres (four inches). The crystal is mounted, cut into slices and ground flat on both sides, before being polished on one side only. The finished 'wafer' is typically half a millimetre (a fiftieth of an inch) thick.

If the process is reasonably simple, and the raw material is so abundant, why is the silicon used in chips so expensive — at £10 per slice?

The answer lies in the absolute necessity for maintaining its purity. Extraordinary care must be taken to ensure that foreign bodies are excluded. Purity levels in the air of wafer fabrication plants are truly remarkable — fewer than 3,000 particles per cubic metre (100 particles per cubic foot). This is more than 100 times as pure as the air in modern hospitals.

The manufacture of an integrated circuit requires a method of microengraving the surface of the chips. In mass production, this is achieved by a process known as photo-lithography, similar in many ways to the methods used to produce this publication.

Each 'layer' of the circuit is treated as a separate entity right through the process. The original artwork is produced by computer-aided methods, and is converted to a line and block photograph, which is then reduced to actual size. The mask itself is formed by reproducing this photograph many times in a grid pattern to cover the whole surface of the wafer.

The wafer is first heated to 1050°C (1920°F) in an atmosphere of pure oxygen. This causes a layer of silicon dioxide to form on the surface, to act as an insulator. This layer is then selectively removed to form 'windows' on the pure silicon beneath. This process is repeated for each successive stage in the building up of the integrated circuit on the face of the silicon substrate.

The oxidised surface of the wafer is first coated with 'photoresist', a light-sensitive material whose solubility is greatly decreased by exposure to ultra-violet light. A mask in the shape of the first layer of the microcircuit is introduced between the coated surface and the light source. The surface is exposed to ultra-violet light, and then 'developed' in a solvent that removes unexposed photoresist.

A similar method is used in making copper etchings. The surface of a copper sheet is coated with wax, a design is then scratched through the wax coat, and the whole immersed in acid. When the wax is removed, the design remains etched into the surface of the copper. Where the sealing coat of wax is unbroken, no reaction takes place.

The first stage completed, the whole process is repeated with different masks and reactive chemicals until the desired circuit has finally been built up. The entire wafer is then coated non-selectively with a further layer of silicon dioxide.

This process may require ten or more repetitions of the coating/masking/etching

process. At each stage, the scope for errors is large, and the failure rate at the quality assurance and testing stage is very high indeed. The more complex the microcircuit and the more densely packed the wafer, the more wastage occurs.

The first stage in testing calls for the entire wafer, with its many hundreds of identical integrated circuits, to be fitted into a computer-controlled rig that tests each circuit.

Nine chip-sized locations on the wafer are often reserved for testing purposes.

The test rig not only marks each reject with an ink spot, but also extracts information on failure rates for each wafer, locating failures within both the wafer and the individual chip and identifying specific faults.

Wafers are then cut into individual chips, and the rejects are extracted manually. The remaining chips are mounted on miniature metal stampings.

Connecting pins on the chip are hooked up to the stamping with very fine wires — again in a process controlled by computer — and the whole is sealed into a plastic or ceramic carrier, from which connecting pins protrude when it goes for final and more exhaustive testing.

Currently, the limit on the degree of miniaturisation, which lies in the photo-lithographic stage of manufacture, is the wavelength of light — around a two-millionth of a metre (20 millionths of an inch). Recent advances are centred on the use of X-rays, which will allow perhaps a 50-fold reduction in the size of microelectronic circuits.

Almost all aspects of chip manufacturing, from the design stage through prototyping, to mass production and final testing, are so complex as to be practically impossible without the use of the very devices that are being produced – a remarkable paradox!

The application of computers to the design process, for instance, allows sub-sections to be pre-defined and called up from memory each time they are needed.

Take the case of a random access memory (RAM) chip, where each individual bit is held in a single-transistor storage cell. To store 2 Kbytes the chip will need to contain an array of 16,384 identical cells. It is a simple matter for the designer to define the structure of a 'master' just once and then instruct the design computer to repeat it 16,384 times.

The use of computers in the design department is not limited to this useful but quite simple trick. Draughtsmen now work, not at drawing boards, but at Visual Display Terminals, and use light pens to 'draw' directly on to the monitor screen. The finished drawing is prepared by the design computer directly using a multi-colour plotter.

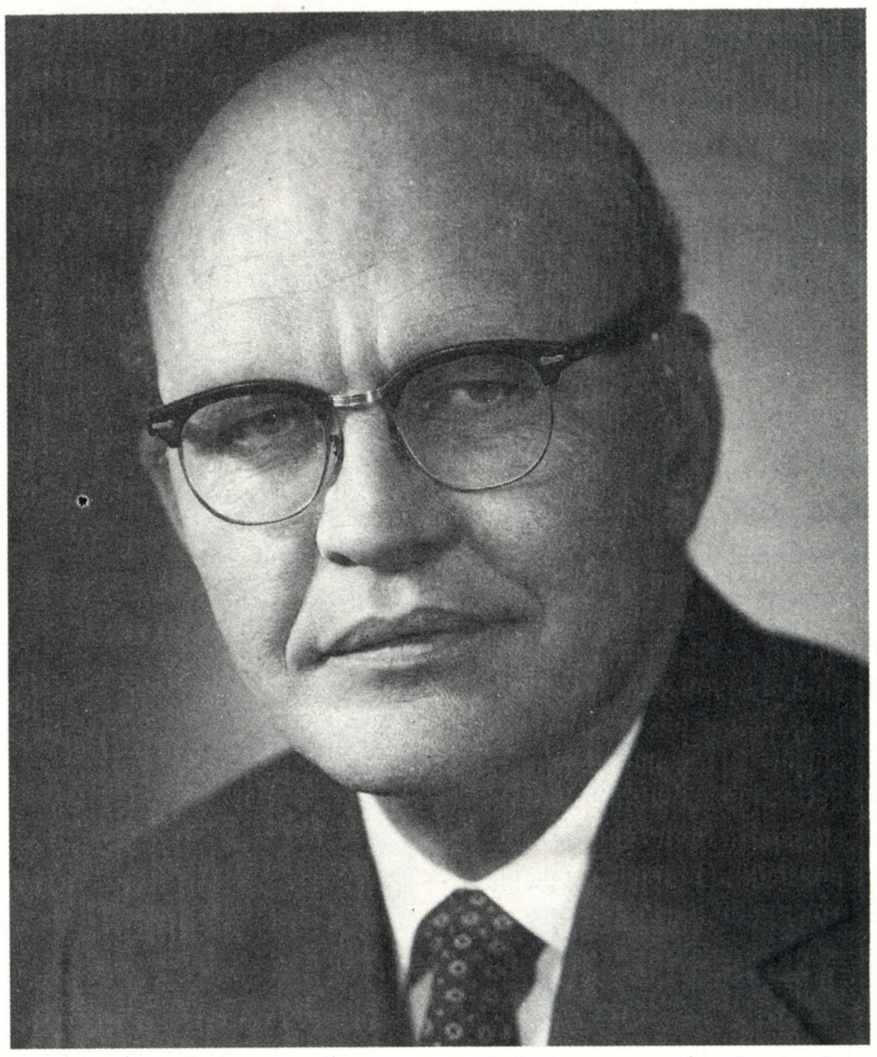
It's not only the graphic design process that is improved by mechanisation. On page 103 we describe the creation of a computerised model of an oil production field. This same modelling technique can be applied to circuit design, also, allowing the designer to try out a variety of different solutions before committing himself to a costly manufacturing process. By this method much of the very expensive trial and error prototyping can be avoided.

When we speak of the 'microelectronics revolution', we are talking about quantum jumps in speed and cost- and size-reduction. Perhaps, at this point it would be worth recalling the characteristics of a valve based computer:

Large physical size
Slow processing speed
High power requirement
Limited memory and instruction set
High cost

All these factors were affected by the discovery of the transistor (see page 46), but in terms of the manufacturing process, the computer industry was still a very labour-intensive, and hence, costly, business. Discrete components still had to be assembled onto Printed Circuit Boards.

It was the discovery of the integrated circuit – and more especially of the microprocessor – that allowed the industry to take full advantage of the advances in computer-controlled manufacture and processing.



COURTESY OF TEXAS INSTRUMENTS LTD

A Microelectronics Revolutionary
Jack Kilby is generally credited with the discovery of the integrated circuit while working at Texas Instruments in 1958. He constructed a single package, half an inch by a quarter that contained a number of transistors. Modern electronic circuits contain hundreds of thousands of devices in the same space

Some of the statistics available are quite breath-taking. For example, in 1959 only one component could be made on an integrated circuit – one diode, for instance, or one transistor. By 1978 the densest of Large Scale Integrated Circuits (LSI) had more than a quarter of a million components on a single chip. Over an even shorter period, 1973 to 1983, the cost per bit of computer memory has declined by a factor of twenty, and the use of electronics components of every type, world-wide has increased by a factor of one thousand. And these trends are likely to continue into the foreseeable future. It has been estimated that the number of devices used every year will increase one hundred-fold over the next three years.

Fact Finder

A computer can select facts and compile lists from the vast amount of information stored in a database

An accumulation of data that is held on and accessed by a computer is known as a 'database'. We all use various non-computerised databases as part of our everyday lives.

The telephone directory is an example of a non-computerised database. However, information does not have to be indexed or stored in a particular order to be a database. On a computer, such fixed ordering actually creates severe limitations.

A database program is a set of routines that allow selections to be made from the data. Such programs range from simple card-indexed systems to complete languages.

Typically, a computerised database will be large and contain data of many different types. But this does not mean that you have to be the owner of a huge machine. Any computer can handle a useful database. The only real limitation is the size and speed of the storage facilities.

We could, for example, make a list called 'people' containing facts about various persons. If we put these on an ordinary card index, we get a list that looks like the one called Personal Index. It is immediately apparent that there are several different kinds of information here. Within each category, many of the items are simple words, and some are numbers. There are limited possibilities within two of the categories: 'sex' can only be male or female and 'marital status' must be one of the set 'single, married, divorced, or widowed'.

It might be useful to make certain items into lists of words and numbers. For example: occupation, name of company, business address, business telephone, and name of superior could usefully be grouped under the heading 'business', and both types and age of car could be members of the car list.

By extending this idea, we can keep all addresses as lists. This is better than keeping an address as a single item, since we might want to know which town a person lives in, but not which street.

'Marital status' can also be extended by adding the name of the spouse where appropriate. This

TONY LODGE

Anyone For Tennis?
We have used the Rubik's cube as an analogy for a raw database — that is, one that contains all the information we may need, but as yet has not been manipulated into the right order. In this example we are looking for a tennis partner (the racquet symbol), who is a car owner (the car), and who is free on the day in question (the red squares)

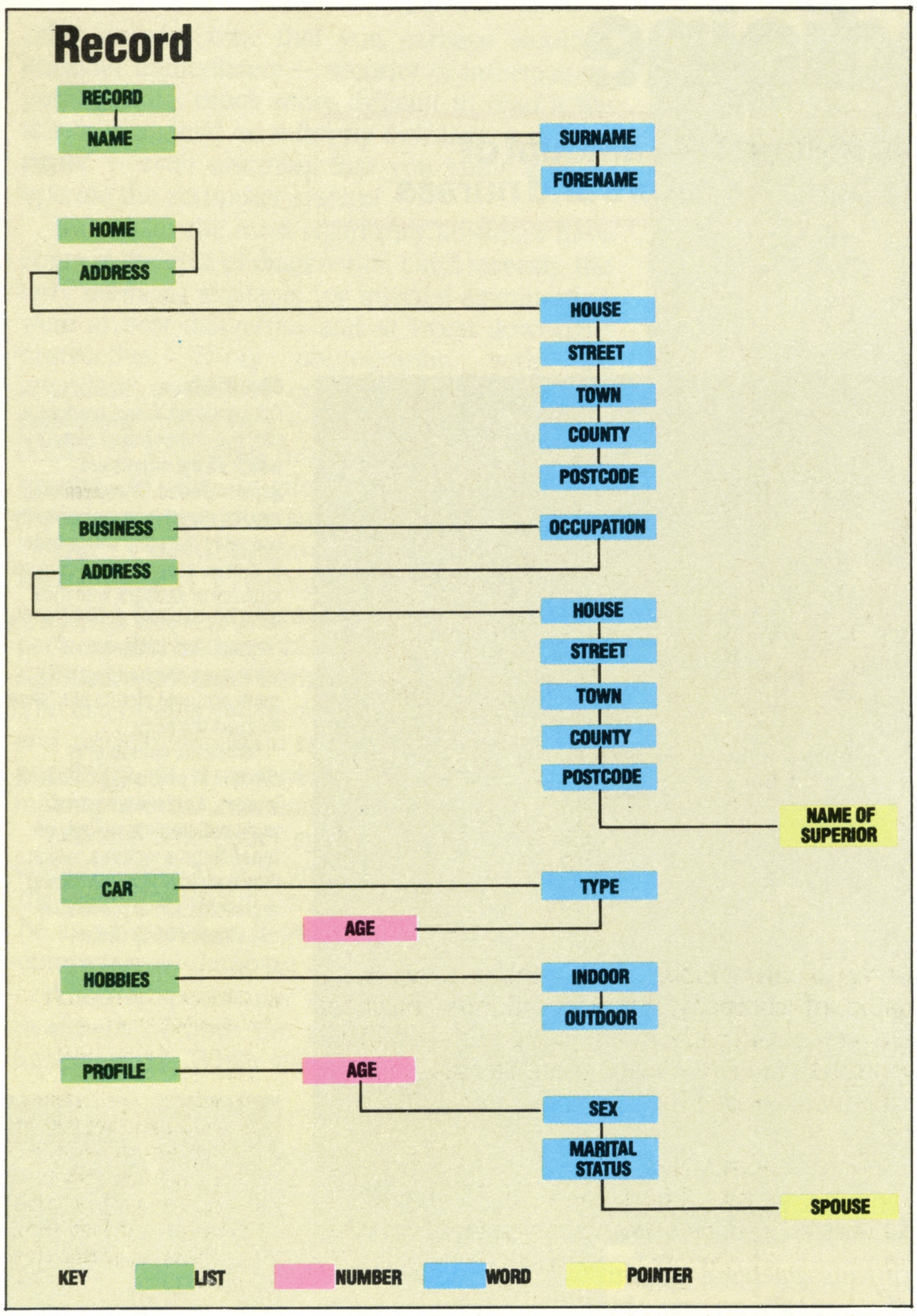
```
DATABASE  HCCSOFT

ENTER DATE AS DD/MM/YY:  14/10/83

.ACCESS

*use FRIENDS
*while FRIENDS, do CARS;
*while HOBBIES, do TENNIS;
*end

_CLOSE
```

