If AND and OR operators are mixed in a condition, AND will be evaluated first.:

```
100 IF A%>5% OR B%>6% AND C5>7% THEN 200
```

The above line is equivalent to

```
100 IF A%>5% OR (B%>6% AND C%>7%) THEN 200
```

AND and OR can be used to perform binary operations on the bits of two integers;

```
10 T%=A% AND 1%
```

AND is performed between the binary bits of A% and those of 1%, and the resulting number placed in T%. For example, bit 0 of 1% is compared with bit 0 of A%; if both are one, a 1 is placed in bit 0 of T%. Otherwise 0 is placed in bit 0 of T%. The other bits of A% and 1% are compared in the same way. The above example has the interesting property that T% will equal 1 if A% is odd and zero if A% is even (a simple way to test whether a number is odd or even). Consider the next line:

```
10 T%=A% OR B%
```

This works like the previous example, except that here an OR is performed between the binary bits of A% and the binary bits of B% and the result placed in T%. For example, bit 5 of A% is compared with bit 5 of B%. If either bit is 1, then bit 5 of T% is set to 1; otherwise it is set to zero.

Special operators: +=, >>, and <<

There are many statements where something is added to a variable. For example:

```
T%=T%+3%        A%(5%)=A%(5%)+C%+2%
B$=B$+","       B(2%)=B(2%)+EXP(2)
C%=C%-1%
```

Using the += command, these statements can be rewritten in the following way, in many cases resulting in faster program execution:

```
T%+=3%          A%(5%)+=C%+2%
B$+=","         B(2%)+=EXP(2)
C%+=-1%
```

The >> and << commands are used only with integers. In the following line, the bits of A% are shifted three places to the right. For a positive number this is the same as dividing by 8 (that is, by 2^3).

```
10 T%=A%>>3%
```

In the next example, the bits of A% are shifted three places to the left. This is the same as multiplying by 8 (by 2^3).

```
10 T%=A%<<3%
```