```
N1 + 2 * Y / Z                N1 + ((2 * Y) / Z)
(N1 + 2) / Y JOIN Z           (N1 + 2) / (Y JOIN Z)
```

Because integer arithmetic is faster than floating point arithmetic, Extended XYBASIC uses integer arithmetic whenever possible. You can often both conserve memory space and increase the execution speed of your program by using integer variables to store values which you know will always be integers in the range -32767 to 32767.

Conversions

As noted in Section 2, you can use both integer and floating point numeric variables in Extended XYBASIC. Because XYBASIC converts between these two variable types automatically, you can write programs in the most natural way.

Of course any integer value can be converted to an equivalent floating point value. Whenever XYBASIC needs to convert a floating point value to an integer, it truncates it to the largest integer less than or equal to the floating point value. For example:

```
NEW
OK
10 I% = 2.5
20 J% = -3.1
30 PRINT I%, J%,
RUN
 2            -4
OK
```

Here XYBASIC converted the value 2.5 to the integer 2 and the value -3.1 to the integer -4. An OV (OVerflow) error occurs if the floating point value is less than -32767 or greater than 32767.

Conversions are also performed automatically whenever a command or function requires an integer argument. For example, the value in an ON command, the dimensions in a DIM command and the subscripts of an array variable reference must be integers, and are converted if they are floating point values.

Since string values cannot be converted to numeric values, a TM (Type Mismatch) error occurs whenever XYBASIC finds a string value where it expects a numeric value, or vice versa. For example:

```
I = "DOG"
TM ERROR: I = "DOG"

OK
S$ = 1.5
TM ERROR: S$ = 1.5

OK
```

In the first example the string "DOG" could not be assigned to the floating point variable I; in the second example the numeric value 1.5 could not be assigned to the string variable S$. The functions CHR$, ASC, STR$ and VAL