This type of delay has an upper time limit determined by the maximum value of Y that can be used. As the Y index register is eight bits, this maximum value is 255. This gives an upper limit of 1,280 microseconds (2 + 2 + (2+3)×255 -1 + 2), or approximately one thousandth of a second. This is a long time in microprocessor terms, but not in human terms. Occasionally, we will require longer time delays. Slight improvements in the time length can be made by adding NOP instructions *within* the loop. For example, adding one NOP instruction improves the maximum delay time to 1,790 microseconds (2 + 2 + (2+2+3)×255 −1 + 2).

For substantially longer delays we must devise another method. The two most common ways of producing long delays are to use a second loop nested around the first, or decrement a larger number, say a 16-bit word made up of two bytes from memory. For each of these methods you may wish to calculate the standard of resolution that can be obtained.

NESTED LOOP COUNTER

```
DELAY SEI
      LDX #$04  ;use X reg as outer loop counter
LOOP1 LDY #$FF  ;use Y reg as inner loop counter
LOOP2 DEY
      BNE LOOP2 ;end of inner loop
      DEX
      BNE LOOP1 ;end of outer loop
      CLI
```

The inner loop of the above program takes 1,276 microseconds (2 + (2+3)×255 −1) to execute. The outer loop controls the execution of the inner loop and performs a DEX and BNE four times. The total time for this delay can be calculated as:
2 + 2 + (1,276+2+3)×4 − 1 + 2 = 5,129 microseconds.

Z80 TIME DELAYS

Each Z80 machine code instruction takes a different amount of time to execute (measured in units called 'T states'), and the Z80 runs at different speeds on different machines. To calculate the real time taken by each instruction, the number of T states for the instruction is divided by the clock frequency of the micro. For example, an instruction that takes four T states to execute on a processor with a clock frequency of 2MHz is performed in two microseconds.

Rodnay Zak's *Programming the Z80* contains timings for all the Z80 instructions. These are the CPU clock speeds for the popular Z80-based machines: ZX81 (3.25MHz); the Spectrum (3.5MHz); Tandy TRS80/Video Genie (1.7MHz); and the Amstrad (4MHz).

To perform a very small time delay, the NOP instruction can be used. This instruction, on a 2MHz micro, will give a delay of two microseconds. A number of these can be used in succession, but longer delays can be achieved by calling dummy routines; for example, the following routine will give a delay of 27 T states:

```
CALL DELAY
RET
```

In this example, the CALL instruction takes 17 T states, and the RET instruction takes 10. Thus, with a processor running at 2MHz, the delay will be 13.5 microseconds. To extend this delay slightly, NOP instructions could be included at the beginning of the routine.

To achieve longer delays, a loop needs to be used. In the following example, a register is loaded with a value, which is then decremented within a loop. The routine gives a delay of 99 T states (or 49.5 microseconds at 2 MHz).

INSTRUCTION	TIME TAKEN (T STATES)
CALL DELAY	17
(DELAY LOOP)	
LD B,5	7
DEC B	4
JR NZ,LOOP	12 (or 7 if true)

The three instructions beginning with LD B,5 are the delay loop itself. As in a 6502 machine code routine, the total time length for this routine is varied according to the value loaded into the register. The total number of clock cycles it takes to perform this code can be expressed as:

$$C = 24 + (N \times 16) - 5$$

where N is the value loaded into the B register.

Nested loop counters can also be used. But here we must take other considerations into account. Firstly, any registers used during such a routine must first be 'pushed' to preserve their contents. Secondly, some machines have hardware interrupts that will upset the timing. The maskable interrupts are disabled and reinstated by the DI and EI instructions. The following routine makes use of nested loop counters:

INSTRUCTION	FUNCTION	TIME (T STATES)
DI	Disables interrupts	4
PUSH DE	Preserves register contents	11
LD D,nn	Value of inner counter	7
LD E,nn	Value of outer counter	7
CALL OLOOP	Jump to outer counter	17
POP DE	Restore contents	10
EI	Reinstate interrupts	4
:		
(OLOOP)		
DEC E	Decrement outer loop	4
RET Z	End if zero	11 or 5
(ILOOP)		
DEC D	Decrement inner loop	4
JP Z,OLOOP	Jump if D is zero	10
JP ILOOP	Else continue with inner loop	10

In this routine, the delay is increased if the value in the E register is increased. The routine will end when a decrement is made on the E register and the result is zero. Note that if the inner loop reaches zero, and the outer loop still has a value larger than one in it, the inner loop will be initialised to 255 and the inner loop will count down to zero before control is returned to the outer loop.

Timed Invasion
Machine code timing delays are necessary in games programs, particularly when there is a moving object on the screen that the player must interact with. A classic example of this is the Space Invaders game. Without timing delays, the movement of the invading aliens would be too fast. Through carefully-controlled timing delays, movement can be controlled as necessary to make the game play properly