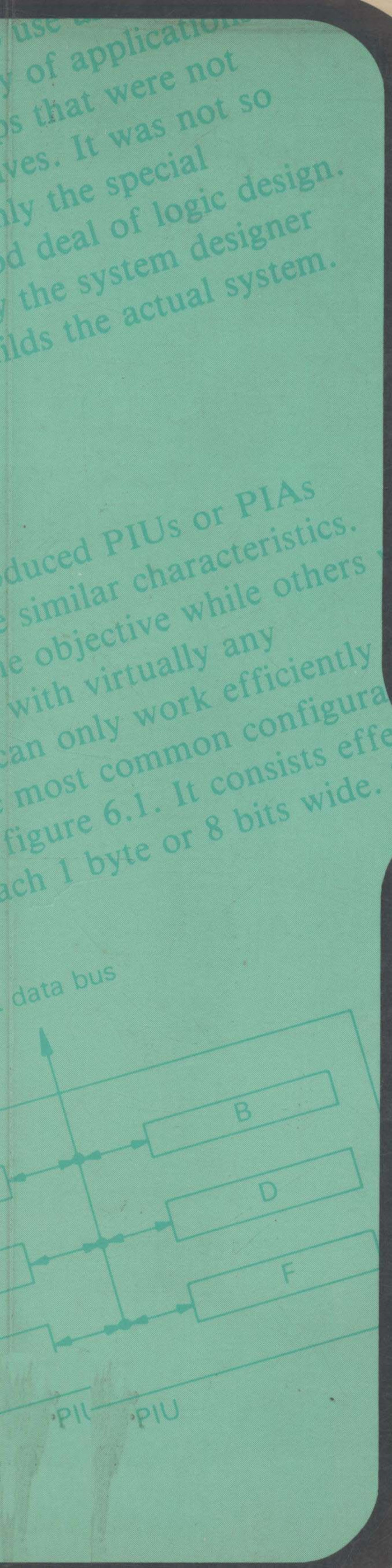

Understanding Microprocessors

B S Walker

Understanding Microprocessors

B. S. Walker

Department of Cybernetics, University of Reading

M

First published 1982 by
THE MACMILLAN PRESS LTD
London and Basingstoke
Companies and representatives
throughout the world

Typeset by Reproduction Drawings Ltd., Sutton, Surrey.

Printed in Great Britain
by Unwin Brothers Limited
The Gresham Press, Old Woking, Surrey

ISBN 0 333 32309 2

Preface

Electronic digital computers and electronic data processors are two names for the same devices. The latter is a better name since it more nearly describes what they do. Microprocessors are physically very small electronic data processors. Although they have similar capabilities to their much larger brethren they are fabricated on a single chip of silicon or sapphire. They must be made in huge quantities to be economical and since they are so made they are cheap. A few years ago it cost a lot of money or academic attainment to join the quite select club of computer users, whereas today anybody can own one or even several. Before long many people, who will have no idea that they are computer users, will own several. They are becoming integral parts in car electrical systems, in domestic automation such as washing machines and they already form the basis of the proliferating numbers of hand-held and desk calculators, electronic TV games and the like.

It is the purpose of this book to try to explain what microprocessors do and how they do it, in so simple a way that people who do not regard themselves as technically inclined can still understand them and develop informed ideas about what the microprocessor may well be able to do and also what it may not do.

For those who are technically minded it is hoped that this book will point the way for them to develop their skills in the highly significant area of automatic systems under microprocessor control.

The principles by which data processors or computers work are elegantly simple—like the wheel—when once understood. It is difficult to imagine why such a simple concept took so long to become obvious. What makes them seem mysterious or difficult to comprehend is that they do things—very simple things—at speeds that we, as humans, find to be incredible because our concepts of speed and time are related to bicycles, cars or aeroplanes but not electrons.

Computers seem to do very complicated things because they do so many simple things so quickly. Yet they are in many respects sluggish compared with the human nervous system, which we take for granted. Even the most advanced man-made control systems are crude by comparison with those in nature. What seem to us quite simple things (because we do not look into them in great detail) like catching a ball, or chewing and swallowing food, or writing a letter, when examined in precise detail do become much more complicated. Processors are artefacts and they need to be dealt with in great detail in many applications; for this reason they too seem complicated. Their designers and builders need to go into this detail, but their users need not.

