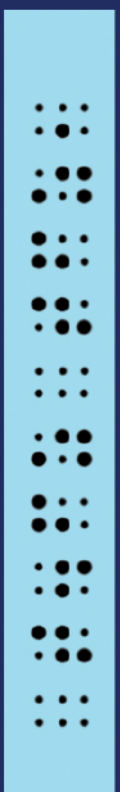
BC

BEFORE COMPUTERS

On Information Technology from Writing to the Age of Digital Data

Stephen Robertson



https://www.openbookpublishers.com

ISBN Paperback: 978-1-80064-029-0
ISBN Hardback: 978-1-80064-030-6
ISBN Digital (PDF): 978-1-80064-031-3
ISBN Digital ebook (epub): 978-1-80064-104-4
ISBN Digital ebook (mobi): 978-1-80064-105-1
ISBN Digital (XML): 978-1-80064-106-8
DOI: 10.11647/OBP.0225

Cover image: Katsushika Hokusai (1760-1849), A merchant making up the account. Wikimedia https://commons.wikimedia.org/wiki/File:A_merchant_making_up_the_account.jpg Public Domain.
Cover design: Anna Gatti.

10. Calculation

Let us now return to numbers.

After the invention of the zero and positional notation by Hindu mathematicians, and the systematisation of the rules of arithmetic by the Arabs, it was only a matter of time before we would turn our attention to calculating by machine.

Machines that calculate?

In one sense, a kind of mechanically-aided calculation had existed for far longer, in the form of the abacus. In fact, the abacus involves a kind of version of the positional notation of the later Arabic system—a column or row of the abacus is still there, even if it has no beads in it, so the zero is represented. And the Arabic rules were embodied in the knowledge of trained abacus-users long before they were codified.

The abacus helps the human to calculate (and also to remember the final result of a calculation)—it does not do the calculation itself. The Hindu-Arabic system (both the representation of numbers and the rules of arithmetic) allowed the use of pen and paper with the same speed and accuracy as the abacus, with the added advantage of retaining a permanent record of the calculation. Machines that calculate? That would be revolutionary, but this system also gives us the foundations of such a revolution. Given that each column of a number is to be treated in exactly the same way as all the others, and the rules have been codified, mechanisation is invited.

One person who first made serious inroads into this idea was Blaise Pascal. In the first half of the seventeenth century, as a precocious teenager, Pascal developed a calculating machine to help in his father's financial calculations. Numbers were dialled on a series of wheels. The machine could be used to add, and, by a process of making subtraction look like addition,

 https://doi.org/10.11647/OBP.0225.10

also to subtract. It could not yet multiply or divide. A similar machine, which has not survived, was invented by Wilhelm Schickard about the time of Pascal's birth, and another famous mathematician, Leibniz, devised another such machine shortly after Pascal.

Logarithms and the slide rule

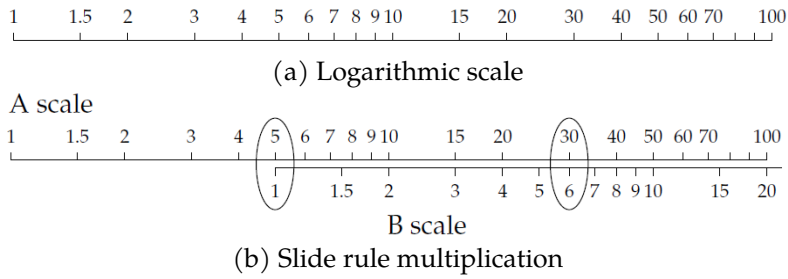
Pascal's and Leibniz's machines represent, in direct mechanical form, the Arabic rules of arithmetic, whereby numbers are made up of discrete digits arranged in columns. This makes them the forerunners of the digital electronic calculators of the late 20th century.

But this is not the only approach to mechanical aids to calculation. A very different method takes what might be described as an analogue approach to calculation (as before, we can regard analogue and digital as opposing principles). This was happening in parallel with the work of Shickard, Pascal and Leibniz. It is clear that the time of mechanical calculation had arrived, even if the methods were still in dispute.

