write instructions that contain the address of a pointer, and which operate on the contents of the location that the pointer indicates (not on the contents of the pointer itself). The advantages of this addressing mode are considerable, especially when combined with the indexed mode. Suppose, for example, that you write an Assembly language routine that searches a data table for a given character, and returns with the index position of the character. Suppose, also, that you want to keep several such tables in different places in memory, and that you want to use the same routine for searching any of them. If the routine is written so that it finds the base address of the search table indirectly via a given pointer location, then it can be used on any table provided that the contents of the pointer location are properly adjusted before the routine is called.

In general, programs require mixtures of these modes rather than pure examples of single modes. The 6502 instruction LDA, for example, can be used in the following modes:

Op-code	Operand	Mode
LDA	#$34	Immediate Mode
LDA	$A2	Zero Page Direct
LDA	$967F	Absolute Direct
LDA	$A2,X	Zero Page, Indexed by X
LDA	$967F,X	Absolute, Indexed with X
LDA	$967F,Y	Absolute, Indexed with Y
LDA	($A2,X)	Indirect, pre-indexed with X
LDA	($A2),Y	Indirect, post-indexed with Y

It is convenient to use a 6502 instruction in this example because it shows the combinations of addressing modes so clearly. Notice that the two indirect versions of the instruction are in zero page as well as indirect and indexed modes. A table such as this may seem confusing at first sight, but actually using the various modes soon makes their significance clear, and so far we have used both LDA and ADC in two modes — immediate and absolute — without confusion.

The table does answer the question posed in the last instalment of the course — how to tell the addressing mode of an instruction when the mnemonic is the same in every case? It can be seen that the format of the operand is different for every mode, and that the only ambiguity possible is whether an instruction such as LDA SYMB1 is zero page or absolute mode. An assembler program will resolve this for you, but if you are assembling the program by hand, you will have to determine whether SYMB1 has been defined as a single byte or as a two-byte quantity.

In general, once you start using an assembler, you can forget about things like op-codes and the number of bytes per instruction, and concentrate on learning the programming techniques of Assembly language. It is important to understand the mechanics of machine code, but Assembly language used with a good symbolic assembler is a much better way of programming, combining the power of machine code with many of the facilities of high-level languages.

Answers to Exercise On Page 178

The Monitor program on page 118 was written in modules with expansion in mind, so adding a menu option is reasonably easy:

SPECTRUM VERSION

1) Adjust the initialisation by editing or adding the following lines:

```
1050 LET LT=5:DIM C$(LT):DIM O$(LT,24):DIM
     X$(16)
1150 LET C$="ADGQB":LET C1=-48:LET
     C2=10-CODE(C$(1))
1280 LET O$(5)="    B - BINARY DISPLAY"
```

2) The input routine at line 2000 has already elicited the start address, standardised it as a four-digit hex number in A$, and converted it to a decimal number in DN, so the binary display subroutine reads:

```
7000 REM**HEX&BIN DISP S/R**
7020 FOR P=DN TO (DN+15)
7040 LET NM=P:GOSUB 3100: PRINT H$,
7060 LET N=PEEK(P): LET NM=N
7080 GOSUB 3000: PRINT B$;" ";
7100 GOSUB 7300: PRINT B$
7120 IF P=65535 THEN LET P=DN+15
7140 NEXT P
7200 RETURN
7300 REM**BINARY BYTE S/R**
7310 LET B$=""
7320 FOR D=8 TO 1 STEP-1
7330 LET N1=INT(N/2)
7340 LET R=N-2*N1
7350 LET B$=STR$(R)+B$
7360 LET N=N1
7370 NEXT D
7380 RETURN
```

BBC/COMMODORE VERSION

Copy the Spectrum version, with the following amendments:

1) Change the initialisation of LT and O$() as in the Spectrum version above, and add C$(5)="B" to line 1150.

2) Line 600 transfers control to the command routine, so on the Commodore 64 and BBC Micro change this to:

```
600 ON CM GOSUB 5000,5500,6000,6500,7000
```

3) On the BBC change line 7060 above to

```
7060 N=?(P):NM=N
```

4) On the Commodore 64 change line 7350 above to:

```
7350 B$=MID$(STR$(R),2)+B$
```