In order to develop an understanding of automatic computers and data processors the easy approach is to examine them and what they do first in a fairly cursory way and then to examine them in increasingly fine detail—but only as far as we wish, or need, or as our curiosity impels us. This will be the approach in this book. Readers are invited to read just so far as they may feel impelled. Having read all of it, they may continue further, as far as they can possibly go, to the limits of human expertise, if they are sufficiently interested and determined.

The understanding of computer or microprocessor principles does not primarily depend on detailed knowledge of electronics, logic design, binary arithmetic or the processes by which microcircuits are made. This knowledge becomes increasingly necessary, however, in the design and implementation stage of practical systems. For this reason the main text has been written with the minimum stress on these topics. The three appendixes following, however, do go into more detail for the reader who feels the need.

Finally there is a problem of terminology—sometimes called jargon. Like every other technology, that of computers and microprocessors does contain words and expressions which imply specific concepts and which it would be unrealistic and tedious to avoid. Computer technologists generally have taken pains to use ordinary words with their normally understood meanings. Whenever computer terminology is introduced in the text, an explanation is given.

Contents

1 *Introduction to Data Processors*

The concept of a processor is a familiar one. It is a contrivance into which we put some material and it operates on what we put in, producing at its output some modified, hopefully improved, form of the input material. For instance a meat canning factory is a kind of processor, as is a mill for making paper or grinding corn into flour. More than a century ago, Charles Babbage built a 'calculating engine'; he called the 'number crunching' part of his engine the 'mill'.

A data processor fits happily into this same concept of a processor. *Data* is a Latin word and means 'things given'. The term, as used today, means 'abstract things', 'items of information expressed in symbols'. We feed data into an electronic data processor and it operates on the given things, or their symbols; it outputs other modified data in a more useful form.

A typical example of a data processing operation is the preparation of the payroll for the work-force of a factory. The symbols representing the names of the employees are entered as data, together with other data relating to their hourly rates and hours worked, all as symbols. The processor operates on this data and in due course prints out the pay slips for the employees describing, in symbols, their pay, tax deductions, stoppages, bonuses, etc.

A different kind of process might be to read in data describing the behaviour of a reaction in atomic physics and to put out data describing the result, perhaps in symbols or perhaps in the form of a graph. Another different process might be used in a library to keep track of book loans or to find references. Different again, a processor may be used to fly an aeroplane. Here the data is not translated in the first place into discrete symbols but is supplied in some more continuous and convenient form. The processor compares data relating to present speed, course, altitude, attitude, etc., with other data provided by the pilot and sends out data to the aircraft controls which brings the actual values into coincidence with the desired values.

The key to the versatility of the data processor is that it contains a variety of data processing elements. Some of the data which goes in is called the program: this data is itself used to control the activities of the elements, how and when they are used and in what order. The data to be processed is then read in and operated on within the machine; the output is due to both.

The program has to specify in fine detail what the processor does to the processed data. Writing about Babbage's engine, Countess Ada Lovelace (Byron's daughter) remarked that 'the machine can do only that which we know how to order it to do'; what she wrote is equally applicable to the most modern data processors.

Figure 1.1 illustrates crudely the basic organisation of a data processor.

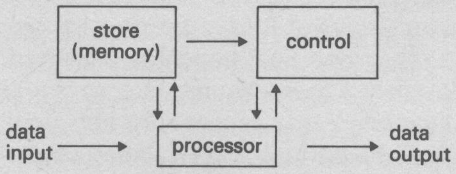

Figure 1.1

Suppose we consider the data processing problem of sorting a mixed-up list of numbered files into a serially numbered list: such a task might be the production of a telephone directory to be listed in number order from an alphabetically ordered one. The processor illustrated in figure 1.1 could perform the job in the following way.

First of all a program would have to be prepared and entered (read in) to the processor. It would necessarily be conceived and written by a human. It would consist of a list of rather simple instructions for the processor to obey in a clearly defined sequence.

Expressing the operation of the program very simply: it would call in the first items of the data to be sorted and store them also in memory. It would then compare the numbers in the first two files, storing the larger as the start of a new list, and keeping the smaller. It would then take in the next file and compare the number in this one with the one it was holding. The larger of these two it would place on the new list, keeping the smaller for the next comparison, and so on. When it had scanned all the original list of data it would repeat the process but on the new list, forming another. It would keep on repeating this process until a scan of the whole list entailed no swopping.

There are many ways in which a 'sort' of this kind can be done by a processor; the one described is neither very elegant nor efficient, but would have advantages in certain circumstances. It does, however, illustrate some important points.