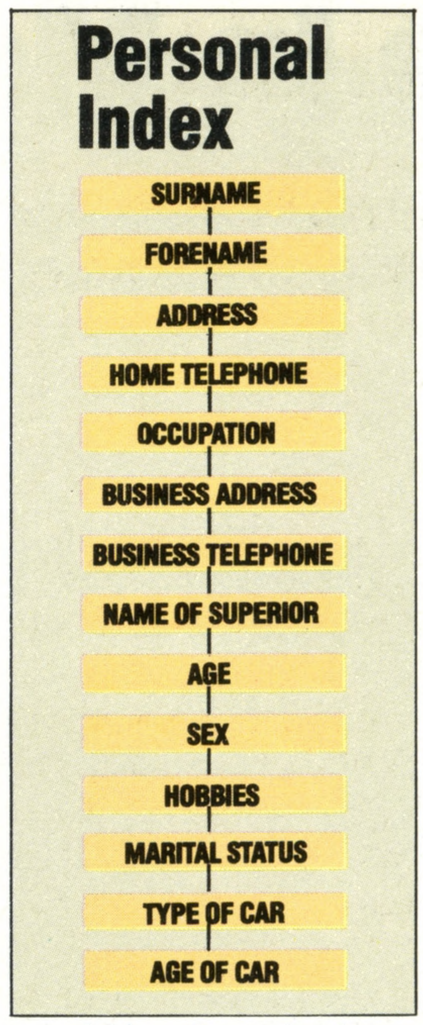
```
]FRIENDS, CARS subset hobbies: TENNIS

FIONA DAVIES          253 8146
CYNTHIA GRIMSHAW      704 1121
ROBERT MITCHELL       877 6202
DAVE WHELAN           582 5360
MARGRET JENKINS       345 1314

]END
```

Asking The Right Questions

Most commercial database programs use codes that are almost programming languages in themselves. The code we've used here is typical. 'ACCESS' indicates to the program that we wish to 'interrogate' or ask questions of a file that we have previously created. 'Use' indicates that we are going to use a sub-set of the file called 'FRIENDS', and 'while' these conditions are true, we are going to pull out all the items that are tagged with 'CAR' or 'TENNIS'.The result is a list of friends, with their telephone numbers, who both drive cars and play tennis

IAN McKINNELL





Organised Information
An hierarchical database leads the user from one piece of information to another, offering choices at every turn. No knowledge of the contents is assumed

could just be a single word, but since this refers to a person, and the file is about people, it would be useful if we could somehow refer to another record instead.

Since each record is in a particular place in the file, it has a number. So we can use the number of the record that describes this person, instead of the person's name, to establish a cross reference.

Such an item of data is called a 'pointer' record. If we use this technique to refer to the person's business superior, the result is a structure that looks like the one called Record.

The difference between a card index and a computerised database is that the former can have only one ordering, usually alphabetical.

The card index is adequate if we want to find out – for example – what company employs a specific person. But what do we do if we want the names of all the employees of a particular company?

Using a card index, we would have to look through all of it, extracting each card for the required data. To do so is not only time-consuming, but is likely to result in errors.

With a computerised system, however, we can ask the machine to look at each record in turn, and to print out the name of every person who works for the company in which we are interested.

Alternatively, we could have the machine re-order the file, with the company field being the most important item. This will result in the same database with the same data, but with a completely different 'shape'. The reordering will place all occurrences of a given company in one group, and that section will give us the names of all employees. In a card index, there is only one primary item, the surname. But in a computerised system, any item can be primary.

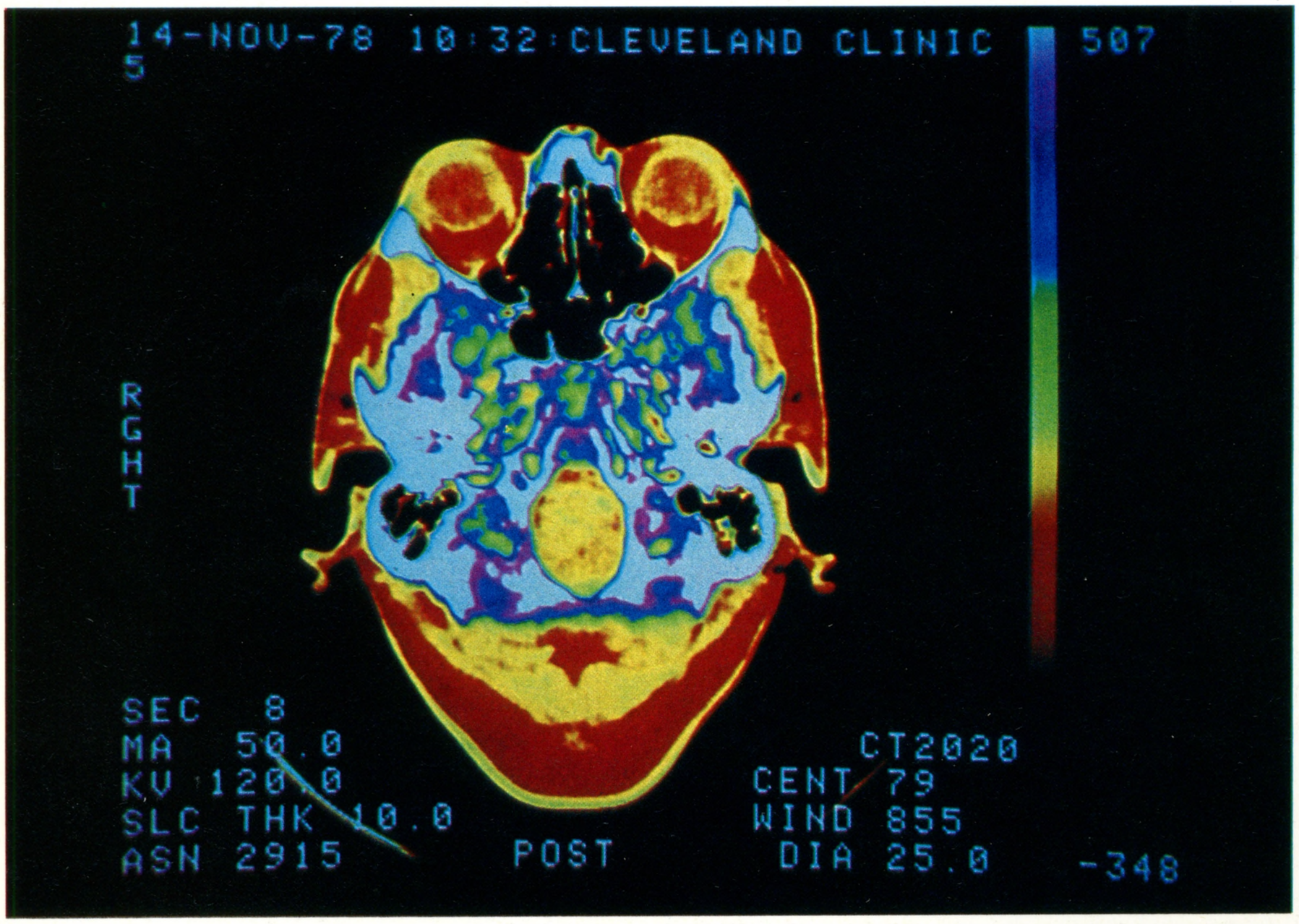
Using this principle, we can 'reshape' the database. For example, the cars owned by people could be primary, or the names of towns in which the people live. This is the great advantage of the computerised database system.

Today's Encyclopædia
Services like Prestel are an attempt to make huge databases available to the general public by means of services that they already have in their own homes. Viewdata systems use the domestic TV as the monitor screen, and a specially devised keypad, connected to the central computer over an ordinary telephone line, as the keyboard. Access to the database is by means of a 'menu' of available services. The choices are displayed on the screen, and the user arrives at the desired 'page' of information by working his way through the hierarchical tree. By entering a Credit Card code, purchases can even be made from home

COURTESY OF PRESTEL LTD

Micros In Medicine

An invaluable asset, the computer has reduced the amount of mundane work taking up the precious time of doctors and nurses



Slice Of Life
Conventional X-rays produce a flat, two-dimensional picture in which all the organs are superimposed. These pictures require expert interpretation. By scanning the body with a beam of X-rays, collecting the data on an array of sensors, and then using a computer to process the 'signals' into images on a screen, we can build up a far more accurate picture of a 'slice' through the body.

While the computer is capable of producing coloured images, as illustrated here, most radiologists still rely on monochrome pictures, which they consider to give a clearer representation of the relative densities of body tissue

In common with so many other professions, where highly trained — and very expensive — personnel are required to expend a great deal of effort in relatively mundane tasks, medicine has benefited considerably from devices as adaptable and inexpensive as the microcomputer.

Shortly after the general introduction of microprocessor-based devices, intensive care units, in particular, changed their working practices considerably. Microcomputers soon became commonplace as monitors of pulse, respiration, and blood pressure, giving instant readings on multiple conditions, and thus freeing valuable nursing staff for less mundane duties.

More recent developments include an increasing reliance on micros in general practice and hospital administration for maintaining patient records, notifying patients of appointments and controlling pharmaceutical stocks.

We talked about one important application of computers in the field of medicine on pages 72 and 73, but it is worth looking now in more detail at how Expert Systems are used in general practice.

Mickie, a simple but successful system for providing general practitioners with an over-view of a patient's current physical condition, is really an 'inexpert' Expert System, in that it does not seek to provide an exhaustive diagnosis, but only provides very general information.

The questions that the system asks of the patient are always in a form that can be answered by 'yes', 'no', 'don't know' or 'don't understand'. If the response is 'don't understand', then the system makes some effort to help the patient towards an answer.

Rather than presenting the patient with a full typewriter keyboard, the response is by means of a box with just four keys, labelled appropriately. When run, the system is rather slow, but that is an artificial limit, geared to average reading speed. The next step in the development of an expert system for general practice would probably be to link this newly acquired data with historical information. If it appeared that the patient has complained of these same symptoms on a previous visit, then the doctor might need to ask only one question to make his diagnosis — 'Are you feeling like you did in...?' Other advantages that computerised medical records

An Unsleeping Eye
As well as giving the nursing staff an indication of the patient's current condition, the monitoring system can also store information for later analysis. Here, a doctor is shown extracting very precise information about the patient's condition during the night. This helps him towards a better diagnosis

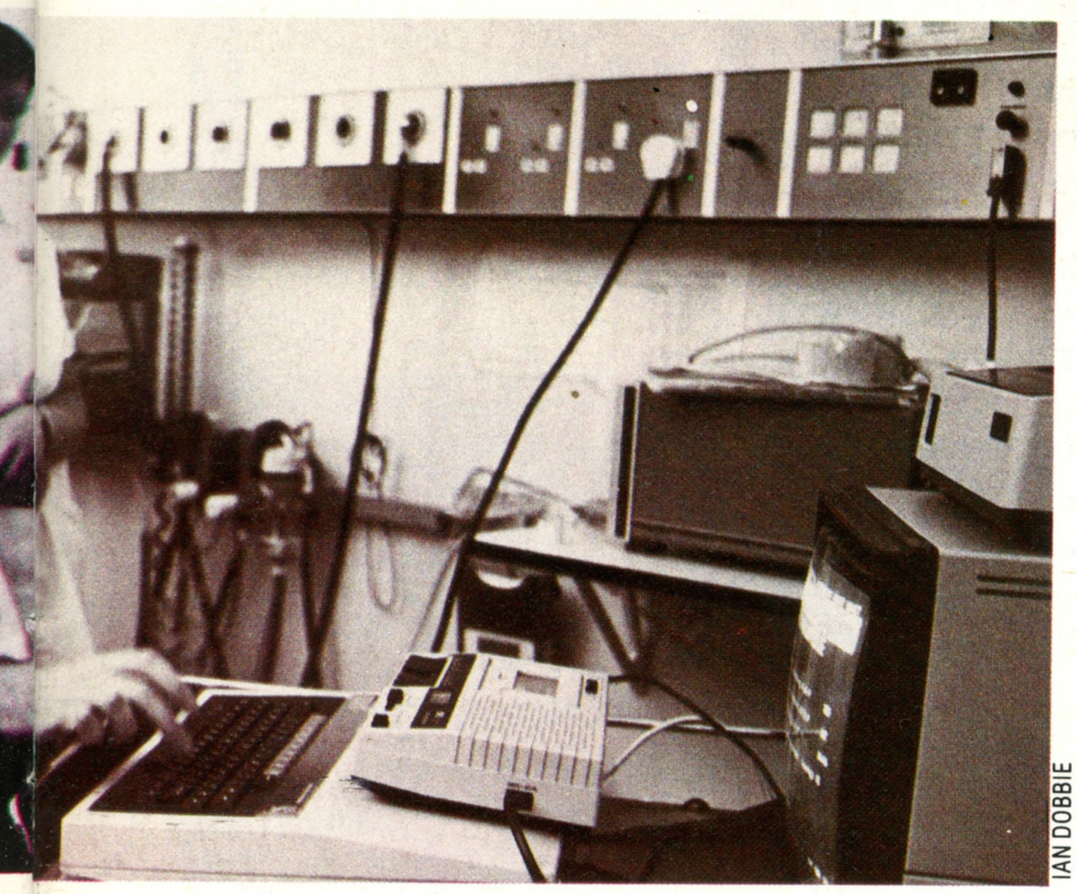
offer include some that you perhaps wouldn't consider immediately — security of information, perhaps. It is much more difficult to read a file that is contained on a floppy disk than it is from paper — even assuming that you know how to operate the computer system!

