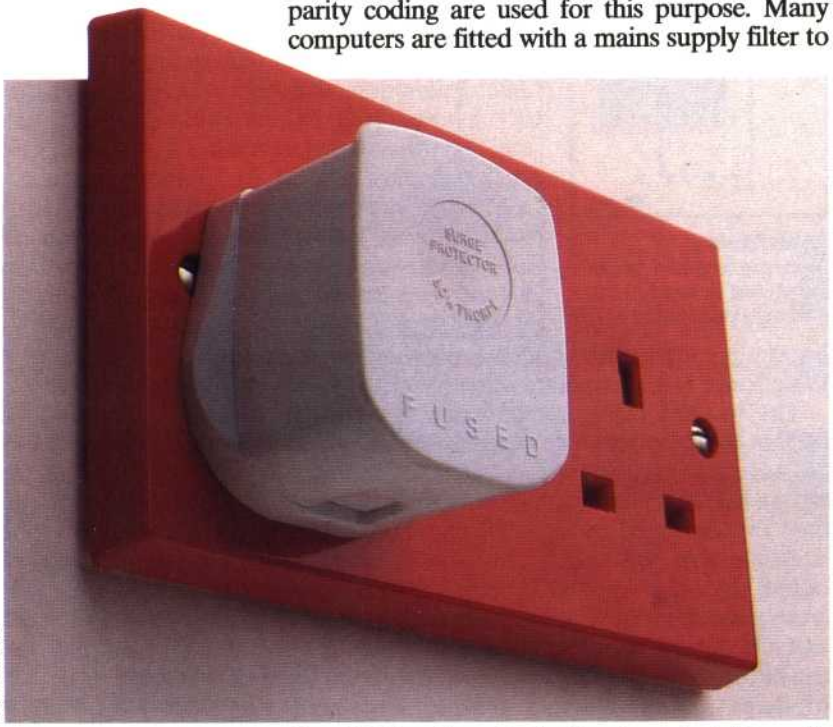
FILTERING

Anyone who has ever used the tone control on an amplifier knows the commonest meaning of *filtering* — altering a signal by blocking the transmission of some of its component waveforms. The simple rumble filter often seen on audio amplifiers, for example, is a variable *high-pass* filter, which passes only those signals whose frequency is higher than some set value: as you turn it up, this lower limit is raised, thus favouring the high-frequency treble sounds.

The filtering of computer data transmissions to remove electrical noise is vital in preventing errors being induced; both frequency filters and error/parity coding are used for this purpose. Many computers are fitted with a mains supply filter to block high-voltage 'spikes' or 'transients' (usually induced by the switching of high-current devices such as lift motors or commercial freezers). These pulses might otherwise get past the computer's voltage regulator, and damage the chips or reset the system.



IAN McKINNELL

Plug-In Protection
Filtering the mains power supply using a plug-in unit like this to suppress high voltage 'spikes' or current surges can save computer users hours of wasted effort by eliminating the accidental resets that may be caused by such transients

Information can be filtered, or *masked*, by logical operations. For example, ANDing the contents of a byte with 10000000 masks or filters the lower seven bits, passing only the most significant bit. If the byte contains a two's complement number, then this filtered result is the sign of the number, indicating whether it is positive or negative.

FLAG

A program variable whose value indicates the state or outcome of a process is called a *flag* — so named because it is analogous to a real flag, which can signify different meanings depending on whether it is up, half-mast or down. The CPU has a *flag register* (also called the *status* or *condition code register*) whose bits (the flags) are set to show the outcome of the processor operations. If an eight-bit processor added 236 to 101, for example, then the zero flag would be cleared, showing a non-zero result, and the carry flag would be set, showing that the result was greater than 255 — the accumulator's numerical limit. Some of the processor's Assembly language instructions vary their actions according to the state of these flags, thus allowing program decision-making.

FLIP-FLOP

Since computers run on binary logic, an electronic device that exhibits two stable states, and will switch states in a predictable way, is an essential requirement for constructing either memory or logic circuits. This essential component is called a *flip-flop*, or bistable (see page 168), and it is usually made by cross-connecting two NAND gates so that the output of each is an input of the other (see page 228).

FLOATING POINT NOTATION

A computer usually has at least two formats for its internal representation of numbers: *integer* and *floating point* notation. Integer notation is reasonably straightforward — two bytes are allocated to each integer variable (e.g. X%) in signed 16-bit notation; the range of possible integers, therefore, is:

Decimal	−32768			+32767
Binary	10000000 00000000	to		01111111 11111111

In floating point notation, however, numbers are represented thus:

$$+\text{number} = +\text{mantissa} \times \text{base}^{+\text{exponent}}$$

$$\text{e.g. } +317440 = +.60546875 \times 2^{+19}$$

Since, in a binary machine, the internal number base is always two, there is no need to store it, so only the signed mantissa and exponents are saved, thus:

Exponent		Mantissa			
S		S			
0	0010011	0	1001101	10000000	00000000
byte 0		byte 1		byte 2	byte 3

Both the mantissa and exponent are stored in *two's complement* form, so that the most significant bit of each is the sign bit. If the mantissa is adjusted so that it is always in the range ($0.5 \leq \text{mantissa} < 1.0$ — or, in binary fractions, $0.1 \leq \text{mantissa} < 1.0$) then it is said to be *normalised;* this means that the first bit of the mantissa after the sign bit will always be one.

The great advantage of floating point notation is that it allow compact storage of very large and very small numbers. As more bits are allocated to the mantissa, so the precision of this format increases; adding bits to the exponent extends the range of expressible numbers. Furthermore, the exponential format makes writing efficient arithmetic routines — especially for multiplication and division — reasonably easy.