First, it illustrates how the use of a very simple procedure repeated many times can perform a complicated process. If it is done by electronic means at great speed, then even such an inelegant repetitive process can be performed quite quickly measured against the human time scale.

Secondly, it illustrates an important aspect of programming such processes. Quite a simple routine, or list of instructions, would be required to perform the comparison and list. This same short routine would be written once but used over and over again. The same would be the case for the routines for reading in each item, for switching from list to list, for testing the number of comparisons and, finally, for printing out the result.

Thirdly, it illustrates the functions and requirements of various parts of the system. The 'input' would be required to handle the volume of input data at a sufficient speed to supply the processor. The 'memory store' would need to hold the program and some or all of the listed data for some or all of the time. The program, once entered would remain unchanged throughout the process. The data would be continually changing. The 'control' would have to be able to decode and give effect to the program instructions.

Finally the output device would need to print out the symbols which formed the resulting list.

Obviously, careful preconsideration of the problem could make the process more efficient. For instance, the numbers on which the comparisons are performed may well be only a small fragment of each file. It could well be possible for the majority of each file to bypass the processor itself or perhaps not enter the system at all.

THE MICROPROCESSOR SYSTEM

For many people the term 'microprocessor' has come to mean not just the device itself but the whole assembly of things that go with it to form a microprocessor-based system. Hence, to these people, the idea of the microelectronic magic chip is somewhat ahead of what the technology has yet been able to achieve. Most microcomputers and microprocessor-based systems consist of a considerable number of chips and an even greater number and volume of other devices used to maintain, control and harness them.

The term microprocessor, accurately used, means only the *central processing unit* (CPU) of the whole system and contains the following.

(1) The *arithmetic and logic unit* (ALU) which does the 'number crunching', the logical operations, comparisons and the like, on data fed into it.

(2) A group of highly specialised memory storage elements which are used in the ALU activities. These are commonly termed the *central registers*. The term register has the same conceptual meaning in the computer context as in the world outside it: it is a thing which holds (and sometimes displays) a number, or piece of symbolised data.

(3) A *control unit* which has two main functions. The first is to fetch instructions from memory and decode them in the proper order. The second is to execute them, that is to control the ALU and its attendant registers and, when required, fetch its data (*operands* is the technical term). It also has to do a host of less obvious things which are often referred to as 'house-keeping', like refreshing the contents of storage elements, counting operations internally and sequencing what are called 'microinstructions'. The 'simple' operations of the ALU are broken down into even more elementary ones by the designer and formed as 'microprograms' of strings of 'microinstructions'.

The whole CPU assembly is fabricated on a chip of silicon with an area of a few square millimetres. The chip is then stuck to a substrate of ceramic or similar material and connected to a set of terminals by gold wires that are very thin indeed. The terminals are rigidly held to the substrate and the whole assembly is encapsulated to make it sufficiently robust for humans to handle with reasonable care. It does not require much imagination nor technical knowledge to appreciate that 'pin-out' is a limiting constraint on microprocessors. Pin-out is the arrangement of terminals on the encapsulated package, joined by the gold wires to the chip. It is extremely difficult to make microcircuit capsules with 40 terminals, though some do exist with as many as 64. Since the CPU has a great number of things to do and has to receive and output all the data that is processed, much ingenuity is required to make everything possible through the limited number of connections. This is one of the principal differences of microprocessors from larger data processors in which the provision of a few or even a lot of extra connections poses no insuperable problems.

The processor of figure 1.1 points to some of the other things needed for a microprocessor system. The CPU contains the 'control' and 'processor' boxes. The next main item that is needed is the memory or store. The main memory of a microprocessor system is also formed of chips much like the CPU but having very many more elements of fewer varieties. The amount of microcircuit store attached to the CPU depends on the tasks the system must perform, on the pocket of the purchaser and on the pin-out. As we shall see later in more detail, the maximum practicable store size for direct access by the CPU is about 64 thousand (64 k) words or *bytes*, that is, 8-bit patterns or store locations. A memory store is organised like apartments in a housing block. Each location capable of holding a unit of data has an 'address'. To put data in or get it out first we must specify to the store controller the address and then send the data into, or read it out of, that address. There are a variety of different types of microcircuit store and these form a topic for consideration later in this book. But generally even for small systems there will need to be, at this stage of the technology, several chips, or capsules, of memory store.

