STARTING FLAG

Having already used the add instruction in previous instalments of the course, we now begin to examine its implications in terms of methods of arithmetic, and the system architecture needed to support them. Here, we look more closely at the processor status register and its part in addition — in particular the role of the carry flag.

The add instruction in both Z80 and 6502 Assembly language is ADC — meaning 'Add With Carry' — a mnemonic of great importance for Assembly language programming. The concept of a 'carry' bit is of particular significance. Let's consider the addition of two hex numbers in the accumulator:

```
   A7      =     10100111
+  3E      =  +  00111110
-----------------------------
   E5      =     11100101
```

Since the accumulator is an eight-bit register, both the numbers to be added and the sum itself must be in the range $00 to $FF (as they are here) or else they will not fit into the accumulator. Does this mean, therefore, that we are restricted to additions in which the sum is less than $100? Consider another addition in the accumulator, one which violates this restriction:

```
   FF      =     11111111
+  FF      =  +  11111111
-----------------------------
   1FE     =     111111110
```

This shows the addition of the largest possible single-byte numbers, and seems to be an illegal addition. It requires a nine-bit accumulator. The solution to this dilemma is suggested in the statement of the problem — we need only an extra bit on the accumulator to contain the largest number that can be generated by the addition of single-byte numbers. That extra bit is required only in the sum, not in the addition operands, and it is required only when there is a 'carry' from the most significant bit of the accumulator.

PROCESSOR STATUS REGISTER

The extra bit is therefore known as the *carry bit*, and it is located in the eight-bit register associated with the accumulator known as the *processor status register* (PSR). This important register is connected to the accumulator and the ALU in such a way that individual bits of the PSR are set or cleared following any accumulator operation, depending on the results of that operation. The contents of the process status register can be regarded as a simple number, but it is usually more informative to treat it as an eight-element array of binary flags, whose individual states show the particular effects of the last operation (a *flag* is any variable whose value indicates the state or truth-value of some condition, rather than being an absolute value. A flag variable usually has only two states or conditions: up or down, on or off, 0 or 1).

When any operation is performed on the accumulator that causes a carry out of the eighth bit, then the carry flag of the PSR will be set automatically to 1; an operation that does not cause a carry will reset (set to 0) the carry flag. This applies only to those operations that might legitimately cause a carry. Some operations, such as loading to or storing from the accumulator, do not affect the carry flag. Whenever we investigate a new Assembly language instruction in the course from now on, we shall want to know which of the PSR (or flag register) bits it affects. Naturally, we shall need to know more about the other PSR flags, but let's finish our discussion of the carry flag first.

In general, when adding two single-byte numbers we won't know in advance what they will be, so we have to be prepared for the sum of such an addition to exceed $FF; usually this will mean reserving two bytes of RAM to hold the result of an addition. Consider, again, the previous addition examples:

Hex Numbers		Carry Flag	Binary Numbers
A7	=		10100111
+ 3E	=		+ 00111110
00E5	=	0	11100101
		No Carry	
FF	=		11111111
+ FF	=		+ 11111111
01FE	=	1	111111110
		Carry	

The result of the addition is represented in both examples as a two-byte number. In the first case, the carry flag is reset to 0 because there is no carry out from the eighth bit of the sum (the two-byte result is $00E5, of which the hi-byte is $00). In the second case, however, there is a carry out from the eighth bit, so the carry flag is set, and the hi-byte of the result is $01.

To be sure of getting the correct result of an addition, therefore, we must store the accumulator contents in the lo-byte of the two-