Properly Addressed

The CPU has to locate instructions and data stored in thousands of bytes of computer memory. We reveal what goes on inside the CPU when program instructions are executed

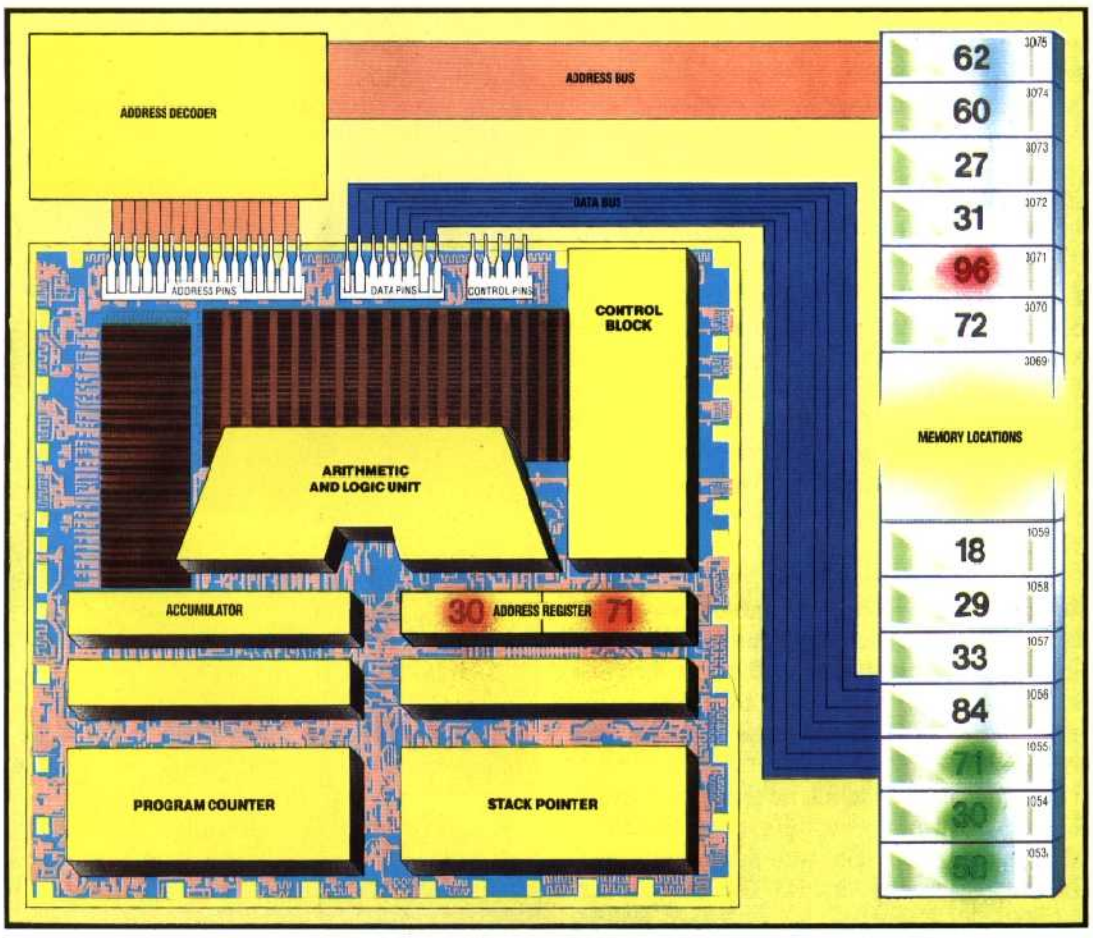
The CPU receives its instructions and data from locations in the computer's memory by setting its address pins to the required binary code for the memory location and then reading the contents of the location into the CPU via the data bus. In actual operation, however, the operation is rather more complicated.

The problem is that the bytes (eight-bit binary codes) in any of the thousands of memory cells in the computer's memory might be instructions, telling the CPU to do something, or data, which the CPU must manipulate in some way. How does the CPU know which bytes are instructions and which are data?

Recognising Codes

First, let's consider what an 'instruction' is. It is a code, in binary, which causes a specific sequence of operations to be performed within the CPU. Thus the code 00111010, if recognised by the CPU as an instruction rather than as just a piece of data, might make the CPU address the next two bytes in memory, read in the data from them, put that data in a special 'address register', set the address pins to the same number, go to the newly addressed memory location, get the contents of that location on the data bus and load those contents into the CPU's accumulator.

This can sound confusing when expressed in words, but what we have just described is one of the methods of memory addressing used in the popular Z80 CPU. The entire process of getting a byte of data from memory into the CPU is shown in the illustration. Suppose the CPU already knows that the next byte accessed from memory will be an instruction (not data) and that this byte resides in memory location 1053. (All the numbers used in this illustration are in decimal notation.) This address, 1053, will be put on the address bus. In binary, this is 0000010000011101. The 16 address pins are switched 'on' or 'off' to correspond to this number. When the 'address decoder' receives this address over the address bus, it 'decodes' it and switches on one, and only one, of its output lines. This is the line that selects



Chains Of Events

Many stages are involved in ever the simplest CPU operation. Instructions, also called 'op-codes', are read into the CPU from memory. These instructions are decoded by the control block and cause specific operations to occur. In this example, instruction 58 is read in from memory location 1053. This particular instruction causes the following chain of events to occur: the byte in the next memory location (1054) will be read in and stored in one half of the CPU's 16-bit address register. The byte in the next location (1055) will be read in and stored in the other half. These two bytes now represent the address (elsewhere in memory) where some data is stored. The contents of the address register are now put on the address bus so that the next memory location accessed will be address 3071. The contents of this address are put onto the data bus and read into the CPU. This byte (96 in our example) is then placed in the CPU's accumulator, where it will stay until operated on by a further instruction. The address bus will then revert back to its previous address + 1, so that it will now be addressing location 1056. The CPU knows that whatever is contained in that location must be an instruction and a similar sequence of operations will be repeated. In this example, the next instruction is 84, which is interpreted by the control block to 'complement' or invert the bits in the accumulator. Since 84 is a 'one byte' instruction, the CPU knows that the byte in the next memory location, 1057, will also be an instruction