Figure 1.1 vaguely indicates 'input' and 'output': these labels cover a multitude of devices and the chips to control them. Generally they are referred to as *peripherals* or *peripheral devices* since conceptually they are attached to the periphery of the system. Among other things, they include paper and magnetic tape data transmitters and receivers, teletypes and line printers, keyboards, cathode ray tube monitors for TV-type presentation, data converters of various kinds, lamps, buttons, switches, 'floppy discs', magnetic and punched card readers—and so the list goes on. More generally these are the contrivances which translate humanly comprehensible data into machine data and vice versa, thus providing the 'man-machine interface'.

Finally, not shown, but important nevertheless, there need to be power supplies, cooling fans and similar unromantic provisions. The microprocessor chip when harnessed is usually only a tiny component in a cabinet or rack of electronic and electromechanical devices.

The system cannot operate without a program. This is a list of instructions to the processor detailing every processing action and the order in which these operations are to be performed. In a dedicated system the program may be held in a *read only memory* (ROM), in which case the system should be able to operate with a minimum of external control, perhaps merely a switch or button to set it to run or stop. In order to prepare such a ROM there will have to be a development stage, however, in which the system under development has to be run and tested and the program debugged (that is, corrected where necessary). It is very unlikely that even the simplest program will be written correct in every detail in the first place. Often, in 'real-time' system design, the exact timing and similar details cannot be correctly forecast before the system is run and adjustments must be made experimentally. In these cases the program must be held in a read/write memory until all its details are completely tested and corrected.

In microprocessor systems the read/write memory is usually also made up of microcircuits; it is called *random access memory* (RAM). This somewhat misleading name merely means that the data held anywhere within the memory is equally accessible, that is, it does not have to be read in a serial sequence; accessing data in a random order entails no time penalty.

Program instructions may be of two kinds, those which require operands to be obtained from memory and those which do not. Those requiring data are called *memory reference instructions* and must contain two parts. The first part defines the function to be performed and the second defines the location in the memory store from which the operand or data is to be obtained. Instructions which are not 'memory reference' are sometimes called *operation instructions*: they must also define a function—this may be to alter the state of the processor or some component of it or to manipulate data that is already held within the processor.

Data on which the processor is to operate is normally held in the RAM. Since the processor operates very quickly, if wanted data is not in RAM, the processor will have to wait for it to be read in through some peripheral device or from a bulk store such as a *floppy disc*, a device which records and reads back data in serial format, storing it on a magnetic film on the surface of a plastic disc. The very rapid growth in the use of microprocessors has led to an equally rapid development of this kind of storage device with the result that the floppy disc has become particularly associated with microprocessor systems.

When the data to be processed is too voluminous to be held economically in RAM, much of the success of the system may depend on the efficient organisation of transfers of blocks of data between the bulk store and RAM, enabling the processor to have available the data it needs without having to wait for it. In larger computer systems, RAM is often called *immediate access storage* and the RAM store the *working store*. Discs and similar bulk storage devices are often called *mass store*, while magnetic tape stores and interchangeable disc units are categorised as *backing store*. Since microprocessors, computers and data processors are so similar in function, terminology applicable to one often finds its way into the literature of the others, often very usefully, but it can be confusing. Microprocessors themselves have already led to the development of a special vocabulary; a number of words and expressions used, however, imply important concepts and we must become familiar with them.

The material operated on by microprocessors is data in the form of symbols. The machine has to be able to interpret and act according to instructions, expressed also as symbols. From the engineering considerations alone, it is necessary that the symbology is the simplest. It is for this reason that computer designers from the earliest days have used a binary notation, that is, restricted to the use of only two symbols which the human writes and interprets as 1, or 0, but which may be interpreted also as logically 'true' or 'false', or electrically as 'on' or 'off'.

From an electronic point of view it is far easier to construct devices which indicate two rather than a multiplicity of states. We do not need to understand much about electronics to understand microprocessors. For this reason all but the necessary minimum amount of electronics has, in this book, been relegated to an appendix, as has the detail of logic elements and other devices and techniques which are used in the engineering implementation of the microprocessor. These affect its performance and physical construction but have little to do with the fundamental principles by which it works.

