Ex: Write 1msec delay segment for 4 mHz oscillator.

Sln: For 1msec we need

$$\frac{1\text{msec}}{1\mu\text{sec}} = 1000 \text{ cycles}$$

1 Cycle For 4mHz OSC $= 4 \cdot \frac{1}{4\text{mHz}}$

$= 1\mu\text{sec}$.

```
movlw 250;      → 1 cycle
movwf Counter   → 1 cycle
Loop
        decfsz Counter,F;   248x1+2 cycles
        goto Loop;          248x2 cycles
```

248x1+2 + 248x2 = 745 cycles

745+2 = 751 cycles in total

If we put one NOP command before decfsz Counter,F command the Total cycles will be 751+250 = 1001 cycles

Here 1ms delay segment
For 4MHz oscillator is as follows

```
Counter equ 0x0C
    movlw 250;
    movwf Counter;
Loop
    NOP; No-Operation
    decfsz Counter,F; Decrement Counter
                      Skip if Zero
    goto Loop
```

↓ 1ms delay program/ran segment.

Ex: Write a 100msec delay segment

```
Counter_Outer  equ 0x0D
    movlw N; N can be any integer (non-negative)
    movwf Counter_Outer

Loop_Outer
    1_ms_Delay_Segment; ← Put here 1ms Delay Segment
    decfsz Counter_Outer;
    goto Loop_Outer;
```

Compute the total number of cycles in previous example for N=100 case

```
Counter_Outer   equ  0x0D
                movlw  N                    → 1 cycle
                movwf  Counter_Outer        → 1 cycle
Loop_Outer      1ms_Delay_Segment           → N x 1001 Cycles
                decfsz Counter_Outer        → (N-1)1+2
                goto Loop_Outer.            (N-1)x 2
```

Total Number of Cycles

$1+1+N\times1001+N+1+2N-2$

$= N\times1004+1$

For N=99 Total Number of Cycles = 99397

i.e, Delay amount is 99397 μsec

$\approx$ 99.5 msec

$\approx$ 100 msec

For more accurate computation 6 NOP operation can be inserted as follows

```
Counter_Outer   equ  0x0D
                movlw 100
                movwf Counter_Outer
Loop_Outer      nop nop nop nop nop nop; → 6 NOP commands are inserted
                1ms_Delay_Segment
                decfsz Counter_Outer;
                goto Loop_Outer
```

Total Number of Cycles = 99957 + 600 (NOP cycles 0x10)

= 99857 μsec

≈ 100msec

↓ More Accurate computation

Ex:

1 sec Delay Segment for 4m Hz Oscillator.

Counter_Outer equ 0x0F

movlw N ; → 1 cycle

movwf Counter_Outer; → 1 cycle

Loop Outer

1ms-Delay-Segment → 1001 cycles (×N)

decfsz Counter_Outer → (N-1)×1+2 cycles

goto Loop-Outer. → (N-1)×2 cycles

Total number of cycles = 1+1+1001N+N-1+2N-2

= 2+1001N+3N-3

= 1004N-1

For 1 sec delay you need

1000000 cycles

Maximum N = 255

1004×255 - 1 < 1000.000

Hence we need another outer Loop

Hence we can write the following segment for 1sec delay

```
Counter_Outer_Most  equ 0x0E

                    movlw 10;
                    movwf Counter_Outer_Most

Loop_O_Most

                    100ms_delay_segment
                    decfsz Counter_Outer_Most,F
                    goto loop_O_Most
```

The above segment creates almost 1sec delay

1 sec - Delay For 4 MHz oscillator. (7)

Nerze

```
Counter2  equ  0x0E
Counter1  equ  0x0D
Counter0  equ  0x0C
          movlw  10
          movwf  Counter2
Loop2     movlw  100;
          movwf  Counter1
Loop1     nop
          nop
          nop
          nop
          nop
          nop
          movlw  250;
          movwf  Counter_0
Loop0     nop
          decfsz Counter0, F
          goto   Loop0
          goto   Loop1
          goto   Loop2
```

1 ms delay

100 ms delay

1 sec delay

Subroutines

Subroutines represents a set of instructions beginning with a label and ending with the instruction return or retlw.

Subroutines are similar to those "functions" used in C++ or Java programming.

Subroutine-Name ; Label

set of instructions

//

//

return or retlw

When a subroutine is called, the content of PC which holds the address of the next instruction to be executed is saved into stack registers. A jump to subroutine segment is performed. After completion of subroutine segment PC (Program counter) is loaded back to its original content and program execution continues