But by far the most significant advances have come in the area of diagnostics. Until recently, the only methods available for internal examination were at best dangerous and at worst downright destructive: X-ray photography, with the attendant possibility of excessive exposure to radiation; endoscopy — the insertion through an orifice of a viewing tube — which could cause trauma to the fragile internal tissues; and exploratory operative surgery.

The development of computerised tomographic (CT) techniques, which utilise a thin scanning beam of X-rays, rather than exposing parts of the body to blanket radiation, are one significant advance both in accuracy and safety. The use of alternative media, such as ultrasonics and nuclear magnetic resonance, holds out the possibility of a completely harmless way for making internal examinations.

Medical emergency systems, too, benefit from computerisation. It is unlikely, for example, that the organ transplant programme in Britain would be as advanced as it was without the benefits of computerised database methods.

Another spectacular advance has been in the treatment of both the physically and mentally handicapped. While still in its infancy, potential

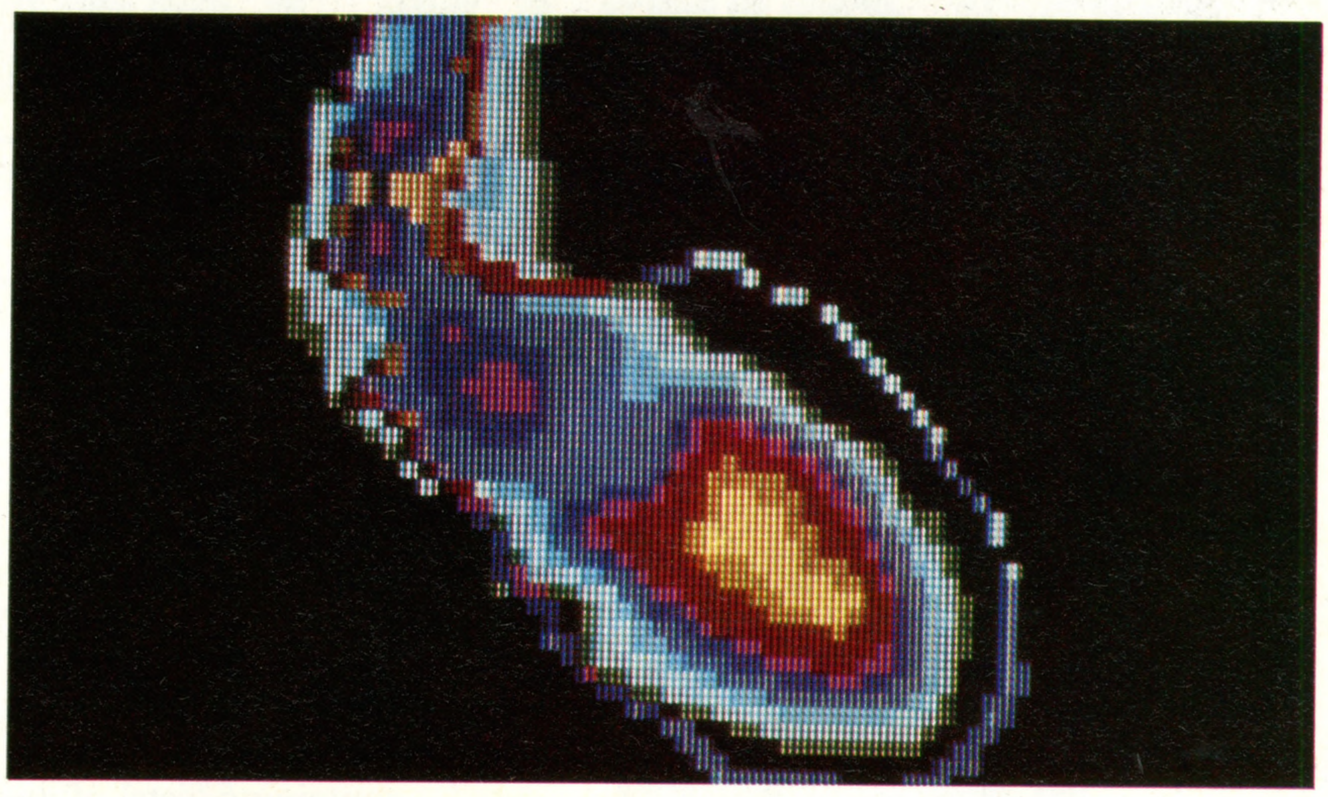


IAN DOBBIE

advances in this field are quite tremendous. For example, imagine the joy of a mute paraplegic at being able to 'speak' for the first time, via an eye-following input device and speech-synthesising output.

Eye following devices measure the movement the eyeball makes as it scans along a line. From a reference point, it is possible to infer the current position, and hence the character being read.

Speech synthesiser chips contain in ROM the 'building bricks' of speech, called phonemes.



IAN DOBBIE

Safer By Far
Nuclear magnetic resonance scans, like the one shown here, produce similar results to computerised tomography scans, but because they do not use X-rays are considerably safer. At the moment, the technique is in the experimental stage. Each scan can take up to an hour

A Picture Of Health
Information on the patient's condition, obtained by sensors attached to the body, is interpreted by a microprocessor, and is then displayed on a specially modified oscilloscope, similar to those used by electronics engineers

Individual phonemes are strung together under program control to make words.

If a severely handicapped person has any degree of movement at all, it can be exploited to form the basis of a computerised communications system. The most popular approach is to replace the standard keyboard with a custom-built unit. It can have very large keys, for example, a benefit to multiple sclerosis and locomotor ataxy sufferers, who have difficulty in locating objects. Or perhaps it could be built to require no pressure.

Alternatively, the machine can present the user with a pre-determined range of prompts at a set rate, for example, a menu for the user's next meal. By pressing a switch at the appropriate point the user tells the computer what he wants to eat. The computer assumes a negative answer if there is no response. In this case the 'keyboard' will be no more than a single switch, accessible by means, perhaps, of a movement of the whole head.

At the least inventive, this approach enables the user to use a word processing package, albeit slowly, and thus produce written work — a tremendous advance for spastics or thalidomide victims.

The Laws Of Thought

A century before the electronic computer was invented, George Boole published his ideas on mathematical logic

COURTESY OF THE ROYAL SOCIETY

George Boole 1815-1864
George Boole was born before electronic computers were ever imagined yet he is one of the founders of the mathematical logic used by today's computers. He was the son of a cobbler and taught himself mathematics in his spare time. He was convinced that the everyday decisions that people make were based on reason, and that this reason could be refined into mathematical logic. He published his ideas in 1847 and almost overnight became famous and was invited to become the first professor of mathematics at the newly founded University of Cork

In 1815, the year of Napoleon's defeat at the Battle of Waterloo, another significant event in the history of mankind occurred.

George Boole was born in Lincoln, England. The son of a cobbler, he was to become one of the geniuses behind the invention of the computer. Although he died in 1864, a century before the microcomputer revolution began, the modern computer could not have evolved without his ideas.

Boole knew that the processes of reasoning that people carry out in their everyday lives can be described in terms of the formal logic pioneered by the Greeks. He believed that if you tried hard enough you could go further and express human reasoning in a mathematical way. Boole set out to do just this; he taught himself mathematics and began his research into the logic of human decisions.

Sets Of Information

Imagine one night you go to a party. You want to dance so you find the room where people are dancing and look for a partner.

The people in the room are either dancing or they are not dancing — they cannot be doing both. The partner you approach must be either a boy or a girl. Obviously, a person can be male or female but not both.

Now Boole would have made a different approach. He would have seen a dance floor containing 'sets' of people, the male set and the female set, or M and F. Boole would also have seen D and W, or the set of people dancing and the set of people waiting to dance.

His partner would have to satisfy two conditions: be a female and also be waiting to dance. Boole noticed the importance of the 'and' connecting the two conditions and gave it a symbol — an upside down ∪. He was then able to list the set of possible dance partners as F ∩ W.

However, if he did not want to dance but just chatter to a friend, he could choose anyone from M or F because these two sets include everyone in the room. Again, he saw the importance of the innocent looking 'or' in the condition and gave it the symbol ∪. So in his logical algebra M∪F contains all the males and all the females in the room.

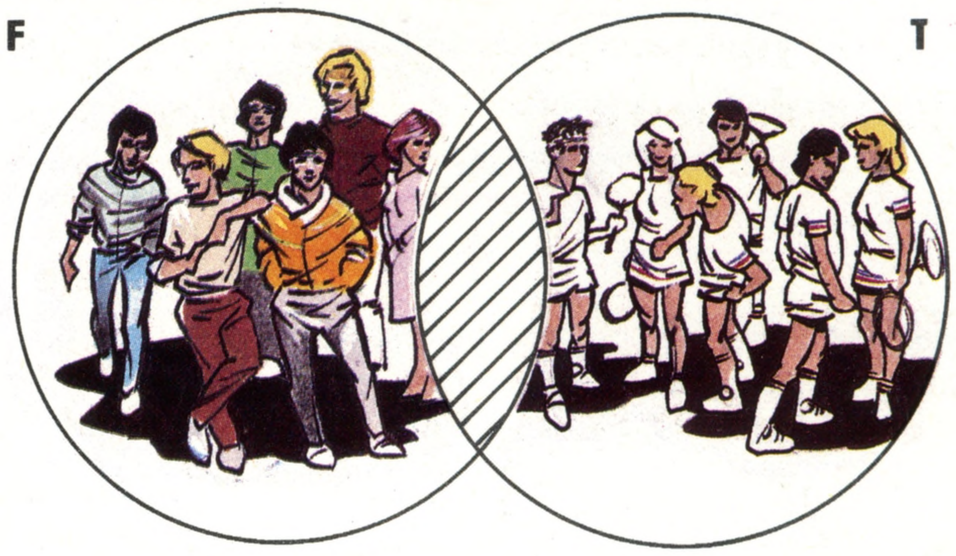
The logical gates found in computers are named after Boole's symbols, such as AND and OR. In BASIC programming we will soon discover two useful commands called AND and OR. But before we pursue these there is a very picturesque interpretation of Boolean logic — invented by the English mathematicians John Venn (1834–1923) and Charles Dodgson (1832–1892), better known as the author Lewis Carroll.

