This version encloses memory addresses in brackets when it means 'the contents of', which reinforces the vital distinction between the address of a byte and what it contains.

The Assembly language program we have given is logically complete, but we shall execute it as a subroutine, so it needs the equivalent of the RETURN command (to refer the program back from the subroutine) to complete it. The opcodes are RTS in 6502, and RET in Z80.

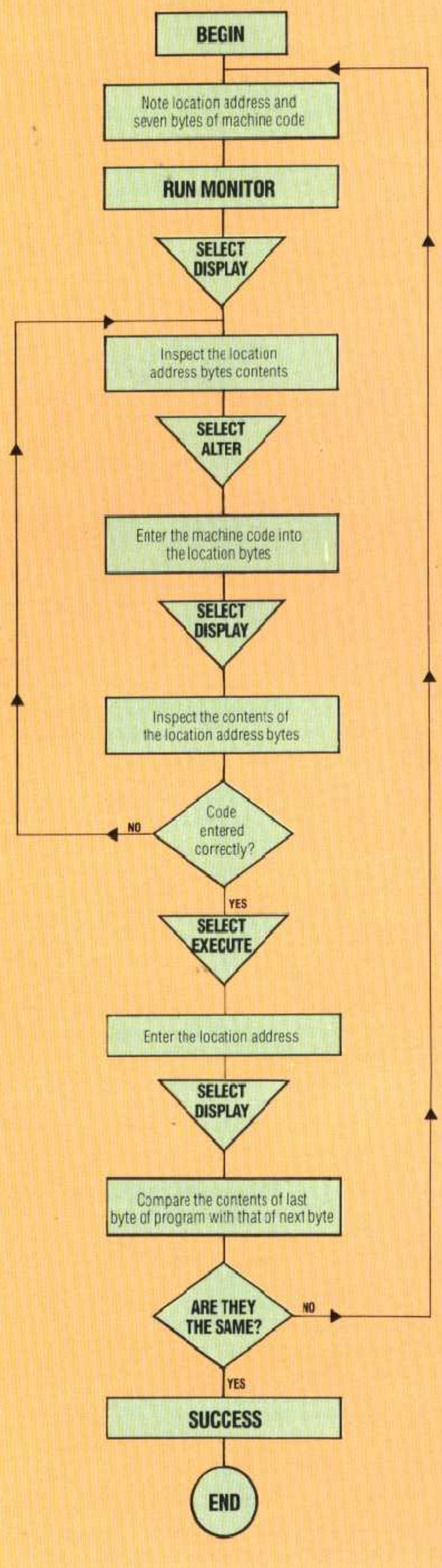
If we are to use this subroutine, we must first translate it into machine code, then store the code somewhere in memory, and cause the microprocessor to execute it. We can use the Monitor program (on page 118) for the latter two tasks; but before that we must first do the translation and decide where the code is to go. This is an unfamiliar decision to BASIC programmers, who never have to think where a BASIC program will be stored — they just type it and RUN it. The storage decisions are made for the programmer by the system designers, and implemented by the operating system.

A machine code program can be stored and executed anywhere in memory, but some places are better than others. The safe places differ from machine to machine, hence the different versions of the program that follow:

6502		
Location Address	**Machine Code**	**Assembly Language**
COMMODORE 64		
$0350	AD 56 03	LDA $0356
$0353	8D 57 03	STA $0357
$0356	60	RTS
BBC MICRO		
$0070	AD 76 00	LDA $0076
$0073	8D 77 00	STA $0077
$0076	60	RTS
Z80		
16K SPECTRUM		
$7FA0	3A A6 7F	LD A,($7FA6)
$7FA3	32 A7 7F	LD ($7FA7),A
$7FA6	C9	RET
48K SPECTRUM		
$FFA0	3A A6 FF	LD A,($FFA6)
$FFA3	32 A7 FF	LD ($FFA7),A
$FFA6	C9	RET

Notice that each version of the program copies its own last byte into the byte that comes after it. The Spectrum 48K program, for example, copies the contents of $FFA6 into $FFA7. Notice also that addresses appear in hi-lo form in Assembly language (for our benefit), but lo-hi form in the machine code translation. A special note should be made that in Z80 the mnemonic is LD in both the first and second instructions, but that the opcodes differ: 3A for the data transfer to the accumulator, and 32 for the transfer from the accumulator.

Using The Monitor Program


BEGIN
Note location address and seven bytes of machine code
RUN MONITOR
SELECT DISPLAY
Inspect the location address bytes contents
SELECT ALTER
Enter the machine code into the location bytes
SELECT DISPLAY
Inspect the contents of the location address bytes
Code entered correctly?
NO
YES
SELECT EXECUTE
Enter the location address
SELECT DISPLAY
Compare the contents of last byte of program with that of next byte
ARE THEY THE SAME?
NO
YES
SUCCESS
END

The program (on page 118) allows you to **ALTER**, **DISPLAY**, and **EXECUTE** memory. Whenever you select one of these options you will be asked for a hex address. This is the address in memory at which to:
1) Start altering memory contents, or
2) Start displaying memory contents, or
3) Start the microprocessor executing a machine code program.

Entering 'X' instead of a number or address will always return you to command level, and at command level 'Q' will terminate program execution.

When you select **ALTER** mode, and have given the address whose contents you wish to alter, that address will be displayed followed by a query; type the new hex contents and hit **RETURN**. The address of the next byte will be similarly displayed for altering. As long as you wish to alter bytes, keep typing the new contents and hitting **RETURN**. If you enter the letter X instead of an address you will be returned to command level.

Loading Into Memory
Look at the program for your machine, note the first location address and the seven bytes of machine code ($0350 and AD,56,03,8D,57,03,60 on the Commodore 64, for example): you are going to use the Monitor program to load these seven hex numbers into the seven bytes of memory from the last location address onwards.
1) RUN the Monitor program, **DISPLAY** the contents of memory from the first location address onwards (from $0350 to $0357 in the Commodore 64, for example).
2) Select **ALTER**, enter the location address, and enter the seven bytes that comprise the machine code program.
3) Select **DISPLAY** again, and make sure that you have correctly entered the machine code bytes into the location address bytes.
4) Select **EXECUTE**, and enter the location address — nothing should appear to happen.
5) Select **DISPLAY**, inspect the location addresses, and you should find that the contents of the last byte of the program have been copied into the next byte