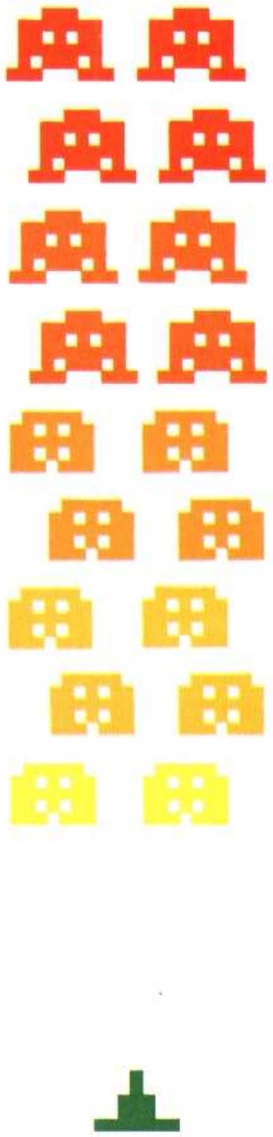
NOT SO FAST

We have often stated that the main advantage of machine code is the speed with which programs are executed. However, Assembly language programmers often find that their programs run too fast, and they need to insert time delays to slow them down. We look at the most popular methods for creating 6502 and Z80 software delays.

Delay loops can be implemented in 6502 Assembly language in several ways. The most obvious and simple method is to load one of the index registers with a value and decrement it within a loop until it reaches zero:

DELAY LOOP	TIME TAKEN FOR EACH OPERATION
LDY #$07	Two cycles
DEY	Two cycles
BNE LOOP	Two cycles (3 cycles if branch to same page; 4 cycles if branch to different page)



Each machine code instruction takes a particular number of clock cycles to execute. Information about these can usually be found with the descriptions of how the instructions operate. For example, the DEY instruction takes two cycles and LDY in immediate addressing mode also takes two cycles. As each cycle takes one microsecond (a millionth of a second), we can calculate the 'real time' taken to execute the delay loop. The total number of cycles can be calculated as follows:

1) The LDY #$07 instruction takes two cycles.
2) The program branches back seven times. Each time there is a branch back then the BNE operation takes three cycles: hence the DEY and BNE instructions take $(2+3)\times7 = 35$ cycles.
3) But the last BNE does not branch back and, therefore, takes only two cycles.

The total number of cycles is, therefore, $2+35-1 = 36$. The time taken to execute the delay is thus 36 microseconds.

There are several problems associated with using machine code delay loops to cause 'real time' delays (that is, delays that can be measured accurately in seconds or microseconds). The first, and most important, is that while a processor is executing a machine code program it regularly suspends this activity to service other parts of the system, such as scanning the keyboard, updating the internal clock, and so on. These breaks in program execution are known as 'interrupts', and two types of interrupt occur on the 6502 chip: NMI (non-maskable interrupt) and IRQ (interrupt request). The name given to the first type of interrupt implies that there is nothing that can be done to stop these interrupts occurring, but it is possible to stop IRQ interrupts that are not vital to the functioning of the processor.

IRQ interrupts can be masked by setting a particular bit in the processor status register to one. This is done by the instruction SEI. IRQ interrupts can be re-enabled by resetting the same bit using CLI. If we mask the IRQ interrupts before entering the delay loop, we can improve its accuracy. If a non-maskable interrupt occurs during execution then this will cause errors in the timing. Our original delay loop listing should be altered as follows to mask interrupts:

INSTRUCTION	FUNCTION	TIME TAKEN
SEI	Disable IRQ	Two cycles
LDY #$07		
DEY		36 cycles
BNE LOOP		
CLI	Re-enable IRQ	Two cycles

Masking the IRQs in this way adds another four cycles to the routine, which will now cause a total delay of 40 microseconds, assuming that no NMIs occur.

Another aspect of delay loops is that of 'resolution' — that is, how the time taken to execute a delay loop varies between one counter value and the next. In our example routine, we loaded the Y register with a value of seven, but if we had used a value of six instead, the delay time would have been 35 microseconds $(2 + 2 + (2+3)\times6 - 1 + 2)$. A value of five in the Y register would have taken 30 microseconds and so on to a minimum resolution of five microseconds.

We can 'fine-tune' our program (to give timings other than multiples of five) by placing NOP instructions outside the loop. An NOP instruction means the processor will perform 'No OPeration', and take two cycles to do it. If we wished to create a delay of 44 microseconds, for example, two NOP instructions could be added to our program before (or after) the loop:

SEI	Two cycles
LDY #$07	Two cycles
NOP	Two cycles
NOP	Two cycles
DEY BNE LOOP	34 cycles
CLI	Two cycles