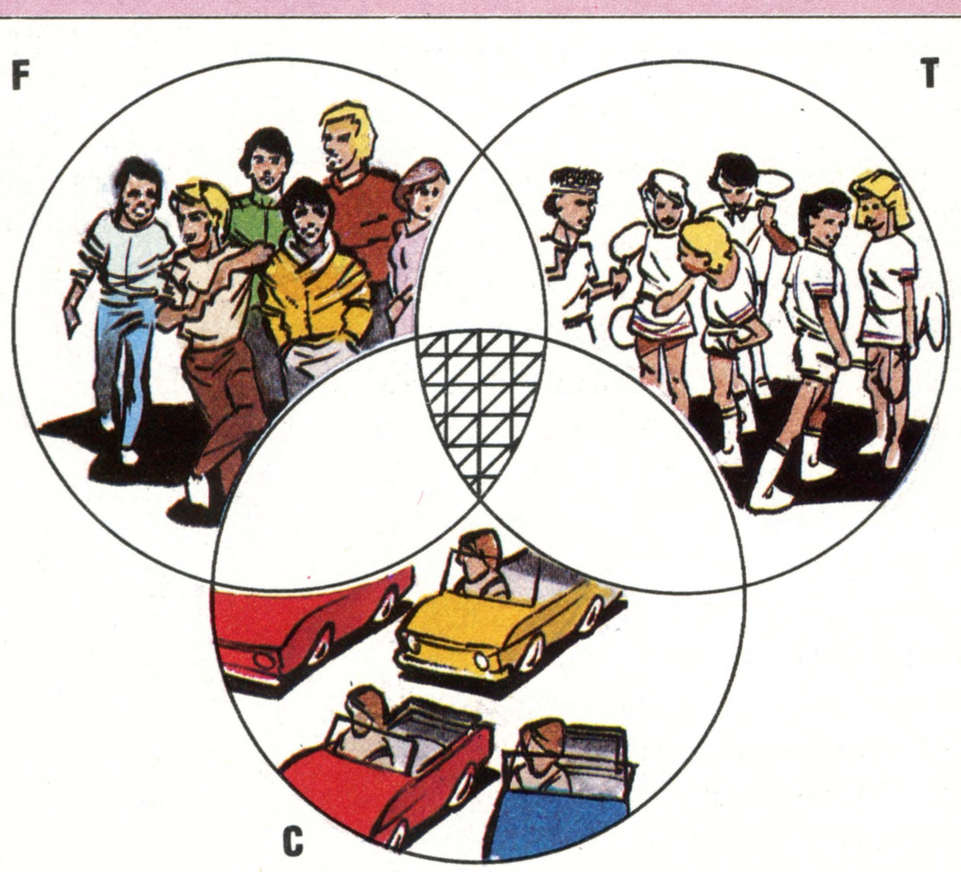
Let us take a practical problem. Imagine you have stored a list of your acquaintances in the memory of your computer. With each name is listed other information — such as the telephone

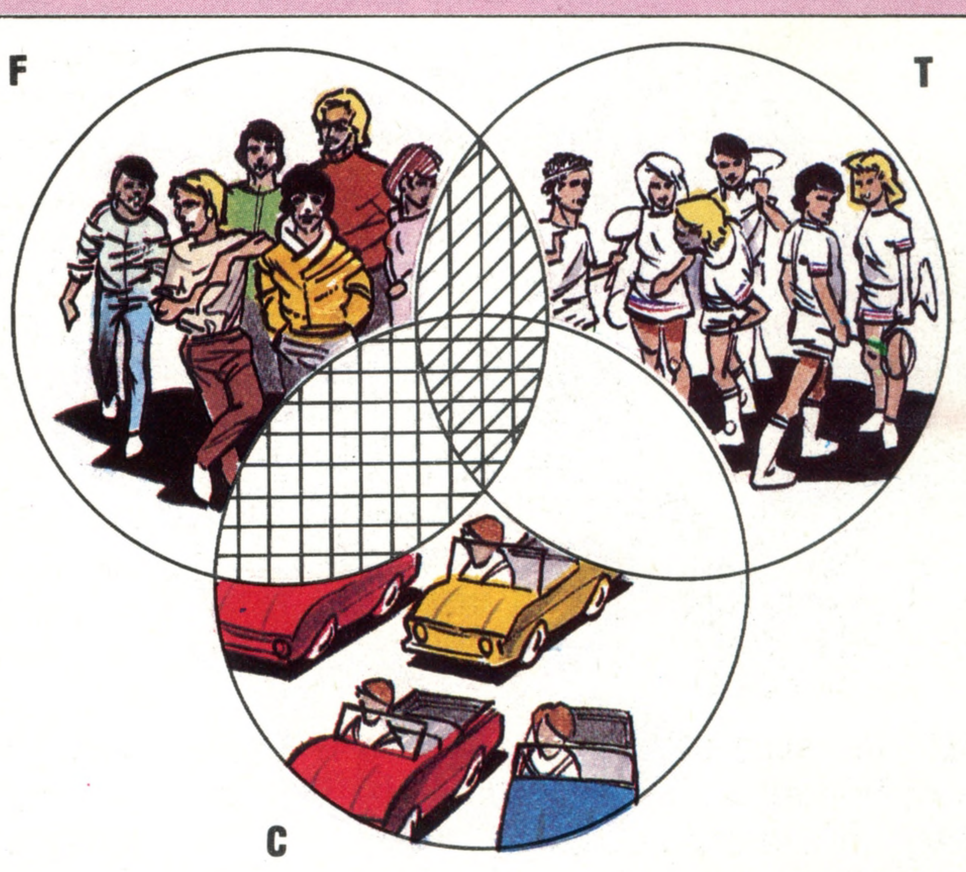
The Friends' Program

```
10 DIM N$(10),D$(10),F$(10),T$(10),C$(10)
15 REM NAME, TEL. NO., FRIEND?,TENNIS?,CAR?
17 PRINT "ENTER DETAILS IN THE FORM:"
18 PRINT "NAME, PHONE, YES/NO, YES/NO, YES/NO"
20 FOR K = 1 TO 10
30 INPUT N$(K),D$(K),F$(K),T$(K),C$(K)
40 NEXT K
45 REM SEARCH ROUTINE
50 FOR J = 1 TO 10
60 IF F$(J)="YES" AND T$(J)="YES" AND C$(J)="YES" THEN GOSUB 100
70 NEXT J
80 END
100 PRINT N$(J),D$(J)
110 RETURN
```

Venn Diagrams







The square box represents the collection, or set, of all the people that are listed in the computer. Generally in Venn diagrams this is called the universal set. The circles within the square represent the individual sets. The friends are identified on the diagram as those people who lie within set F.

Not all the people will play tennis, but those who do are identified in set T. Since there will be some people who both play tennis and are friends, the two circles overlap (F∩T).

Set C identifies those people who own a car. This set overlaps the other two and the people who satisfy all three conditions are contained in the area where all three circles intersect (F ∩ T ∩ C). The last diagram represents the same sets, (F,T,C,) but the conditions that need to be satisfied have changed. You have decided that you would like either to play tennis or go for a drive. The set of acquaintances who are friends and play tennis is shaded with diagonal lines. The set of acquaintances who are friends and own a car is shaded with horizontal lines. The combination of these two shaded areas represents the friends with whom you could play tennis or go for a drive

KEVIN JONES

number, hobbies, etc. One afternoon you decide you want to play tennis at a court on the other side of town. You need a friend (as opposed to an acquaintance) who plays tennis and owns a car. Your computer is instructed to print out the name and telephone number of every person who satisfies all three conditions: plays tennis AND owns a car AND is a friend. The program on the left first asks for information about each acquaintance: are they friends, do they have a car, do they play tennis? It is assumed that you have 10 acquaintances but you can change the number of acquaintances to any you want (remembering to change the 10 in the bracket of the DIM statement in line 10). The list is then scanned using an IF . . . THEN command into which has been inserted a multiple condition. Most BASICS allow the IF . . . THEN command to work on a condition made up of separate subconditions joined by the commands AND and OR. Finally, the name and telephone number of any acquaintance who satisfies the condition — that they are a friend and play tennis and have a car — is printed.

Very complex combinations of logical functions are found in some programs. Boole's algebra, which was little more than a curiosity in his own lifetime, has come into its own in the age of the computer.

Dragon 32

A Welsh computer boasting a new chip and a set of sophisticated graphics

Dragon Data, as the name and logo suggest, is a Welsh company, backed by investors such as the Welsh Development Agency and the Prudential Group but originally founded by toymakers Mettoy.

The Dragon 32 was introduced for Christmas 1982 and achieved instant success because of its full 32 Kbytes of RAM and its Microsoft BASIC interpreter.

The Dragon 32 is highly compatible with the Tandy Color Computer: it is possible to use Tandy add-ons and some game cartridges, but not the cassettes. The two machines use the same microprocessor, the Motorola 6809E (see box), where most home computers favour the 6502 or Z80.

A set of sophisticated graphics commands gives greater control than that offered by many machines with better maximum resolution. Examples of such commands are: DRAW, CIRCLE, PAINT, COLOUR and MOVE.

The Dragon can only play one note at a time, as distinct from computers with more than one 'voice', which can generate chords. But the BASIC commands available make it far easier than on most machines to produce a recognisable tune.

Though well-endowed with interfaces, there are few expansion devices for the Dragon 32 apart from joysticks.

The upgraded version of the Dragon incorporates 64 Kbytes of RAM, and thus offers the user the possibility of small business applications, putting the Dragon into competition with machines such as the Commodore 64.

CHRIS STEVENS

Dragon Disk Drive
This unit comes in two parts: a disk controller card that plugs into the Dragon's cartridge slot, and the main box, which contains one disk drive and costs £275

Keyboard
The Dragon 32 has a typewriter-style keyboard.

The cursor movement keys for up/down and left/right are inconveniently situated at opposite ends of the keyboard. The BREAK key interrupts the operation of a program and CLEAR wipes the screen.

Although the design suggests that a small television might be placed on top of the casing, this is not recommended

Reset Button
Pressing this has the same effect as switching the computer off and on again but results in less wear on the power supply

Cassette Port
An ordinary cassette recorder can be plugged in here, and Dragon 32 BASIC allows control of the cassette's motor

Joystick Ports
The Dragon 32 can cope with two joysticks for playing games

Printer Port
A standard eight-bit parallel interface allows the Dragon 32 to work with most makes of printer

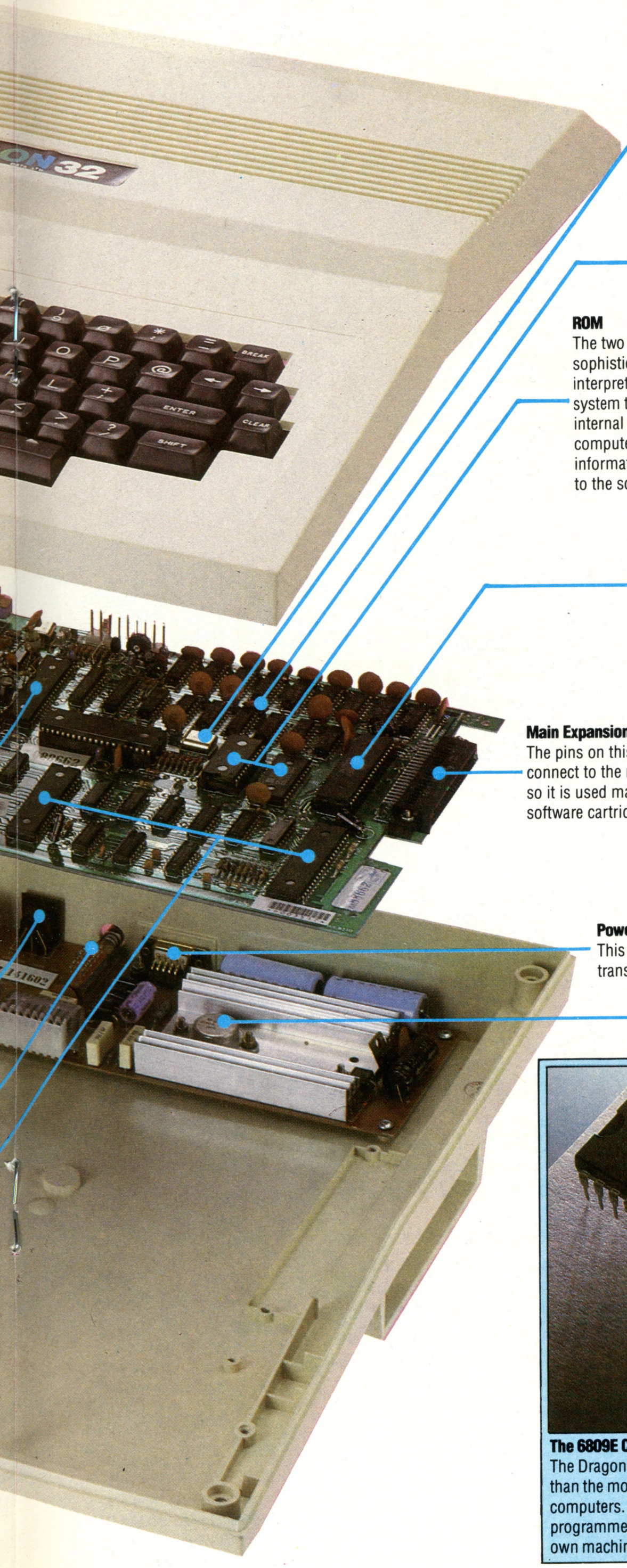
Video Controller
This chip generates the video signals required by a television or monitor screen from the characters and graphic symbols stored in memory

RF Modulator
Converts the video signals to a form suitable for input to the aerial socket of a television set

Video Interface
Separate red, green and blue signals are provided here for driving the appropriate monitor. This system yields the best quality results

On/Off Power Switch

Interface Chips
Two 6821 Peripheral Interface Adaptors (PIAs) handle all the necessary conversion of signals from the CPU to the keyboard, cassette recorder and external devices such as the printer



Quartz Crystals
Quartz crystals provide timing signals so that all parts of the computer operate in synchronisation. One controls the speed of the microprocessor, the other primarily the video circuitry

