to another. In fact, the CPU and the system are more complicated than that, but it's not a misleading view. You can think of the CPU as a master controller that sets lesser switches throughout the system to control the flow of electricity, and thus controls the flow of information indirectly, rather than routing all information physically through itself.

The effects of the CPU's switching operations can be classified for our purposes as: arithmetic operations, logical operations, memory operations, and control operations. These operations are all the results of switching information through different paths in the system and in the CPU, and to the CPU they all seem like the same sort of thing.

Arithmetic operations are really the most important feature of the machine. The CPU can add two numbers together, or subtract one from the other. Subtraction is achieved by representing one of the numbers as a negative number and adding that negative number to the other number; 7+5=12 really means:

(plus 7) added to (plus 5) equals (plus 12).

7−5=2 really means:

(plus 7) added to (minus 5) equals (plus 2).

Multiplication and division are regarded as repeated additions or subtractions, so the CPU can be programmed to simulate these processes as well. If the CPU can cope with the four rules of arithmetic, then it can cope with any mathematical process. It is well to remember, however, that all its mathematical potential relies on the ability simply to add two numbers.

Logical operations for our present purposes can be described as the ability to compare two numbers: not merely in terms of relative size, but also in terms of the pattern of their digits. It's easy to see that seven is bigger than five because we can take five away from seven and still have a positive result. The CPU has the ability to do that sort of comparison, and it can also compare 189 with 102 and recognise that both numbers have the same digit in the hundreds column. It may not seem a very useful ability as yet, but its use will become more evident later.

The CPU can perform essentially two memory operations: it can copy information from a memory location into its own internal memory, and it can copy information from its internal memory to another memory location. By doing these two things one after another it can therefore copy information from any part of memory to any other part of memory. For the memory to be any use, the CPU must be able to do both these things, and these two operations are all it needs for complete management of the memory.

Control operations are really decisions about the sequence in which the CPU performs the other operations we have described here. It's not important at the moment to understand them any better than that: if you accept that the CPU can make decisions about its own operation, then that is sufficient at this stage.

So the CPU can do arithmetic, it can compare numbers, it can move information around in memory, and it can decide its own sequence of operations. This is a simple list of procedures, and yet it completely descibes or specifies an ideal computing machine! If the CPU can do those four things, then by doing them in the right sequences it can perform any computable task. The right sequence, of course, is the computer program for the particular task, and that's where we as programmers come in. If the CPU had the ability to generate its own operation sequences, then there would be no need for us.

You may not be convinced that the four types of operation we have described are a sufficient description of a conceptual computer, so let's think about a BASIC program in terms of the general operations performed. What are these fundamental operations? In any program you have variables, which are simply the names of places in memory where information is stored. Most programs perform some sort of arithmetic upon some of these variables. Having done the arithmetic, a program will often compare two pieces of information and as a result will execute one set of instructions or another. Information usually comes into a program from the user at the keyboard, and goes out to the user via the screen.

Except for the sentence about input and output, this description contains no more than the four elemental CPU operations put into different words. And, if you accept for the moment that to the CPU all Input/Output devices are just special areas of memory, then the picture of the ideal computer executing actual programs is complete. Consequently, the execution of a program can be described as a directed flow of information into, around, and out of the computer; you supply some information via the keyboard, that information is manipulated by your program, and some information appears on the screen.

If the idealised computer is just a CPU and some memory, then before going any further we should investigate computer memory: what is it, and how does it work?

Imagine a simple electrical circuit consisting of a battery, a switch, and a light bulb: if the switch is closed the light goes on, and stays on until the battery runs down or until the switch is opened. Then the condition of the light bulb — ON or OFF — is a piece of information, and the whole circuit is a memory device recording that information. Suppose now that the switch is placed at the entrance to a factory, and the light is placed in the Manager's office. When the first employee arrives at the factory, he or she closes the switch at the entrance, and the Manager in the office can see that the light is on and therefore knows that someone has turned up for work. The Manager doesn't have to be in the office when the light goes on; he or she can look at the light bulb at any time to find out whether someone has