The principle of the logarithm was developed by Napier, around the beginning of the seventeenth century. The characteristic of logarithms is that they convert multiplication into addition. That is, in order to multiply two numbers, you take their *logs* (that is, you look up in a table the logarithm of each number). Then you add them together, and then you take the *antilog* of the result (that is, do the reverse lookup in the table). Great efforts were made to compile accurate log tables over the next two or three centuries.

An alternative is to have the numbers represented on a scale (like a ruler or measuring tape—see Figure 16), but spaced according to their logarithms rather than in the usual equal spacing. Then two numbers can be multiplied by adding their lengths on this scale. Initially this was done by having a single scale and using dividers, but subsequently the idea of two scales that could be slid against each other replaced the single scale. The slide rule was born—see Figure 16.

In setting up a multiplication on a slide rule, you move B scale so that the 1 (on B) lines up with one of the numbers you want to multiply (on A). Then you look up the second number on B and read off the result from the corresponding position on A. So a number is a position; you can set it more or less accurately (just as you can measure with a ruler more or less



(a) Logarithmic scale

(b) Slide rule multiplication

Figure 16: Slide rule. Diagram: the author.

accurately). The digits that you might use to write down the number are not involved. This is why we might describe it as analogue; the position is an analogue representation of the number, not a digital one.

Provided that some degree of approximation was acceptable, the slide rule principle was an effective method for multiplication and division for three centuries or so before digit-based calculators could compete in this domain, and remained in use for most of another century. But in the late-twentieth-century IT revolution, when digital principles ran riot over vast regions of human endeavour, the slide rule lost its status as the pre-eminent method of calculation favoured by scientists and others.

However, I jump ahead. First, we have to see how the digital triumph began.

The comptometer

Many inventors (or would-be inventors) of mechanical calculators, from the mid-nineteenth century on, had in mind the idea of a key-driven device. That is, one would key in the number on some form of keyboard. This looks like a different principle from the Pascal system of setting dials. However, it shares with Shickard, Pascal and Leibniz a digital view of calculation.

This idea took some time to become a reality. Essentially, the difficulty lay in precisely those rules of arithmetic that the Arabic mathematicians had given us, and which had inspired the quest for mechanical calculators in the first place. The particular rule that causes difficulty in the design of key-driven calculators is the carry rule: carrying from one column to the next. Part of the problem is the recursive nature of carry: a carry to a second

column might trigger a carry to a third, and so on. (Pascal had, and resolved, the same problem with his wheel-based calculator.)

It is interesting to speculate here, as in the discussion of the typewriter, why keys were regarded as so important. We have of course the very considerable history of keyboard musical instruments, and later, while the key-driven calculator was still struggling, we have the successful development of the Hughes telex-like machine and later the typewriter. It seems that something about the keyboard as a transparent method of controlling a mechanical device really appealed to Victorian inventors—a sort of Platonic ideal of fingertip control which one can see reflected in many 20th and 21st century approaches to design.

At any rate, the mechanical problem was eventually solved, and by about 1890, there were full-function key-driven *comptometers* for standard arithmetic operations—'full-function' implies that they could be used to multiply and divide as well as to add or subtract. In fairly short order, there were also printing calculators, and then, in the twentieth century, electrically powered and eventually electronic devices.

Babbage

A different view of mechanical calculation is due to Charles Babbage, the nineteenth-century mathematician and inventor (active from the 1820s until his death in 1871). His name is associated with two machines that are seen as the ancestors of modern computers, the Difference Engine and the Analytical Engine. Actually, he failed to produce a working version of either machine, although forms of the Difference Engine were made subsequently. The design of the much more ambitious Analytical Engine anticipated modern computers in many interesting ways.

These machines, like comptometers, can be described as digital (the representation of numbers used decimal digits rather than the binary form common today). The Difference Engine was a calculator on a grand scale, but aiming at automating the repetition of many similar calculations, rather than just a single calculation at a time. Thus it can be seen as anticipating the idea of *programming* a machine to do many calculations. The Analytical Engine would have been programmable to undertake many different kinds of tasks, by implication going well beyond what was seen as arithmetic calculation.