RAM
The Dragon 32's standard 32K of RAM comes in 16 2K chips

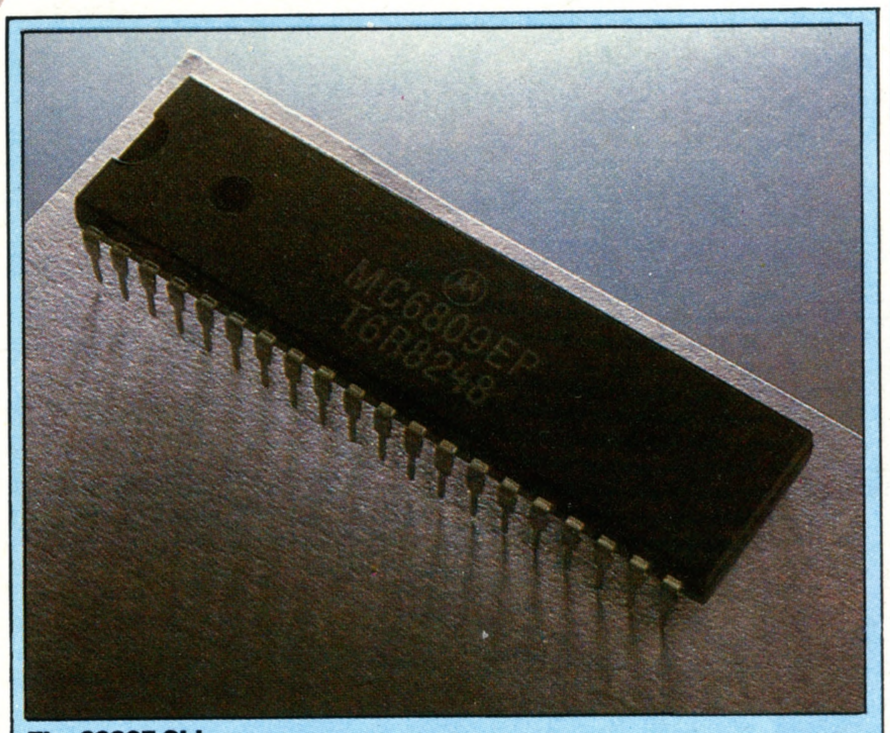
ROM
The two ROM chips contain the sophisticated Microsoft BASIC interpreter and the operating system that governs all the internal functions of the computer, such as transferring information from the keyboard to the screen

Microprocessor
The Motorola 6809E is faster than the more popular 6502, though there is less software written for it. The same chip is used in the Tandy Color Computer — one reason why both machines are so compatible

Main Expansion Port
The pins on this interface connect to the microprocessor, so it is used mainly for plug-in software cartridges

Voltage Regulators
An external transformer reduces the mains voltage to the appropriate voltage used inside the computer. These regulators, with the cylindrical capacitors in an adjacent position, are needed to smooth out any variations or fluctuations. They are mounted on a large metal heat sink to avoid overheating

Power Socket
This is where the mains transformer plugs in



The 6809E Chip
The Dragon 32 uses the 6809E eight-bit microprocessor rather than the more popular 6502 or Z80 chips used in most computers. The 6809E is highly regarded by many programmers. It is easy to write programs for the 6809E using its own machine language

DRAGON 32

PRICE
£175

SIZE
380 × 325 × 97mm

WEIGHT
2.1 kg

CLOCK SPEED
1 MHz

MEMORY
The 32K of RAM in the Dragon 32 represents quite a substantial amount of memory power, complemented by 16K of ROM containing the Microsoft BASIC interpreter and operating system

VIDEO DISPLAY
The screen can hold 16 lines of 32 character positions. A total of eight colours is available, reducing to two colours at the maximum graphics resolution of 256 × 192 dots

INTERFACES
Cassette, joysticks (two ports), television monitor, cartridge slot, parallel interface for printer, etc.

LANGUAGE SUPPLIED
BASIC

OTHER LANGUAGES AVAILABLE
None

COMES WITH
Power supply unit, aerial lead, instruction manual

KEYBOARD
Typewriter-style layout with 53 moving keys

DOCUMENTATION
The user's manual, written by the computer's designer, Richard Wadman, has been produced to a high standard.

All responses from the computer are printed in a different colour, and there are plenty of helpful hints and tips.

Dragon 32 BASIC is taught from an elementary level to the level of comprehensive graphics and sound. This is followed by a set of appendices including helpful screen layout grids. These are used to create graphic designs before programming.

There is a good choice of books available about the Dragon 32, both instructional courses on programming and listings of programs

Staying In Focus

You can now buy a dedicated screen for your computer, which will give you high quality display for graphics and games

As more and more people begin to use computers, videos, and other equipment which needs screens, the prices of specialised monitors are starting to fall.

Originally a good colour monitor would cost six or seven hundred pounds, but nowadays it is possible to buy one for about £350, or even less. Monochrome monitors are much cheaper, typically in the £100 to £150 range.

With the ever-expanding graphic capabilities of microcomputers, most of which have colour, it is a very good idea to get a colour monitor.

There are two main types of colour monitor, one type being known as RGB (standing for Red — Green — Blue), and the other as composite video. An RGB monitor is controlled directly, with the three guns which actually produce the colours being turned on and off by the computer. The pulses which are used to synchronise the computer with the monitor are also produced directly by the computer.

There are two types of sync-pulse, one for each line of the picture, and one for each complete picture. At the end of each frame, the monitor is sent a short pulse, which tells it that the frame is now complete, and that the electron-beam (and thus the dot which it produces) must be returned to the top left-hand corner of the frame.

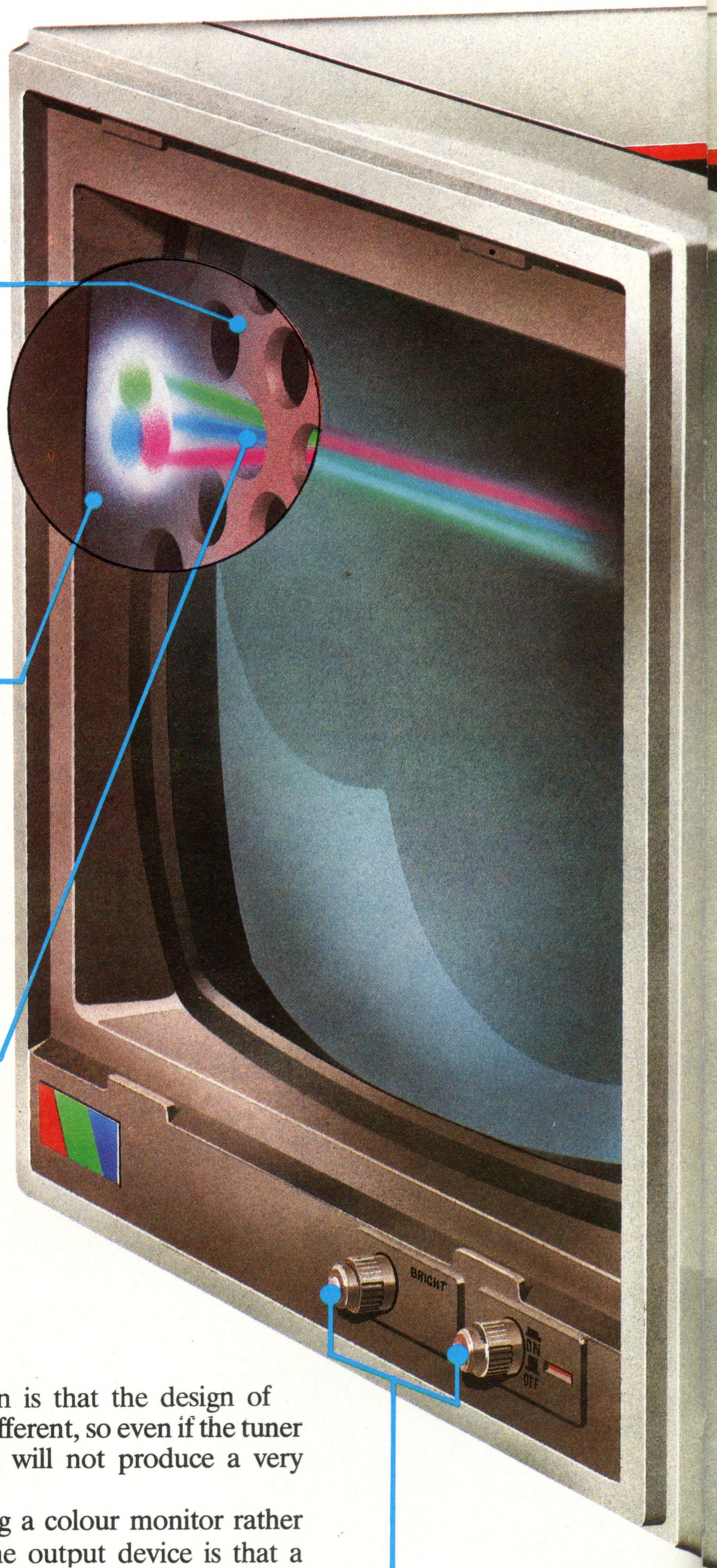
A similar process occurs at the end of each line, indicating that this particular line is complete, and that the electron-beam must be returned to the left-hand side of the screen, ready for the next line.

In an RGB monitor, each of these signals (Red gun, Green gun, Blue gun, line-sync and frame-sync) is sent to the monitor down its own individual wire.

A composite monitor, on the other hand, is a closer relation to a television, since all the signals are combined into one, then sent to the monitor via a co-axial cable. Once inside the monitor, the line-sync, frame-sync and the three colour signals are separated again and used to control the image.

A monitor is a television without a tuner. In fact, it's possible to turn a monitor into a television by adding a tuner, or to modify an ordinary television by taking the channel selection mechanism out.

However, this is definitely not recommended, because there are dangerously high voltages inside any piece of equipment that contains a cathode ray tube. Even professional technicians would approach this problem with trepidation.



Screen Grid
To ensure that the electron guns are pointed at exactly the right place on the screen, a grid or mask is incorporated into the surface of the tube

Screen Phosphor
The coloured image is made up (as shown in the diagram) of three colours. Different substances are laid on the glass. When irradiated by the electron beam, they glow in either red, green or blue, thus giving the coloured image, depending on the intensity of the beam at that point

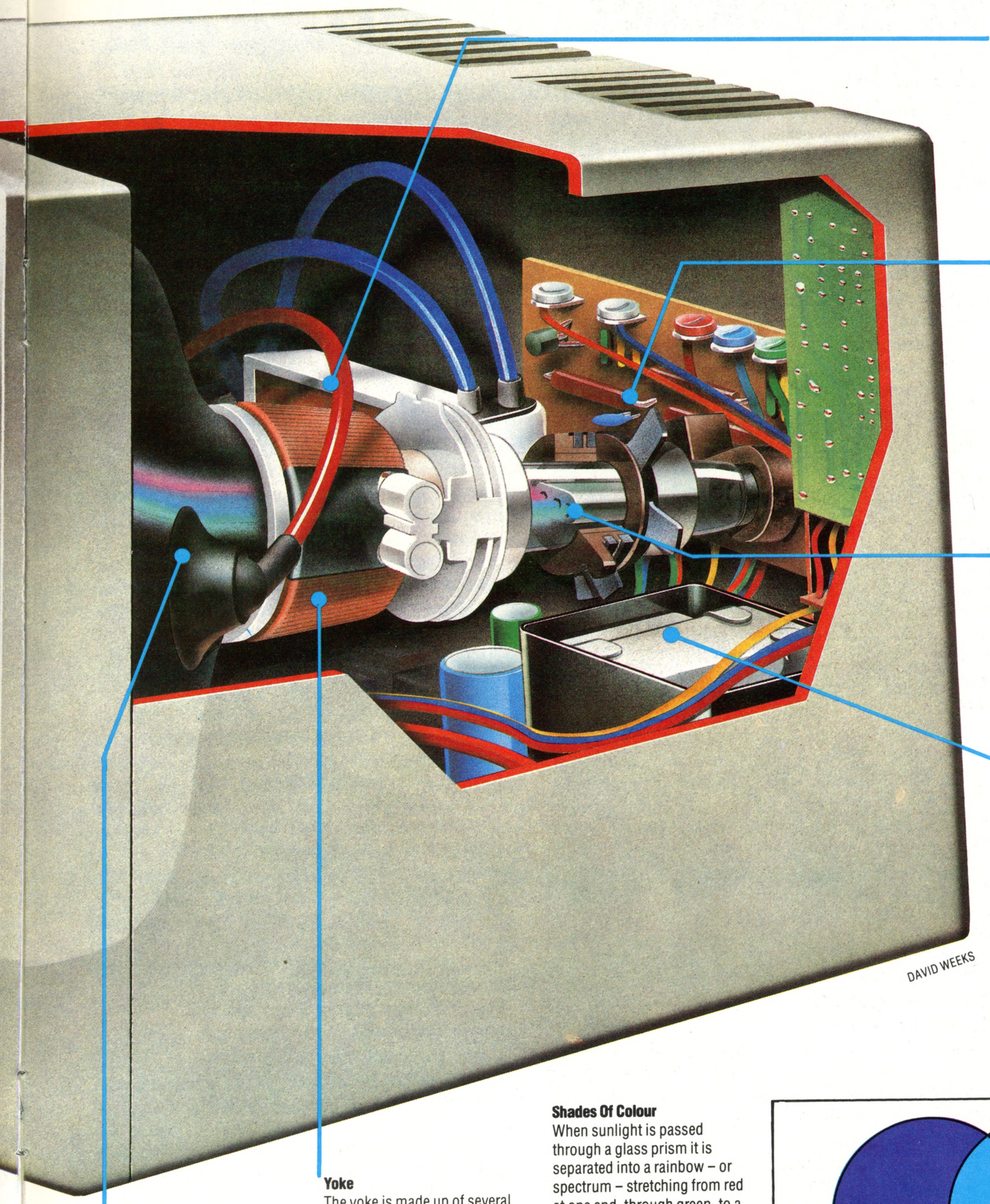
Electron Beams
There are three electron beams in the tube, each of which 'excites' a different phosphor to produce a coloured dot

Controls
As on a television, there are various controls. Vertical and horizontal hold are commonly accessible to the user. Colour intensity and other variables are usually not meant to be adjusted and are kept under the cover

The other main reason is that the design of the circuits is slightly different, so even if the tuner is removed, the result will not produce a very good monitor.