Bits

The symbol space in which we can record a 0 or 1 is called a *bit*. The *bit* is, to the student of information theory, the minimum quantity of information required to resolve the ambiguity of equiprobable alternatives. The computer technologist uses it more loosely as the abbreviated form of the term 'binary digit'. The microprocessors with which this book is primarily concerned operate on arrays of 8 bits, called 8-bit binary words or more shortly 8-bit *bytes* or, by common usage, just 'bytes'. For example

0 1 0 0 1 1 1 0

is a byte of 8 binary digits or bits and is a pattern of 1s and 0s. Such a pattern can be translated or 'transduced' for the human from some electronic representation into, for instance, a row of lamps, lit denoting '1' and unlit denoting '0'. A similar pattern can be formed by punching holes across a strip of paper tape which could be punched by a teletype and transduced by a photoelectric device into electronic signals which the processor can manipulate or interpret.

Registers

An array of electronic devices can be constructed each of which has the capability of being set to 1 or reset to 0, by an electronic signal, and it remains for the time being in that state. Such an array is called a *register* because it registers a pattern, sometimes having the significance of a number. Registers are a fundamental component of the microprocessor.

Logic Elements

The logic values, 1 or 0, can be transferred from one register to another as electrical signals or voltage levels. These transfers are regulated by logic elements or *gates*. All the fundamental operations of the microprocessor are achieved by these transfers. Data is manipulated or changed by the logic units between registers in a logically defined way. For instance, the patterns contained in two registers can be compared digit by digit by logic elements and the result registered as another pattern of digits. The resultant pattern might indicate that the two original patterns are identical, or that one is numerically larger than the other; it might likewise be arranged that the resultant represents the numerical difference between the contents of the two registers, or their numerical sum.

As with electronics, so also a minimal knowledge of logic and logic elements is required for the understanding of computers and microprocessors. For the reader who lacks this knowledge, appendix B should provide an adequate introduction.

Highways or Buses

Data patterns being transmitted from register to register generally require as many wires or conducting paths as there are bits in a pattern or word. Due to the limitation on the number of terminals to a chip (pin-out) there cannot be many such sets of conducting paths. Thus, particularly in microprocessors, when several registers need to be interconnected, this is done via a common highway or *bus*. The output of one register is gated on to the bus at the same time as is the input of the receiving register. The word bus comes from the Latin *omnibus*, meaning 'for all (things)'—it has become abbreviated by electrical engineers just as it has for public transport users. An 8-bit microprocessor depends heavily on its 8-bit wide, or 8-conductor, data bus to which all its working registers are connected. Necessarily such a system entails the time-sharing of the bus if several transfers are required between different pairs of registers, and this has to be arranged by the designer. Fairly simple data manipulation operations may require a succession of time slots to be allocated to these transfers within the device and these are controlled within the microprocessor by sets of the operations called microinstructions, forming the microroutines or microprograms. Generally, although these are transparent (or invisible) to the user, they do become apparent when dealing with timing and speed considerations, since the processor may take several 'clock' pulses to perform an operation. The term 'clock' is used for the basic timing pulses which the machine needs to synchronise its operations. For most microprocessors the clock pulse rate is one or two million pulses per second.

Microprocessor Configurations

In this book we are mainly concerned with 8-bit microprocessors of the most conventional kind. These have the entire CPU on one chip with separate chips for memory stores of various kinds and ancillary units for peripheral handling. Commonly the whole assembly is marketed as a single entity, mounted on a printed circuit board. As a very rough guide, the whole device costs ten to twenty times the quoted cost of the microprocessor chip itself. It is normally equipped with a ROM which contains a set of utility programs for operating and monitoring the processor. Microprocessors are designed primarily to be used in dedicated roles: these system assemblies are intended to help the designer to make a start in developing and testing the dedicated systems.

An important and emerging configuration is the single chip microprocessor system. These have the CPU, storage and peripheral controllers all on the one chip and are of particular use for highly specialised roles such as pocket calculators. At present most of these devices have to be programmed during manufacture and require a somewhat complicated design approach. Devices are now appearing, however, for use in much the same kind of role but which are programmable by the user by a fairly simple technique. These single-chip devices will be described later in this book.

A third category of devices will only be mentioned; these are processor chips which either work on 16-bit data or are of what is called 'bit-slice' construction. The former are especially suitable to form the processing units of 16-bit minicomputers and are used as such. The bit-slice devices are an arrangement in which the processor is devided up into its major organs each on a chip and can be put together to suit the designer's requirements for minicomputer or even large array processing computer systems. They are beyond the scope of this book but ample descriptive literature for them is available from the manufacturers.