The ostensible purpose for developing these engines was an odd throwback. Babbage did not imagine them, or at least did not sell them to his backers, as serving general calculation purposes. Rather they would be used specifically to calculate the tables—tables of logarithms and the like—that would be published and distributed, in the already established way, to allow people to do their own, unmechanised, calculations and computations. A major reason for the work was the known fact that existing tables contained many errors, primarily because of the involvement of people doing tedious repetitive tasks at all stages of their construction.

Input and output

This aim, generating printed tables, led Babbage to think about other issues than calculation, in particular the input and output stages. For output, he sought to automate part of the printing process: in particular, to have the machine construct the plates for printing, thus avoiding typesetting errors. (Plates for printing, as opposed to movable type, have a long history, briefly discussed in Chapter 4.) This anticipated by more than a century the revolutionary effect of the computer on the printing industry.

He also addressed the question of input—both of numbers and (in the case of the Analytical Engine) instructions to the machine as to what to do. The notion of keyboard input, so important to the comptometer inventors, was of no interest to him. Instead, he proposed making use of Jacquard's invention of punched cards, which we encountered in an earlier chapter. Jacquard successfully used punched cards to control looms; Babbage would set them the task of controlling his Analytical Engine. In the event, the punched card idea would be taken up at the end of the nineteenth century for another purpose altogether, as we shall see in the next chapter. But eventually, in the 1950s, they would be used in much the way envisaged by Babbage.

Computability

Babbage and his collaborator Ada Lovelace began to develop some general notions of what might be 'computable', in other words, what kinds of task might be susceptible to being delegated to a machine. Arithmetic calcula-

tion was clearly in this category, having been reduced to sets of rule-driven steps ("algorithms") by the tenth-century mathematicians of the House of Wisdom in Baghdad. But Babbage and Lovelace believed that the possibilities went far beyond calculation.

Although a working Difference Engine was eventually constructed, Babbage's ideas mostly died with him. The twentieth-century inventors who brought the modern computer into being were largely unaware of Babbage's work.

Nevertheless, the notion of *computability*, what might in principle be computable, would be taken up by twentieth-century mathematicians such as Kurt Gödel, Alonzo Church and Alan Turing. In the 1930s, Alan Turing described (as a mathematical abstraction) a general-purpose computing machine—and subsequently made major contributions to the code-breaking effort in the Second World War which, as we shall see in Chapter 12, led on to the development of actual computers.

But something else happened in the 1890s, which brought machines into a rather different form of work in a very practical way. This is the subject of the next chapter.

On numbers and machines

Depending on your age, you (or perhaps your children) may well have learnt at school about binary numbers and binary arithmetic. Using basically the same positional notation system as bequeathed to us by the Arabs, but only two digits, 0 and 1, we can represent any numbers and perform any arithmetical operations. In this scheme, the rightmost digit represents the units, but the next to the left represents the twos rather than the tens, and the next after that the fours $(2^2 = 4)$ rather than the hundreds.

You will probably also know that computers operate with bits (bit is an abreviation of *bi*nary digi*t*), and you may well associate the digits of binary arithmetic with the bits in the computer. Thus inside computers, numbers are in binary, because that's what computers know about and work with, right?

Well, no, not quite. Actually, it is entirely possible to hold all numbers and do all arithmetic inside a machine in the traditional decimal Arabic system. The system is called BCD (binary coded decimal); it involves a direct

representation of each of the decimal digits, and rules for arithmetic operations on them that would be familiar to a primary-school child. Furthermore, retaining the conventional decimal structure has some advantages over converting to pure binary—for one thing, it is difficult to approximate a number according to the rules usually applied to decimal numbers, if you are operating in the pure binary representation. Babbage's machines were to have worked in something akin to BCD; modern electronic calculators usually work in BCD.

In fact, there is more than one pure binary version. There is a form of representation, slightly different from the one described above though using essentially the same arithmetic, called *two's complement*. This has some advantages, including simplifying dealing with negative numbers, and is often used in computers. Thus we have at least three different forms of representation and two different forms of arithmetic. There are also different ways to represent fractional numbers, very large or very small numbers, and numbers that require great accuracy.

As a general rule, machines can convert numbers from one representation to another, internally, and back again. Thus we do not see all these various representations or methods—we just see the results in the usual decimal system.