The reason for using a colour monitor rather than a television as the output device is that a television will only work with a signal which is overlaid with an Ultra-High Frequency carrier-wave, from which we get the acronym UHF. This means that the nice clean signal generated by the computer has to be encoded, sent down the wire, and then decoded again. Doing this results in a 'messy' signal, which gives a fuzzy picture.

A monitor, on the other hand, does not need the same modulation and de-modulation of the signal, and thus produces a cleaner, sharper picture. This is much easier on the eye, and makes your programs look much more professional.



High Tension Circuits
Because cathode ray tubes need very high voltages, they must have a rectification circuit to boost the mains voltage (240v) to the required level

Main Circuit Board
The circuits necessary to produce the controlling currents which move the beam and turn the guns on or off are found here. Part of the line synchronisation section, which works at very high frequencies, can be utilised as a switching-power supply for the tube itself

Guns
A colour monitor, like a colour television, has three colour guns, red, green and blue

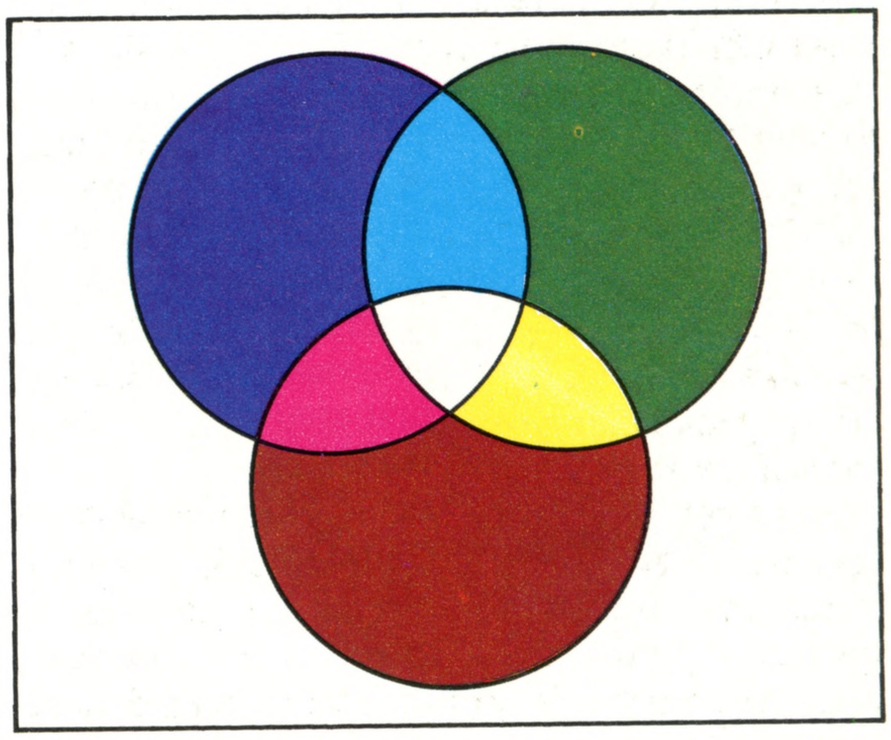
Power Supply
A cathode ray tube must be driven by very stable DC voltages, and requires fairly heavy currents, so a large transformer is needed

DAVID WEEKS

Yoke
The yoke is made up of several large coils that produce powerful magnetic fields. These vary rapidly, so that the dot on the phosphor is moved about, producing the picture

Anode Attachment
Once the beam is ejected from the guns, it is accelerated by the high voltage field. This must be at the other end of the tube and is applied by means of this large, heavily insulated plate, which is on the end of the cable

Shades Of Colour
When sunlight is passed through a glass prism it is separated into a rainbow – or spectrum – stretching from red at one end, through green, to a blue-violet at the other. If this spectrum is then passed through another prism, similar to the first, the colours recombine to give the original sunlight (often called 'white light'). This process of recombination or addition is used in a colour monitor. By adding different strengths of the three main colours, red, green blue, every colour can be created.



MARK WATKINSON

Organise Your Program

We sort a program using built-in functions to rearrange information

This week's program illustrates how relatively complex programs can be broken down into simple sub-programs or subroutines that can be written and tested separately.

Apart from the advantage of being separately testable, the use of subroutines allows the development of the program to follow a logical progression. There are many approaches to writing a program in BASIC. One of the commonest is 'trial and error': you sit down at the computer and start entering lines of BASIC without having thought out carefully what is required to make the program work. This method leads to badly structured programs that will often not work the first time. If the structure of the program is not clear, it is not easy to find the mistakes or 'bugs'.

A much better approach is to sit down with a notebook and work out the structure of the program first, in steps of ever-increasing refinement, until a correct and working program can be written. A flow diagram will also help (see page 104). Let's see how this is done.

Problem: Write a program that will input a number of names, the forename followed by surname. Now reverse the order of each name so that the surname comes first, followed by a comma and a space, and by the forename. It will then sort the names into alphabetical order and print them out.

For example, if the names BILL JONES and FRED ASHTON were entered (in that order), the program would print out:

ASHTON, FRED
JONES, BILL

Before even attempting to write a program to do this, first write down the desired input and output in the most general terms:

Step 1
Input names in random order, first name first
Output names in alphabetical order, surnames first

This simply clarifies what we want to be done. This is an essential first step to a properly organised program. The next step is to refine the stages in the first step and make sure that the program still works. Do not, at this stage, get into too much detail. Simply write down in a little more detail, the stages involved:

Step 2
Find out number of names to be input
Enter names
Reverse names
Sort names
Print names

Look at the above list and check that it will work. Can you see anything wrong with it? Are there any flaws in the logic? If not, you are ready to go on to the next stage of refinement.

The stages we arrived at in Step 2 are small enough and simple enough to be written separately as small sub-programs. Sub-programs are called subroutines in BASIC. Let's give the subroutines names to make them easier to identify. Subroutine 1, to find out the number of names to be input, can be called FINDNUM. Subroutine 2, to enter the names, can be called ENTER. Subroutine 3, to reverse the names, can be called REVERSE. Subroutine 4, to sort the names, can be called SORT. Finally, Subroutine 5, to print the names can be called PRINTNAMES.

Step 3.1 FINDNUM
Prompt the user to input required number
Get the required number N
Use N to set up string array
Step 3. 2 ENTER
If number of names is less than N,
prompt user to input another name
Add name to string array
Step 3.3 REVERSE
Find length of string (name)
Find 'space' in string
Put characters in string up to 'space'
into temporary string variable
Put characters in string from 'space' to
end into another temporary variable
Add comma space to end of variable
Assign second followed by first
temporary variables to original string
Step 3. 4 SORT
Compare first item in array with next item
If first item is bigger than next one
(i.e. higher in the alphabet), swap
Compare second item with third
Swap, if necessary
Repeat until all pairs are compared
Go back to beginning of array and
repeat comparison of pairs until no swaps
have taken place

Note: This sort routine is exactly the same as the one used in the previous part of the Basic Programming course. The 'swap' part will be dealt with as a subroutine called from within the SORT subroutine.

Step 3.5 PRINTNAMES

Print each item in the array until all items have been printed

Each of the steps needed to build this program has now been worked out in a reasonable amount of detail. The SORT routine has only been sketched roughly since it was dealt with fully in the last instalment of the course. And SWAP, which is 'called' from within this subroutine, has been left out completely. Let's now see how easy it is to convert programs worked out in English into a program in BASIC.

Step 4
1. FINDNUM

The three lines in Step 3.1 translate directly into BASIC statements. The user is prompted by a PRINT statement, the number is found by using an INPUT statement and the array is dimensioned by using the DIM statement:

```
PRINT "HOW MANY NAMES DO YOU WISH TO
  ENTER?"
INPUT N
DIM A$ (N)
RETURN
```

The variable N now contains the maximum number of names to be entered. The DIM statement dimensions a string array. String variables contain strings of alphanumeric characters instead of numbers. A string variable name always ends with a 'dollar' sign. A$ alone could only contain one string. DIM A$ (N) creates an array that can contain 'N' strings. Subscripted variables have been dealt with earlier in the course.

The RETURN statement transfers control back to the main program at the line following the subroutine call. Values assigned to variables in the subroutine will be 'carried back' to the main program and can be used elsewhere in the program, even in other subroutines.

2. ENTER

As long as the number of names entered is less than N, the user needs to be prompted to enter a name and this name must be added to the string array. This calls for creating a FOR-NEXT loop; we know that the first name in the array will be its first element, and that the last one will be the Nth, so:

```
FOR X = 1 TO N
PRINT "ENTER NAME"
INPUT A$(X)
NEXT X
RETURN
```

That should suffice to enter all the names into the array. But sharp readers will have spotted what happens when we come to reverse the order of the first and last names in the REVERSE subroutine. Each element (name) in the array will have to be pulled out again, then reversed, and then put back in the array. Rather than complicate and lengthen the program by doing that, it would be simpler to call the REVERSE subroutine from within the ENTER subroutine after each name has been typed in. The name can then be reversed before it is assigned to the array. To do that, we just have to add one line, thus:

```
FOR X = 1 TO N
PRINT "ENTER NAME"
INPUT A$(X)
GOSUB [REVERSE]
NEXT X
RETURN
```

All the names in the array will now be in reversed order (surname first, followed by forename) and will therefore be ready for sorting.

3. REVERSE

To reverse the order of names, we need to know where the 'space' is separating the first name from the surname. When we know where the space is, we can use various functions to pull out parts of the string and assign those parts to other strings. Functions in BASIC are commands that perform a predefined operation on the value following the function name. This part is always in brackets. Many functions are 'built in' but it is also possible to define your own. A typical 'built in' function is SQR ( ). This function 'returns' the square root of the value inside the brackets. So: LET A = SQR (9): PRINT A will print a 3.

REVERSE uses the functions LEN (to find the length of the string), INSTR (to locate the position of the space), LEFT$ (to remove a specified number of characters from the left of the string) and

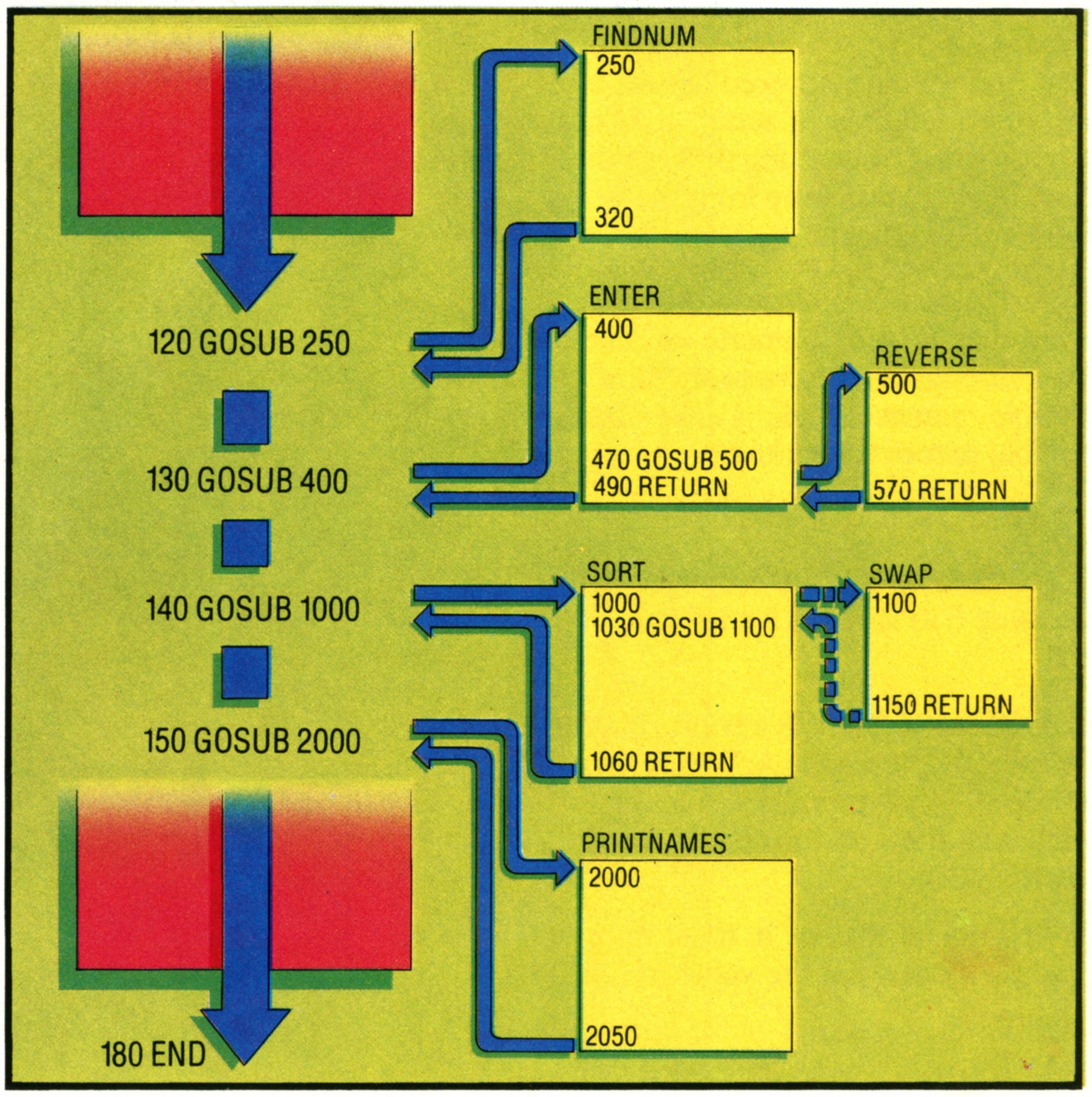
Programs Within A Program

The main program this time is very short. All the real work is done in the sub-programs (called subroutines in BASIC). Each of the steps needed to make the program work are separated and written as short 'mini-programs'. These are then simply linked together by the main program.

