MEMORY MONITOR

The hexadecimal number system appears to be a complicated and cumbersome alternative to our everyday decimal, but it is in fact an extremely useful and easily understood way of dealing with memory addresses and their contents when faced with the limitations of an eight-bit byte system of memory.

At this point in the Machine Code course, it's worth returning to the question of number representation. We are already familiar with the decimal (or denary) number system which we use most of the time, and we've investigated the binary system (see page 18). It is well to remember that both the decimal and binary systems are simply alternative expressions of the same concept — number. Most human beings, for example, have the same number of fingers per hand. You may say that the number is five, and someone else may call it *fünf*, or *cinq*, or *pente*; but a moment's empirical investigation will show that you're all talking about the same quantity or number — it is only your representation systems that are different. Different but equivalent. There is a one-to-one correspondence between all numbers expressed in English and all numbers expressed in any other language, and there is internal consistency in all these systems. Arithmetic yields the same results irrespective of the language used to describe the individual components of an arithmetic expression.

Different number systems are exactly similar to different languages. The number of fingers on a normal human hand is not changed by being called *fünf* or five, neither is it altered by being written 5 or 101b (the b here showing that 101 is to be interpreted as a binary number). The only reasons for choosing one system or the other are either custom or convenience.

We find it convenient to use decimal representation at first because it is the number system most commonly used around us. But it is not the only system. Digital clocks, for example, use a bizarre system of arithmetic: part decimal, part modulo 60 (there are 60 minutes in an hour and 60 seconds in a minute), and part modulo 24 (24 hours in a day). Before 1971 British money was reckoned in units of 12 (pence in a shilling) and 20 (shillings in a pound). Learning to use this system took years of agonised schooling — how many people ever really learned how to express shillings and pence as decimal fractions of a pound?

When talking about computers, we find it instructive to begin by talking about binary numbers because they so closely model the computer's electrical operations, being simply sequences of on-off switch states. If we only ever wanted to talk about single-byte numbers, then binary might serve as a complete alternative to decimal — translating eight-bit binary into decimal becomes surprisingly easy with a little practice. Unfortunately, memory addresses in particular and useful numbers in general are usually too large to be fitted into one byte, so computer programmers and engineers over the years have felt the need for a number system with the logical convenience of binary, as well as the range of decimal. Two systems have been used for these reasons: hexadecimal and octal. The first, now standard in microcomputing, is usually called *hex*, and is based on the number 16. Octal, based on 8, has been widely used in mainframe computing, but is increasingly being replaced by the hex system.

RUN TIME

One Edinburgh branch of a national chainstore sold 75 48 K Spectrums in less than 15 minutes just prior to Christmas 1983 — only a few days after Sinclair Research had celebrated the production of the millionth unit of that model

USING HEX NUMBERS

In looking at decimal and binary representation, we have seen two consequences follow from the choice of number base: the base is the number of unique digits needed in the system, and it is the multiplicative factor in the positional notation. For instance, there are ten unique digits in decimal (0–9), and the value of a decimal digit is multiplied by ten each time it shifts leftwards in a decimal number.

Hexadecimal, therefore, requires 16 unique digits, and they are the digits 0–9 and the letters A–F. Counting in hex is simply a matter of working through the single digits and then re-using them in positional notation. The hex number after 9, therefore, is A (decimal 10); next is B; next C; and so on until F (decimal 15). That exhausts the single digits, so the hex number after F is 10 (say this as: 'one-zero hex'), which corresponds to 16 decimal. From this we can see how the single digits are used, and that the value of the columns in a multi-digit hex number increases by a factor of 16 with leftward movement. In a decimal number we call the columns: Units, Tens, Hundreds and Thousands. By comparison, in a hex number the columns are: Units, Sixteens, Two-Hundred-and-Fifty-Sixes and Four-Thousand-and-Ninety-Sixes. By comparing the changes in the binary column with the changes in the hex column, you should be able to see the major advantage of hex numbers: the range of a four-bit binary number is exactly that of a single-digit hex number (i.e. 0 to 15 decimal). Some examples should make this clear: