REGISTERED ADDRESS

So far in the course we have taken a detailed look at how the CPU manipulates memory, using registers such as the accumulator and ALU. Now we can begin to look more closely at how simple procedures are performed in machine code. Here, we concentrate on the basic arithmetical operations of addition and subtraction.

The differences in operation between the Z80 and 6502 microprocessors in the way they go about performing these simple arithmetical tasks reveal the different philosophies behind their design. The Z80's many registers, with their sophisticated set of operation instructions, typify the processor itself — elegant, complex and powerful. The much simpler 6502 architecture and operation set seem to suggest an altogether humbler processor, which is robust and practical but apparently not quite in the Z80 class. This impression is accurate as far as it goes, but the 6502's wealth of addressing modes and its use of zero page as an extra index register, give it a subtlety and versatility that will enable it to dominate the home and business micro world for some time to come.

The great advantage of the Z80's registers is their flexibility — they can be treated simultaneously as both two-byte or single-byte registers, thus allowing enormous addressing scope. The 6502, on the other hand, has no two-byte registers, but is able — by way of its addressing modes — to treat zero page as an array of single-byte and two-byte registers.

ARITHMETICAL BASICS

We have seen that the CPU registers permit a variety of possible memory accesses, but manipulating memory usually requires something more than simply loading, storing and comparing its contents. The ability to perform the four operations of arithmetic is essential to a computer system, yet both the Z80 and the 6502 support only addition and subtraction. Multiplication and division must be programmed, as must the addition and subtraction of numbers larger than $FF. This is a limitation of both of the CPUs, though the educational value to the programmer of having to invent multiplication and division algorithms is enormous. On the 16-bit processors that succeeded the Z80 and 6502, however, both operations are supported, thanks to the greater speed and power of the CPUs.

We have used the ADC ('add with carry') instruction and a variety of INC ('increment') instructions, in doing single-byte arithmetic on both CPUs. Here are the two ways of adding the contents of two two-byte memory locations:

6502		Z80	
ADDR1	DW $7E60	ADDR1	DW $7E60
ADDR2	DW $4A51	ADDR2	DW $4A51
SUM	DS $03	SUM	DS $03
BEGIN	CLC	BEGIN	LD A,$00
	LDA ADDR1		AND A
	ADC ADDR2		LD HL,(ADDR1)
	STA SUM		LD DE,(ADDR2)
	LDA ADDR1+1		ADD HL,DE
	ADC ADDR2+1		LD (SUM),HL
	STA SUM+1		ADC A,$00
	LDA $00		LD (SUM+2),A
	ADC $00		RET
	STA SUM+2		
	RTS		

The single-byte method employed on the 6502 can be used on the Z80, but the register-pair method used in the Z80 version has no 6502 equivalent. Notice the strategies used to handle the various carry possibilities, starting with the CLC (6502) and AND A (Z80) instructions that clear the carry flag prior to the addition, and ending with the modification of the third byte of SUM. Allowing for the maximum result is vital in all arithmetic.

Subtraction can be treated similarly to addition, both processors having a SBC ('subtract with carry') instruction although two-byte subtraction is supported on the Z80. Because of the possibility of generating a negative result in subtraction, however, we must now begin to investigate the binary representation of algebraic sign.

To start, we need say no more about negative numbers than is implied by this statement:

If A+B = 0 then it follows that A= −B

which implies that if A is a positive number, then its negation or complement is the number which when added to A gives a result of zero. For example, if A is the single-byte number $04, then its single-byte complement is $FC:

$04+$FC=$100

Remembering that $100=$00 (if we have only a single-byte register for holding the result), this complementary representation means that subtraction can be seen as addition with negative numbers. That is:

A−B is the same as A+(−B)

Thus, $08−$05 is the same as $08+(−$05), and (−$05)=$FB (as $FB+$05=$100), which means that our original subtraction problem can be re-