If your computer does not have enough memory space to run a large XYBASIC program, you may still find it possible to run it after you rewrite it to decrease its memory usage. Just thinking carefully about a program often lets you save space by eliminating extra steps, but sometimes you will need to know how XYBASIC uses space. Some of the byte-saving techniques suggested below are good programming practice, while others are recommended only if you must conserve space at all costs.

Each line in a program has a fixed overhead, regardless of the number of decimal digits in the line number. You cannot save space by using lower line numbers, but you can conserve by using : and ' to decrease the total number of lines. Grouping related commands and comments on a single line also makes programs more comprehensible.

XYBASIC stores each reserved word (command or function name) of a program in a single byte, and stores every other character (including spaces) in a single byte. You can therefore conserve space by using fewer characters, for example by eliminating spaces, eliminating REMarks, eliminating LETs, and shortening variable names. Each of these techniques usually makes your program harder to read and debug, though.

Using XYBASIC's full variety of commands and functions also lets you write more compact programs. A repeated section of code can be made into a subroutine and called with GOSUB. FOR-loops can replace some loops defined with IF / THEN and GOTO. ON commands can replace sequences of IF / THEN commands. Using the right function, for example TEST (I,2) instead of RSHIFT(I,2) AND 1, also produces more efficient programs.

Besides the space used to store your program, XYBASIC uses free memory to hold information about variables and control information about GOSUBs and FORs. Each variable uses space, so you can save bytes by using the same variable at different places in a program. Each GOSUB uses space until you RETURN, so you must be careful to always RETURN from subroutines. Similarly, each FOR uses space until NEXT exits from the loop, so you should avoid GOTOing out of FOR loops. However, the space used is recovered if you reenter the loop.