```
NEW
OK
10 INPUT "A, B =" A, B
20 PRINT A; "\"; B; "="; A \ B
30 PRINT A; "MOD"; B; "="; A MOD B
40 PRINT "ANOTHER (Y OR N)?";
50 X = GET
60 IF X = 0 THEN 50
70 PRINT CHR$(X)
80 IF X = 89 THEN 10
RUN
A, B =? 10,3
 10 \ 3 = 3
 10 MOD 3 = 1
ANOTHER (Y OR N)?Y
A, B =? 15,4
 15 \ 4 = 3
 15 MOD 4 = 3
ANOTHER (Y OR N)?N

OK
```

Here XYBASIC executes lines 50 and 60 until a character is typed, making X = 0 false. Then the typed character is echoed with CHR$, and if the character is Y (ASCII 89) the program returns to line 10. Notice that the program would NOT work correctly if lines 50 through 80 were replaced by

```
50
60 IF GET = 0 THEN 60
70 PRINT CHR$(GET)
80 IF GET = 89 THEN 10
```

In this case line 60 waits for a character, but its value is not saved and line 70 tries to GET the next character (and probably PRINTs ASCII 0, a null character). Then line 80 tries to GET another character.

In Extended XYBASIC you can use the string function GET$ as well as the numeric function GET. GET$ is described in Section 4.

DELAY

For many applications you may want your computer to act like a stop watch, measuring a certain amount of time. XYBASIC allows you to do this with the DELAY command. Its first argument specifies the number of minutes you wish to delay, its second the number of seconds, and its third the number of hundredths of seconds. For instance,

```
DELAY 0, 4, 50
```

will delay for four and one-half seconds. The second and third arguments are optional.

You can interrupt a DELAY with <control-C>, suspend it with <control-S>, or abort it and proceed with the next instruction by typing any other character. ENABLEd interrupts (described in Section 10) are not active during a DELAY.