byte location, then store the carry flag as the hi-byte of that location. There is no single instruction for storing the carry flag, but the ADC op-code was formulated with precisely this operation in mind: ADC actually means 'add the instruction operand to the current contents of the carry flag, then add that result to the contents of the accumulator'. Addition is thus a two-stage process, in the first of which the current state of the carry flag is used, while in the second stage the state of the carry flag is updated.

This means, then, that before beginning an addition, we must consider the current state of the carry flag, since it will be added into the addition sum proper: hence the two unexplained instructions in previous instalments, CLC and AND A. The former, a 6502 instruction, means 'clear the carry flag', and does exactly that. The Z80 version, AND A, means 'logically AND the accumulator with itself'. While not designed solely to reset the carry flag it does have that effect and doesn't affect anything else, so is often used as a Z80 equivalent of the 6502's CLC.

Having cleared the carry flag before starting an addition, therefore, we must store its contents afterwards. This is achieved by adding the immediate value $00 to the hi-byte of the result. This won't affect the byte if the carry flag is clear, but will add 1 to it if the carry flag is set.

All of what we have said in this instalment leads to the first method for single-byte arithmetic:

1) Clear the carry flag
2) Load the accumulator with one number
3) Add in the second number
4) Store the contents of the accumulator in the lo-byte of a two-byte location
5) Load the accumulator with the contents of the hi-byte
6) Add in the immediate value $00
7) Store the contents of the accumulator in the hi-byte

When this procedure is turned into Assembly language we get:

COMMON TO BOTH PROCESSORS		
Label	**Directive**	**Operand**
BYTE1	EQU	$FF
BYTE2	EQU	$FF
LOBYTE	EQU	$A000
HIBYTE	EQU	$A001
	ORG	$A020

Z80		6502	
Op-code	**Operand**	**Op-code**	**Operand**
LD	A,$00	LDA	#$00
LD	(HIBYTE),A	STA	HIBYTE
AND	A	CLC	
LD	A,BYTE1	LDA	#BYTE1
ADC	A,BYTE2	ADC	#BYTE2
LD	(LOBYTE),A	STA	LOBYTE
LD	A,(HIBYTE)	LDA	HIBYTE
ADC	A,$00	ADC	#$00
LD	(HIBYTE),A	STA	HIBYTE
RET		RTS	

Remember that the values given for LOBYTE, HIBYTE and ORG are for example only — you must choose values appropriate to the machine that you use. Notice that the first two instructions of the program load $00 into HIBYTE, so that it's not corrupted by random data. We don't have to clear LOBYTE in the same way because its starting contents are overwritten with the lo-byte of the result.

It is worth remarking again about the differences of approach between Z80 and 6502 Assembly language as seen in the example. The 6502 code reads quite simply once you're used to it — the mnemonics themselves and the use of '#' to signal immediate data make the meaning of each instruction clear. The Z80 version is less straightforward because the LD mnemonic is used for all data transfers whether into or out of the accumulator. Also, there is no '#' symbol to signal immediate data, only the absence of brackets around the operand indicate this. Thus LD A,BYTE1 means 'load the accumulator with the immediate data BYTE1'; whereas LD A,(HIBYTE) means 'load the accumulator from the address HIBYTE'. In the full Assembly language listing there is no ambiguity in the meaning of such instructions, since the hex value of the op-code uniquely identifies the instruction. This may seem to beg the question, however — the op-code may be unique, but if there is a choice of unique op-codes, how does the assembler (or the person doing the assembly) choose between them? The answer lies in the Addressing Mode, which will be the topic of the next instalment.

Finally, we should take note that the processor status register contains other flags as well as the carry flag, which we'll examine briefly now, and return to in detail later in the course:

Z80 PSR:	**S**	**Z**		**H**		**P/V**	**N**	**C**
Bit Number	7	6	5	4	3	2	1	0
	MSB							LSB
6502 PSR:	**S**	**V**		**B**	**D**	**I**	**Z**	**C**

PSR BIT	Z80
7	(S)=SIGN
6	(Z)=ZERO
5	unused
4	(H)=HALF-CARRY
3	unused
2	(P/V)=PARITY/OVERFLOW
1	(N)=SUBTRACT
0	(C)=CARRY

6502	PSR BIT
(S)=SIGN	7
(V)=OVERFLOW	6
unused	5
(B)=BREAK	4
(D)=BCD MODE	3
(I)=INTERRUPT	2
(Z)=ZERO	1
(C)=CARRY	0

For our present purposes the important flags are the carry, sign and zero flags. We have seen that after an addition the carry flag holds the value of the carry out of the eighth bit of the accumulator. The sign flag is always a copy of the eighth bit (bit 7) of the accumulator, and the zero flag is set to 1 if the accumulator contents are zero, and reset to 0 if the contents are non-zero.