FINE TUNING

Our course on program design has so far concentrated on 'structured' programming methods. Using the techniques we have suggested will make your programs easier to design and debug, but will do nothing to make them run any faster. Here we consider ways to increase program execution speed.

Structured programming and good program layout are techniques that make programs easier to use, but do not improve program efficiency. To make programs run faster and use less memory space, it is often necessary to sacrifice clarity in a program's design. So we should bear in mind, when 'tuning up' a piece of code, that almost anything that is done to make it faster will invariably make it more difficult to read, understand and debug.

The inherent slowness of interpreted languages like BASIC means that there will be times when programs will run at an unacceptable pace and must be speeded up. The most efficient way of speeding up a BASIC program is to compile it. However, very few micros support a true BASIC compiler — there are disk- and cassette-based compilers on the market, but most of them support only integer BASIC, and may require special formatting of your program before compilation. Compiling is a slow process, especially during program development, and especially when the system is cassette-based. The compiler will occupy user memory, and the more comprehensive its facilities, the more RAM it will take from the user program area. In general on home micros, compiling is recommended only for fully tested and debugged programs.

File accesses slow programs down more than any other single cause. In a program that frequently reads from and writes to disk or tape (a database program, say) delays are inevitable. Access to a record in a random access file on a floppy disk takes an average of about a quarter of a second. Access to data in serial files takes longer (and varies with the length of the file) and tape accesses are considerably longer. If these delays are causing problems, it may be possible to reduce the number of accesses by reading in more data at once and storing it in RAM, and by 'saving up' updates to files until the end of the session. Interactive programs often cause problems because the user is left staring at a screen for several seconds. A partial solution here is to reorganise the program so that files are read and written while the user is busy doing something else (reading a screenful of instructions, for instance).

Another cause of slowness is real arithmetic. Real numbers are ones with decimal places (integers are whole numbers). Because of the decimal part, fetching a real number from memory and performing an arithmetical operation on it requires many more machine cycles than doing the same for an integer. In programs with a lot of arithmetic, it pays to replace all the variables involved with integer variables (e.g. SUM should be replaced by SUM%). Savings of around 20 per cent can be achieved for even moderately numerical programs and 'number-crunching' applications stand to gain by as much as 50 per cent.

Designing a faster algorithm is one of the best ways of speeding a program up. Some sources of algorithms have already been recommended in this course. Try these, and be on the lookout for those published in computer magazines. Otherwise, devising algorithms is a matter of creativity and insight. BASICS usually have a wealth of inbuilt functions (such as INSTR, SGN, LOG, and so on) that are very fast. This speed is a result of their being written in machine code and using the best algorithms available. It is often worth checking the manual again to see what inbuilt functions are offered before coding your own version. User-defined functions, implemented with the command DEF FN, also run quickly. This command is most useful in programs with repeated calculations or a repeated sequence of string manipulations, where it can replace a subroutine call, which is much slower.

Writing routines in machine code generally makes them run faster. This is because interpreted languages translate program lines into machine

Micro Compilers

BBC Micro Turbo Compiler Salamander, 17 Norfolk Road, Brighton BN1 3AA. 0273 771942	Cassette	£9.95
Commodore 64 DTL-BASIC Compiler Dataview Wordcraft Ltd, Radix House, East Street, Colchester CO1 2XB. 0206 86914	Cass/Disk	£14.95
Spectrum Softek FP Softek IS, Combined FP and IS, Softek International, 12/13 Henrietta St, London WC2. 01-240 1422	Cass Cass Cass	£19.95 £9.95 £24.95