• OSBYTE enables us to affect the behaviour of many OS routines, by passing control codes and parameters in the A, X and Y registers of the 6502. In BASIC, we can access the OSBYTE routines via the *FX command. *FX is followed by either two or three numbers: the first number is the control or function code passed to the A register; the second is the number passed to the X register; and the third is the number passed to the Y register. The third parameter is not required by all OSBYTE calls. In machine code, OSBYTE is called at address &FFF4. These two versions are equivalent in function:

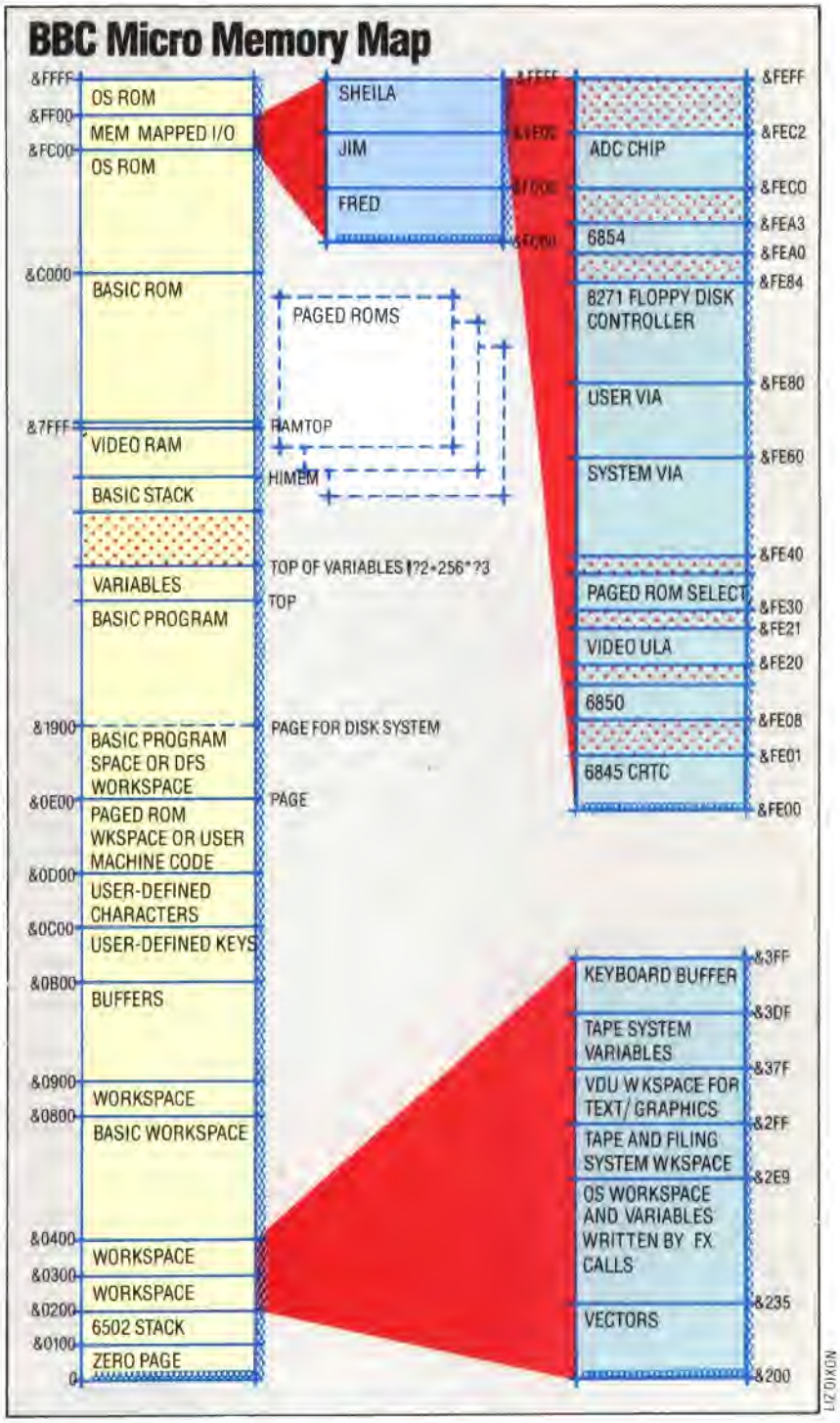
BASIC	Assembly Language
*FX4,1	LDA #4 LDX #1 JSR &FFF4

The value in the A register defines exactly what a particular call to OSBYTE will do. The above call, for example, affects the actions of the cursor keys; the parameter passed over in the X register specifies whether the cursor keys retain their normal editing function or whether they simply return an ASCII code.

It's a sad fact of life that all OS routines cannot be affected by OSBYTE. Its main drawback is the limit on the number of parameters that you can pass over to the routine. If we ignore the contents of the A register, which tells the OS which routine within OSBYTE we wish to use, then we can only pass two parameters in the X and Y registers. If we want to pass any more than this then we use the second routine, OSWORD.

• OSWORD enables us to do such things as sound generation, disk reads and writes, and so on. This is the difference between OSWORD and OSBYTE; OSBYTE affects *how* the OS does certain tasks, and OSWORD enables us to *do* particular tasks. OSWORD obtains its parameters from a *parameter block* that is pointed to on entry to the OSWORD routine by the 6502 X and Y registers. This parameter block is situated in RAM, and its size and arrangement depend upon the OSWORD call being made. The A register contains a function code that determines which of the OSWORD functions are to be executed by the OS. Once the registers and the parameter block have been prepared, OSWORD is called at address &FFF1. OSBYTE and OSWORD are the principal means of entry to the OS; because of their importance, we'll look at them in greater detail in later parts of the series.

Other OS routines are entered, from BASIC, by typing in an asterisk (*) followed by the command. The presence of the * causes the command following it to avoid the BASIC interpreter and be passed to an OS routine that bears the name OSCLI, meaning the 'Operating System Command Line Interpreter'. This interprets the OS commands that are typed in and acts upon them by calling the appropriate routines in the OS. Such commands are often called * or Star Commands. The table shown lists those Star Commands recognised by the BBC; any not recognised, spelt incorrectly or without the correct number of parameters will usually give the Bad Command error message.



We can pass commands to the Command Line Interpreter (CLI) by using the OS routine or by the direct method. OSCLI's uses are twofold: first of all, it enables us to pass the commands shown in the table to the CLI from machine code should we want to; and secondly it enables us to pass BASIC string variables over to the CLI. The programs that follow feature both these uses. Notice that the integer variables, A%, X% and Y%, pass their values directly into the A, X and Y registers. X% (or the X register) and Y% (or the Y register) point to the position in memory of the string of characters, that