MULTIPLE CHOICE

As we continue our investigation of Assembly language arithmetic, we consider the problems associated with subtraction, and the various ways of dealing with them. We also begin to look at the programming of multiplication in machine code, and introduce a new class of logical operations — the Shift and Rotate op-codes.

Both the Z80 and 6502 support the SBC (SuBtract with Carry) instruction, but their implementations are quite different. On the 6502, the carry flag is used to handle the *borrow* facility, which is the equivalent in subtraction of the carry facility in addition. In Z80 Assembly language, SBC works in exactly the same way as the ADC instruction — the carry flag is set or reset to indicate the result of the operation.

Suppose that we add $E4 to $5F using ADC (having cleared the carry flag first). The result in the accumulator is $43, and the carry flag is set, showing that the true result is $0143. There has been an overflow into the carry flag because the accumulator cannot contain the full result.

Now suppose that on the Z80 we again clear the carry flag, and subtract $E4 from $5F: the result in the accumulator is $7B, and the carry flag is set. If we now add $7B to $E4 (having cleared the carry flag once again) we find the result in the accumulator to be $5F, and the carry flag is set. This is entirely consistent, as can be seen:

$5F – $E4 = $7B	Carry Set
$5F = $E4 + $7B	Carry Set

If we take the carry flag's state as indicating that a negative result has occurred, then we can interpret $7B as a two's complement number:

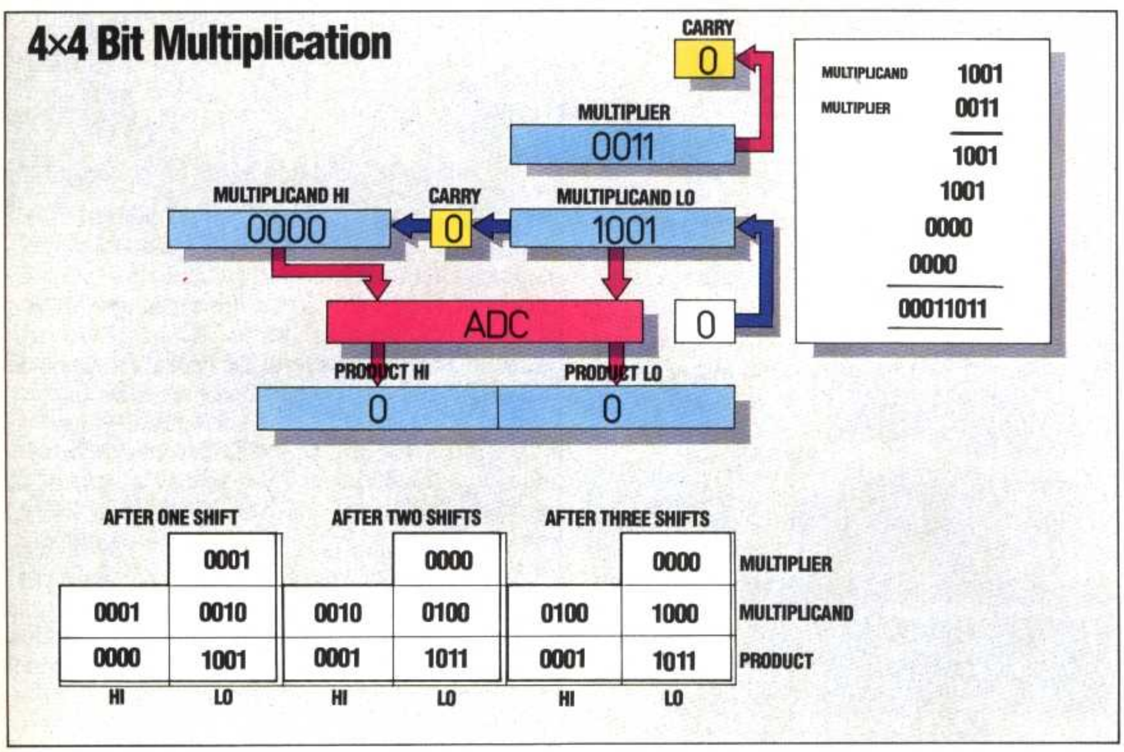
$7B In Binary		= 01111011
Take Away One		– 1
Gives One's Complement		01111010
	Negate	
Gives Two's Complement		10000101 = $85

We should expect to find, then, that $5F — $E4 results in the negative number –$85. Let's check this result in decimal:

$5F =	95 decimal
–$E4 =	228 decimal
$85 =	–133 decimal

Clearly, this all makes sense as far as it goes. Suppose now that the subtraction in question was actually a two-byte sum: $375F — $21E4.

HI	LO		
$37	5F	=	14175 decimal
–$21	E4	=	–8676 decimal
$15	7B	=	5499 decimal



Shift Times
This example shows four-bit multiplication for the sake of clarity—the number of bits does not affect the algorithm. The worked example shows how the product is formed by the addition of zeros or shifted versions of the multiplicand, depending on whether each bit of the multiplier is zero or one. The multiplier bits are right-shifted through the carry flag, while the multiplicand bits are left-shifted from lo-byte to hi-byte through the carry flag