When the program is run, each time a GOSUB statement is encountered, the program branches to the specified subroutine line number and that section of the program is then executed. The end of the subroutine is indicated by the RETURN statement. On reaching this, the program returns to the point immediately after the GOSUB that called the subroutine.

Subroutines can be 'nested' within subroutines. The ENTER subroutine calls another subroutine called REVERSE, and SORT sometimes calls another subroutine called SWAP.

Breaking down a problem into separate subroutines linked by a simple main program makes the development and testing of programs far easier



TONY LODGE

RIGHT$ (to remove a specified number of characters from the right of the string). We will not discuss in detail exactly how these functions work at the moment. We will take a more comprehensive look at functions in BASIC in the next part of the course.

4. SORT

SORT, and the SWAP subroutine called from within it, follow closely the routines used last time.

5. PRINT NAMES

This is very straightforward:

```
FOR Q = 1 TO N
PRINT A$(Q)
NEXT Q
RETURN
```

Now all that remains is to write the main program. It's as simple as this:

```
REM MAIN PROGRAM
GOSUB [FINDNUM]
GOSUB [ENTER]
GOSUB [SORT]
GOSUB [PRINT]
END
```

We have put the 'names' of the subroutines in square brackets. A few BASICS are able to call subroutines by name, but most have to use line numbers. When the program is actually written out, the appropriate line numbers are inserted in place of the subroutine names. Appropriate REMs and PRINT messages are also added.

Exercises

Now that we have covered almost all of the most important features of BASIC, it is time to check your progress by working through these exercises. They range in difficulty from the very easy to the moderately difficult.

■ **Variables** Put a circle around the expressions below that are valid numeric variables, and draw a cross through the expressions that are not valid variable names at all. Leave the valid string variable names unmarked.

A B6 2Z D$ 15 X$ A12 D9 Q81 Q5 6F H$

■ **Arithmetic 1** Write a short program to assign the value 6 to variable B and then PRINT the value of B.

■ **Arithmetic 2** Write a short program to assign the value 5 to variable A, 7 to variable B and 9 to variable C. Add the values of these three variables and assign the sum to variable D. PRINT the value of variable D.

■ **Arithmetic 3** Look at these lines of BASIC and then work out what the value of C will be.

```
LET C = 5 + 4 * 3
PRINT C
```

```
10 REM  THIS PROGRAM SORTS NAMES
20 REM  INTO ALPHABETICAL ORDER
30 PRINT "FIRST DECIDE HOW MANY"
40 PRINT "NAMES YOU WANT TO ENTER"
50 PRINT "THEN ENTER THE NAMES IN"
60 PRINT "FIRSTNAME(SPACE)LASTNAME"
70 PRINT "ORDER."
80 REM
90 REM  THIS IS THE MAIN PROGRAM
100 PRINT
110 PRINT
120 GOSUB 250
130 GOSUB 400
140 GOSUB 1000
150 GOSUB 2000
160 REM
170 REM  END OF MAIN PROGRAM
180 END
250 REM  SUBROUTINE TO FIND NO. OF
260 REM  NAMES TO BE ENTERED
270 PRINT "HOW MANY NAMES DO YOU"
280 PRINT "WISH TO ENTER?"
290 PRINT
300 INPUT N
310 DIM A$(N)
320 RETURN
400 REM  SUBROUTINE TO ENTER NAMES
410 PRINT "ENTER NAME IN THIS FORM:"
420 PRINT "FIRSTNAME(SPACE)LASTNAME(CR)"
430 PRINT "E.G. JILL THOMPSON"
440 FOR X = 1 TO N
450 PRINT "ENTER NAME"
460 INPUT A$(X)
470 GOSUB 500
480 NEXT X
490 RETURN
500 REM  SUBROUTINE TO REVERSE ORDER OF NAMES
510 LET L = LEN(A$(X))
520 LET S = INSTR(A$(X)," ")
530 LET C$ = LEFT$(A$(X),S - 1)
540 LET F$ = RIGHT$(A$(X),L - S)
550 LET F$ = F$ + ", "
560 LET A$(X) = F$ + C$
570 RETURN
1000 REM  SORT ROUTINE
1010 LET S = 0
1020 FOR P = 1 TO N - 1
1030 IF A$(P) > A$(P + 1) THEN GOSUB 1100
1040 NEXT P
1050 IF S = 1 THEN GOTO 1000
1060 RETURN
1100 REM  SWAP SUBROUTINE
1110 LET T$ = A$(P)
1120 LET A$(P) = A$(P + 1)
1130 LET A$(P + 1) = T$
1140 LET S = 1
1150 RETURN
2000 REM  PRINT SUBROUTINE
2010 PRINT
2020 FOR Q = 1 TO N
2030 PRINT A$(Q)
2040 NEXT Q
2050 RETURN
```

■ **Arithmetic 4** What result will be printed in this program?

```
LET A = 3
LET B = 2
LET C = 9
LET D = 4
LET E = (A + B) * (C - D)
PRINT E
```

```
LET E = 5
LET E = E * E
PRINT E
```

■ **Comparisons 1** What value of X will be required for the PRINT message to be printed?

```
70 LET A = 5
80 LET B = X
90 LET R = B - A
100 IF R = 0 THEN GOTO 120
110 GOTO 10
120 PRINT "CONGRATULATIONS! YOU HAVE WON"
999 END
```

■ **Comparisons 2** What is the smallest value of X that will make the program jump to line 300?

```
250 IF X > 6 * 100 THEN GOTO 300
```

■ **Comparisons 3** What is the smallest value of Z that will make the program jump to the 'congratulations' message?

```
340 IF Z < 10000 THEN GOTO 500
350 IF Z >= 10000 THEN GOTO 520
:
:
500 PRINT "YOUR SCORE IS TOO LOW. TRY AGAIN"
510 GOTO 600
520 PRINT "CONGRATULATIONS. YOU ARE NOW A
MASTER"
530 GOTO 700
```

■ **Print 1** Assume that the value of T is 50. Write a PRINT statement that will print THE VALUE OF T IS 50. Hint: Put the 'message' in double quotes, use a semi-colon and the variable name.

■ **Print 2** Look at the following short program and complete the PRINT statement so that the program will print a score message like this:

```
SORRY, BUT YOUR SCORE OF 175 IS TOO LOW
```

Complete the line so that the actual value of the score can vary each time.

```
620 REM VARIABLE S IS THE SCORE SO FAR
620 IF S <= 500 THEN GOTO 640
630 GOTO 700
640 PRINT "SORRY"
```

■ **Print 3** What message will be printed when the program is run?

```
200 LET A$= "THE HOME COMPUTER COURSE ?"
210 LET B$ = "HOW DO YOU LIKE ";
220 PRINT B$
230 PRINT A$
```

■ **Input 1** INPUT is one way of assigning a value to a variable. If the following program is run, which key will need to be pressed for the program to print out an answer of 12?

```
60 INPUT N
70 LET N = N * 2
80 PRINT N
```

■ **Input 2** What will be printed here?

```
100 PRINT "PLEASE TYPE YOUR NAME"
110 INPUT N$
120 PRINT "HI ";N$;"I'M YOUR COMPUTER"
```

Basic Flavours

This program will not run on the Atari 400 and 800 because their string handling is so different from that of other machines

DIM

On the ZX81 and Spectrum replace line 310 by:
310 DIM A$ (N,30)
This creates a string array called A$ that has N elements, each of them 30 characters long.
On the Lynx replace line 310 by:
310 DIM A$ (30) (N)

GOTO

In line 1050 the command GOTO 1000 comes immediately after the word THEN. In this case, most computers allow you to omit the word GOTO; so line 1050 might be written
1050 IF S=1 THEN 1000

INSTR

This command is not available on the ZX81, Spectrum, Commodore 64, Vic-20 and Oric-1.
On the Commodore machines and the Oric-1 delete line 520 and replace it by:

```
515 LET S= 0
520 FOR P=1 TO L
523 IF MID$(A$(X),P,1)=" " THEN LET S=P
524 IF S=P THEN LET P=L
525 NEXT L
```

On the Spectrum and ZX81 delete lines 510 to 560, and replace them by:

```
510 LET D$=A$(X)
520 LET L=LEN D$
530 LET S =0
540 FOR P=1 TO 1
550 IF D$(P)=" " THEN LET S=P
560 IF S=P THEN LET P=L
570 NEXT P
580 LET C$=D$( TO S-1)
590 LET F$=D$(L-S)
600 LET A$(X)=F$+", "+C$
610 RETURN
```

LEFT$

RIGHT$

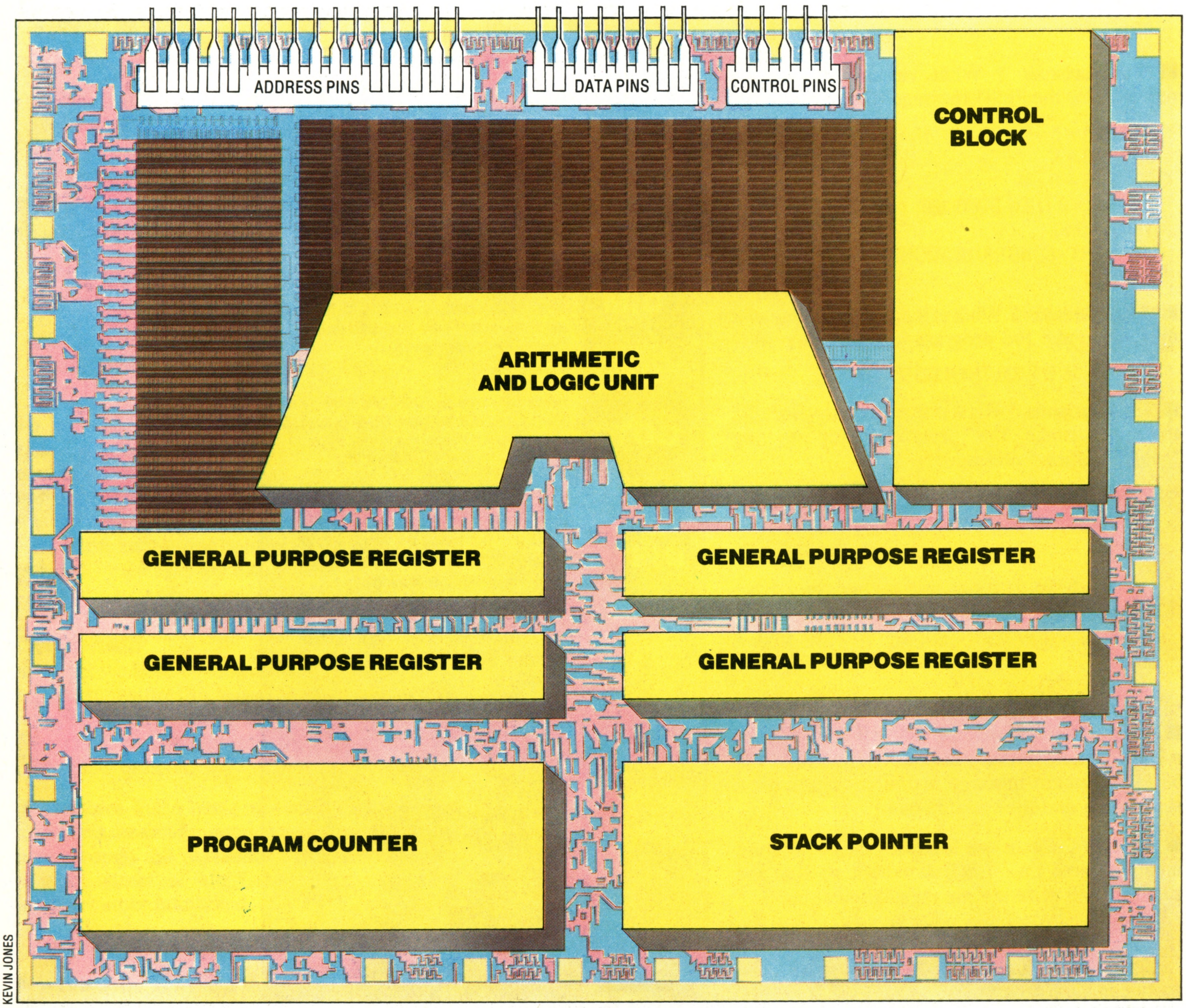
MID$

None of these commands is available on the Spectrum or ZX81. Their equivalents in Sinclair BASIC are:

LEFT$(Z$,N)	replace by Z$(TO N)
RIGHT$(Z$,N)	replace by Z$(LEN(Z$)−N+1 TO)
MID$(Z$,P,N)	replace by Z$(P TO P+N−1)

The Nerve Centre

All paths of activity lead in and out of the computer's 'Central Processing Unit'



KEVIN JONES

The computer's function is controlled by the CPU — the Central Processing Unit. The words can be interpreted fairly literally: Central — at the heart of the computer; Processing – it does the work; Unit – it stands alone. A very simple computer (see illustration) could consist of nothing more than a CPU, some memory and some I/O (input/output) circuitry.

