CPUs in 8-bit microcomputers (this includes almost all small home computers) are usually packaged in a single chip with 40 pins, 20 on each of the long sides. Every one of these pins (apart from the 0 and the +5v power supply pins) carries signals into and out of the CPU from other devices such as memory or I/O circuits.

Typical 8-bit CPUs have 16 'address' pins. These pins are connected to the 'address bus'. Each of these pins carries an output signal representing either a one or a zero. Between them, they can have 65,536 different combinations of ones and zeros. They are used to select specific memory locations.

There are also eight 'data' pins, which are connected to the 'data bus'. The data pins carry data into the CPU from memory or I/O, or data from the CPU to memory or I/O.

A number of other pins carry 'control' signals. Some of these signals are outputs from the CPU and others are inputs. We will see how the control signals are used shortly.

Inside the CPU there are a few small one-byte or two-byte memory cells called registers, some of which are reserved for special purposes. The others are used for the temporary storage of information and are called general purpose registers. There are two other important functional 'blocks' in the CPU, the ALU and the 'control block'.

The abbreviation ALU stands for Arithmetic and Logic Unit. This part of the CPU performs arithmetical and logical operations, including (but not limited to) adding, ANDing, ORing and moving bits to the left or right within a byte.

The control block is a special circuit designed to make the CPU behave in accordance with the instruction received from memory. We can take a specific example, using the instruction codes for the popular Z80 CPU. If the coded instruction 11000110 is received from memory, the CPU will add the contents of the next byte in memory to the contents of one of the registers inside the CPU. If we then want to store the result of that addition in a specified memory location, the next instruction received by the CPU will have to be 00110010, followed by two bytes specifying the actual location in memory where we want the result to be stored.

Suppose that the result of the addition was 37 (using decimal notation), and that the two bytes following the instruction specified address location 33126 (again using decimal notation). The instruction code would cause the control block to set the address pins to the binary equivalent of 33126 (in binary this is 1000000101100110). It would then cause the control pins to send out signals telling the memory to expect some data and that the data must be stored (memorised). It would then cause the data pins to be set to the binary equivalent of 37 (in binary this is 00100101). This data would then pass down the data bus to the memory and would be stored in the memory location specified by the address bus. If, at some later stage, the CPU needed this data to process in some other way (to print on the screen, for example), a different instruction could be sent to the CPU. The control block would interpret this as meaning 'address memory location 33126, get the byte from it and store it temporarily in one of the internal registers'.

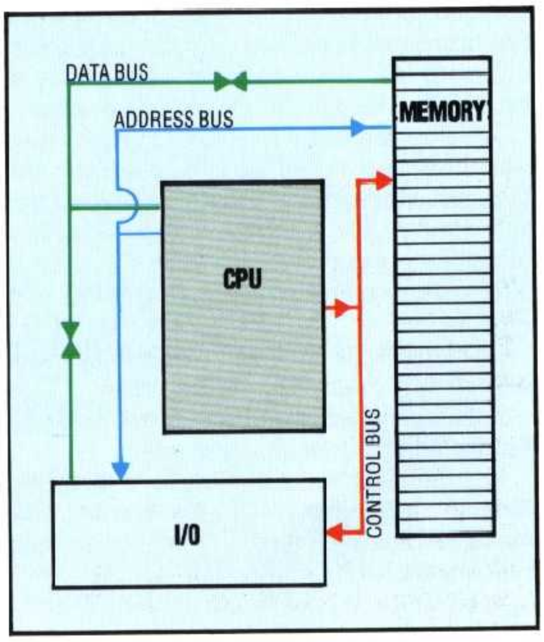
The number of registers or temporary memory cells inside the CPU depends on the CPU. They will either be eight-bit (one byte) registers or 16-bit (two byte) registers. The specialised registers are usually given special names such as the 'stack pointer', 'program counter' or 'accumulator'. The general registers are usually referred to as 'the X register', 'the Y register', 'the C register' and so on.

One of the 16-bit registers, and one of the most important, will be the 'program counter' register. This internal memory cell always contains the address (in binary) of the next instruction in memory due to be executed. When the time comes to get the next instruction for the CPU, the contents of the program counter will be put on the address bus, and the byte at the location will be transmitted (via the data bus) to the CPU.

The most important of the eight-bit registers is the 'accumulator'. This is the register that usually stores (temporarily) the result of operations performed by the ALU, bytes brought in from memory or I/O, or the place where bytes are temporarily stored immediately prior to being sent out to memory or I/O.

This introduction to CPUs has been very general; specific points will be covered in detail later. The aim has been to show that special instructions read in from memory cause the CPU to perform specified operations and to set the address pins to access particular memory locations. Data is fetched from these locations, or sent to them, over the data bus. The instructions also cause the control bus signals to indicate to memory or I/O whether data is to be 'read' or 'written'.

Under Control
The illustration shows a CPU with memory 'registers', an Arithmetic and Logic Unit comprising hundreds of logic gates (that perform such operations as adding, ANDing and complementing binary numbers), and a control block. The control block accepts the coded instruction (in binary), interprets it, and causes all the other parts of the CPU to behave appropriately. For example, if an instruction means that the contents of the accumulator should be stored in a particular memory location, the control block will put the address on the address pins, send control signals telling memory to store data, and put the contents of the accumulator on the data bus for transmission to the memory



Following Orders
A very simple computer may consist of nothing more than a CPU, some memory and an I/O circuit. The memory will store special instructions that make the CPU perform specified actions. It will also store data for the CPU to process in accordance with the instructions. The I/O circuit will be needed for the CPU to communicate with the outside world. If the computer is controlling a washing machine, the I/O circuit will input signals from the buttons on the front panel and output signals to switch motors and heaters on and off.

The instruction codes for the CPU will be in binary. Each different model of CPU has its own set of instruction codes