The I/O is needed for the computer to communicate with the outside world. In a very simple application, a computer in a washing machine would need I/O in order to turn on motors and heaters. The memory is needed to store instructions for the CPU and data for the CPU to process. The data processed by the CPU may include numbers and binary codes that represent characters (letters, digits and signs such as @ and !).

If some of the memory locations contain instructions to the CPU, and some are data for the CPU to process, how does it know which ones are instructions and which ones are data? To answer this question, we need to take a look inside the microcomputer.

CPUs in 8-bit microcomputers (this includes almost all small home computers) are usually packaged in a single chip with 40 pins, 20 on each of the long sides. Every one of these pins (apart from the 0 and the +5v power supply pins) carries signals into and out of the CPU from other devices such as memory or I/O circuits.

Typical 8-bit CPUs have 16 'address' pins. These pins are connected to the 'address bus'. Each of these pins carries an output signal representing either a one or a zero. Between them, they can have 65,536 different combinations of ones and zeros. They are used to select specific memory locations.

There are also eight 'data' pins, which are connected to the 'data bus'. The data pins carry data into the CPU from memory or I/O, or data from the CPU to memory or I/O.

A number of other pins carry 'control' signals. Some of these signals are outputs from the CPU and others are inputs. We will see how the control signals are used shortly.

Inside the CPU there are a few small one-byte or two-byte memory cells called registers, some of which are reserved for special purposes. The others are used for the temporary storage of information and are called general purpose registers. There are two other important functional 'blocks' in the CPU, the ALU and the 'control block'.

The abbreviation ALU stands for Arithmetic and Logic Unit. This part of the CPU performs arithmetical and logical operations, including (but not limited to) adding, ANDing, ORing and moving bits to the left or right within a byte.

The control block is a special circuit designed to make the CPU behave in accordance with the instruction received from memory. We can take a specific example, using the instruction codes for the popular Z80 CPU. If the coded instruction 11000110 is received from memory, the CPU will add the contents of the next byte in memory to the contents of one of the registers inside the CPU. If we then want to store the result of that addition in a specified memory location, the next instruction received by the CPU will have to be 00110010, followed by two bytes specifying the actual location in memory where we want the result to be stored.

Suppose that the result of the addition was 37 (using decimal notation), and that the two bytes following the instruction specified address location 33126 (again using decimal notation). The instruction code would cause the control block to set the address pins to the binary equivalent of 33126 (in binary this is 1000000101100110). It would then cause the control pins to send out signals telling the memory to expect some data and that the data must be stored (memorised). It would then cause the data pins to be set to the binary equivalent of 37 (in binary this is 00100101). This data would then pass down the data bus to the memory and would be stored in the memory location specified by the address bus. If, at some later stage, the CPU needed this data to process in some other way (to print on the screen, for example), a different instruction could be sent to the CPU. The control block would interpret this as meaning 'address memory location 33126, get the byte from it and store it temporarily in one of the internal registers'.

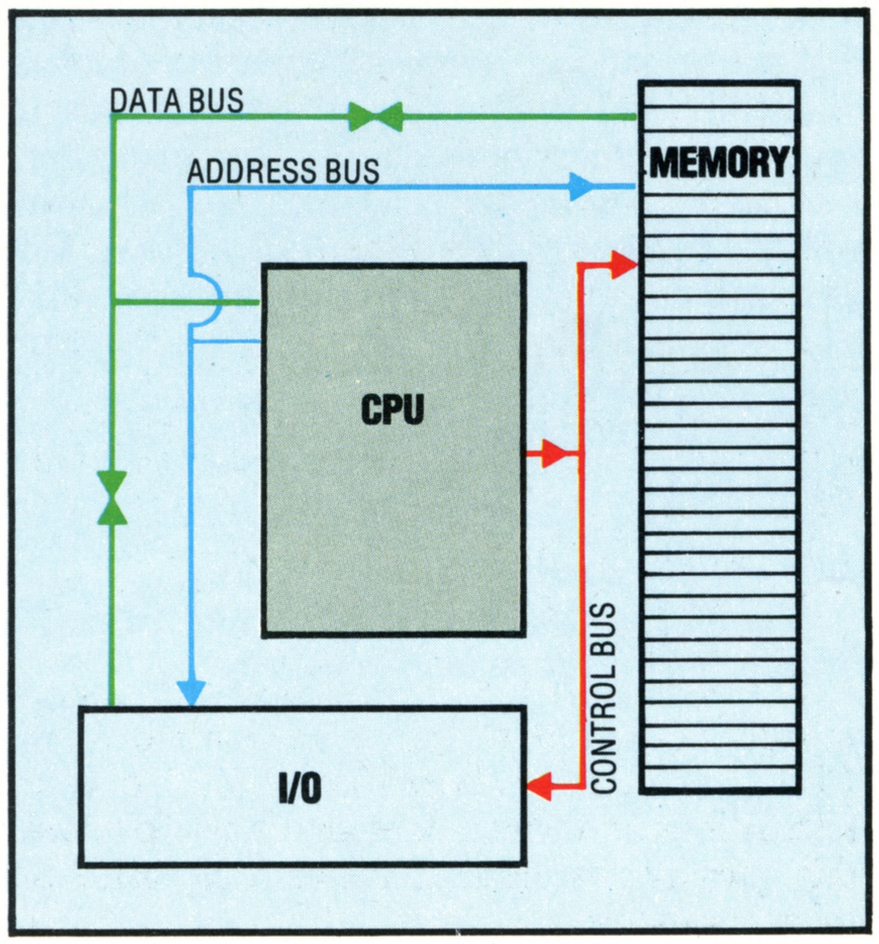
The number of registers or temporary memory cells inside the CPU depends on the CPU. They will either be eight-bit (one byte) registers or 16-bit (two byte) registers. The specialised registers are usually given special names such as the 'stack pointer', 'program counter' or 'accumulator'. The general registers are usually referred to as 'the X register', 'the Y register', 'the C register' and so on.

One of the 16-bit registers, and one of the most important, will be the 'program counter' register. This internal memory cell always contains the address (in binary) of the next instruction in memory due to be executed. When the time comes to get the next instruction for the CPU, the contents of the program counter will be put on the address bus, and the byte at the location will be transmitted (via the data bus) to the CPU.

The most important of the eight-bit registers is the 'accumulator'. This is the register that usually stores (temporarily) the result of operations performed by the ALU, bytes brought in from memory or I/O, or the place where bytes are temporarily stored immediately prior to being sent out to memory or I/O.

This introduction to CPUs has been very general; specific points will be covered in detail later. The aim has been to show that special instructions read in from memory cause the CPU to perform specified operations and to set the address pins to access particular memory locations. Data is fetched from these locations, or sent to them, over the data bus. The instructions also cause the control bus signals to indicate to memory or I/O whether data is to be 'read' or 'written'.

Under Control
The illustration shows a CPU with memory 'registers', an Arithmetic and Logic Unit comprising hundreds of logic gates (that perform such operations as adding, ANDing and complementing binary numbers), and a control block. The control block accepts the coded instruction (in binary), interprets it, and causes all the other parts of the CPU to behave appropriately. For example, if an instruction means that the contents of the accumulator should be stored in a particular memory location, the control block will put the address on the address pins, send control signals telling memory to store data, and put the contents of the accumulator on the data bus for transmission to the memory



Following Orders
A very simple computer may consist of nothing more than a CPU, some memory and an I/O circuit. The memory will store special instructions that make the CPU perform specified actions. It will also store data for the CPU to process in accordance with the instructions. The I/O circuit will be needed for the CPU to communicate with the outside world. If the computer is controlling a washing machine, the I/O circuit will input signals from the buttons on the front panel and output signals to switch motors and heaters on and off.

The instruction codes for the CPU will be in binary. Each different model of CPU has its own set of instruction codes

John Von Neumann

This brilliant mathematician lends his name to the design of modern computers

ASSOCIATED PRESS

Complementary Couple
John von Neumann with his second wife Klara, herself an accomplished programmer of early computers

Only a Hungarian could enter a revolving door behind you and emerge ahead of you. So said John von Neumann in describing the competitiveness of his fellow countrymen.

He was no exception. His own ambition coupled with an extraordinary intelligence took him to the highest scientific posts in the United States.

Neumann was born into a wealthy Jewish family of the Austro-Hungarian Empire. His mathematical skills were spotted while he was still quite young, and by the age of 25, he had collected two degrees and a doctorate and was discussing mathematical problems on an equal footing with such eminent men as Einstein and the mathematician David Hilbert.

Neumann was never blind to worldly problems. With the collapse of the Austro-Hungarian Empire after the First World War, he adopted the title of 'von' and slipped into the academic life of defeated Germany. At the same time he was building up his American contacts, spending the winters at Princeton University, in New Jersey, and the summers in Europe administering his father's estates.

When the Second World War broke out he was already safely established in America.

Von Neumann made his name in mathematics by patching together the theory of sets, which Bertrand Russell had undermined with his logical paradoxes. Von Neumann was fascinated by quantum physics and also the theory of games. He invented the Monte Carlo method, which uses random numbers to solve mathematical equations.

When the war came to America he was immediately brought into the Manhattan Project where he enthusiastically joined in the production of the atomic bomb. While stationed at Los Alamos he would often drive 120 miles to eat at his favourite Mexican restaurant and in his later years at Princeton it was said that he destroyed a car a year by his wild driving.

He was still engaged in the Manhattan Project when he learned that attempts were being made to build an electronic computer and he had himself invited onto the ENIAC project. The work was under the control of electronic engineers but as the first mathematician involved he saw the problem differently and drew up a report that was to become the blueprint for the modern computer.

After the war he became increasingly involved with the defence policy of the United States. But he remained committed to mathematical research and designed the first computer for Princeton University. It was called JOHNIAC. At a party to celebrate its completion he had a model of it carved in ice.

In his 50's he developed cancer and was subsequently confined to a wheelchair.

Although he had been an agnostic all his life, he became a Catholic in his last months.

A friend said about his death: 'I think von Neumann suffered more when his mind no longer functioned than I have ever seen any human being suffer before.'

In his honour the design of computers is known to this day as von Neumann architecture.

1903
Janos Louis Neumann born in Budapest 28 December

1921
First mathematical paper published in collaboration with his tutor

1933
Becomes a professor at Princeton University, New Jersey, the haven for Einstein and other emigré mathematicians from Europe

1942
Publishes a book on the application of Games Theory to economics

1943
Appointed consultant to the Manhattan Project at Los Alamos, New Mexico

1944
Learns about the secret ENIAC project

1945
Attends first atomic bomb tests

1947
Report on ENIAC and draft designs for a new computer to be called EDVAC (Electronic Discrete Variable Computer) to be built using these proposals

1951
JOHNIAC – his personally designed computer becomes operational at Princeton University

1951-1953
President of the American Mathematical Society. Works on automata theory

1955
Appointed US Atomic Energy Commissioner by President Eisenhower

1957
Dies of cancer on 8 February. As a special honour a whole volume of the 'Bulletin of the American Mathematical Society' is devoted to his life and work

Mentathlete

Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair

THE HOME COMPUTER COURSE BINDER

Now that your collection of Home Computer Course is growing, it makes sound sense to take advantage of this opportunity to order the two specially designed Home Computer Course binders.

The binders have been commissioned to store all the issues in this 24 part series.

At the end of the course the two volume binder set will prove invaluable in converting your copies of this unique series into a permanent work of reference.

Buy two together and save £1.00

✻ Buy volumes 1 and 2 together for £6.90 (including P&P). Simply fill in the order form and these will be forwarded to you with our invoice.

✻ If you prefer to buy the binders separately please send us your cheque/postal order for £3.95 (including P&P). We will send you volume 1 only. Then you may order volume 2 in the same way – when it suits you!

Overseas readers: This binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in Issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain their binders **now**. For details please see inside the front cover.

Binders may be subject to import duty and/or local tax.

THE LAST WORD IN LOGIC