RITARDI SOFTWARE

Per ritardi superiori alle unità di us, occorre usare un loop, con il quale si possono realizzare ritardi di centinaia di us.

Un ciclo macchina dura 0.5us (quattro periodi di clock): $\longrightarrow T = \frac{1}{8MHz} * 4 = 0.5us$

Per es. il ciclo formato dalle seguenti 4 istruzioni impiega **301** cicli macchina per essere eseguito da un PIC cloccato a 8MHz, cioè **150,5us**.

```
        movlw   100     <-- 1 ciclo   ; 100 = K   [ num. da mettere nel contatore,
        movwf   cont    <-- 1 ciclo                 cioè num. di ripetizioni
loop:
        decfsz  cont,1  <-- 1 ciclo   [ <-- 2 cicli se cont si azzera
        goto    loop    <-- 2 cicli     goto è saltato se cont si azzera (nessun ciclo)
```

Come risultano i 301 cicli cioè i 150,5us ?

Le prime due istruzioni durano 1 ciclo ognuna, poi il decremento e il goto vengono eseguiti 99 volte, con il decremento senza goto (2 cicli) che è eseguito una sola volta al'ultimo ciclo.

Quando il decremento non dà risultato 0 dura 1 ciclo (0.5us) seguito dal goto che dura 2 cicli (1us).
Quando invece il decremento dà risultato 0 dura 2 cicli (1us) ma il goto è saltato (nessun tempo).

Esercizio

Si vuol generare su RC6 un'onda quadra con periodo T = 400us (T/2=200us).
Calcolare il K da mettere nel contatore e scrivere il programma per il lampeggio di RC6.

Il tempo T/2=200us richiede un n. di cicli macchina pari a:

n_cicli= Tempo / tempo di ciclo ⟶ 200us / 0,5us = 400 cicli

(tempo di un ciclo)

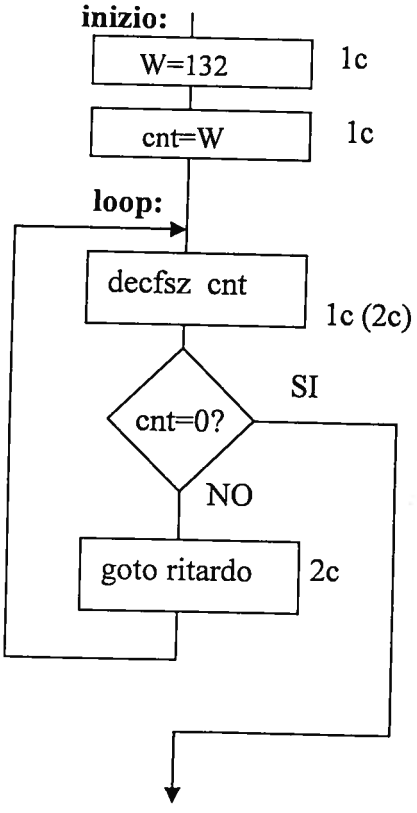
Trovo il numero K di ripetizioni, da mettere nel contatore:

K in W e W in cont — decfsz e goto — decfsz senza goto

$$n\_cicli = 2 + (3 * K) + 2 \longrightarrow 400 = 4 + (3 * K)$$

da cui si ricava K che è il n. di ripetizioni da mettere nel contatore:

$$K = \frac{400-4}{3} = 132$$



Il programma è:

```
        list        p=16f877a
        #include    <p16f877a.inc>

        banksel     TRISC
        movlw       0bfh        ; solo PC6 è uscita
        movwf       TRISC
        banksel     PORTC

inizio:
        movlw       132         ; 1c = 0,5us
        movwf       cont        ; 1c = 0,5us
ciclo:
        btfsz       cont,1      ; 1c = 0,5us se non è azzerato / 2c = 1us se è azzerato
        goto        ciclo       ; 2c = 1us  se non è azzerato / nessun tempo se è azzerato

        movlw       040f
        xorwf       PORTC
        goto        inizio
END
```

CICLI ANNIDATI

Per realizzare **ritardi di durata maggiore** del precedente, si devono nidificare i loop.

Analizziamo ad esempio il ciclo seguente, che inserisce in ritardo di circa 100ms:

```
delay:
            clrf        cont_1
            clrf        cont_2
delayloop:
            decfsz      cont_1,1
            goto        delayloop
            decfsz      cont_2,1
            goto        delayloop
        return
```

delay è l'inizio della subroutine
delayloop viene chiamato internamente e serve come inizio del ciclo di ritardo

decfsz cont_1,1 decrementa il contenuto di un registro.
Se il valore raggiunto è zero viene saltata l'istruzione successiva, mentre se il valore raggiunto non è zero viene eseguita l'istruzione successiva.
goto delayloop riporta all'inizio del ciclo di ritardo.

Una volta raggiunto lo zero con il contatore cont_1, vengono eseguite le seguenti istruzioni:

```
decfsz cont_2,1
goto   delayloop
```

che decrementano il registro seguente (cont_2) fino a che anche questo raggiunge lo zero.

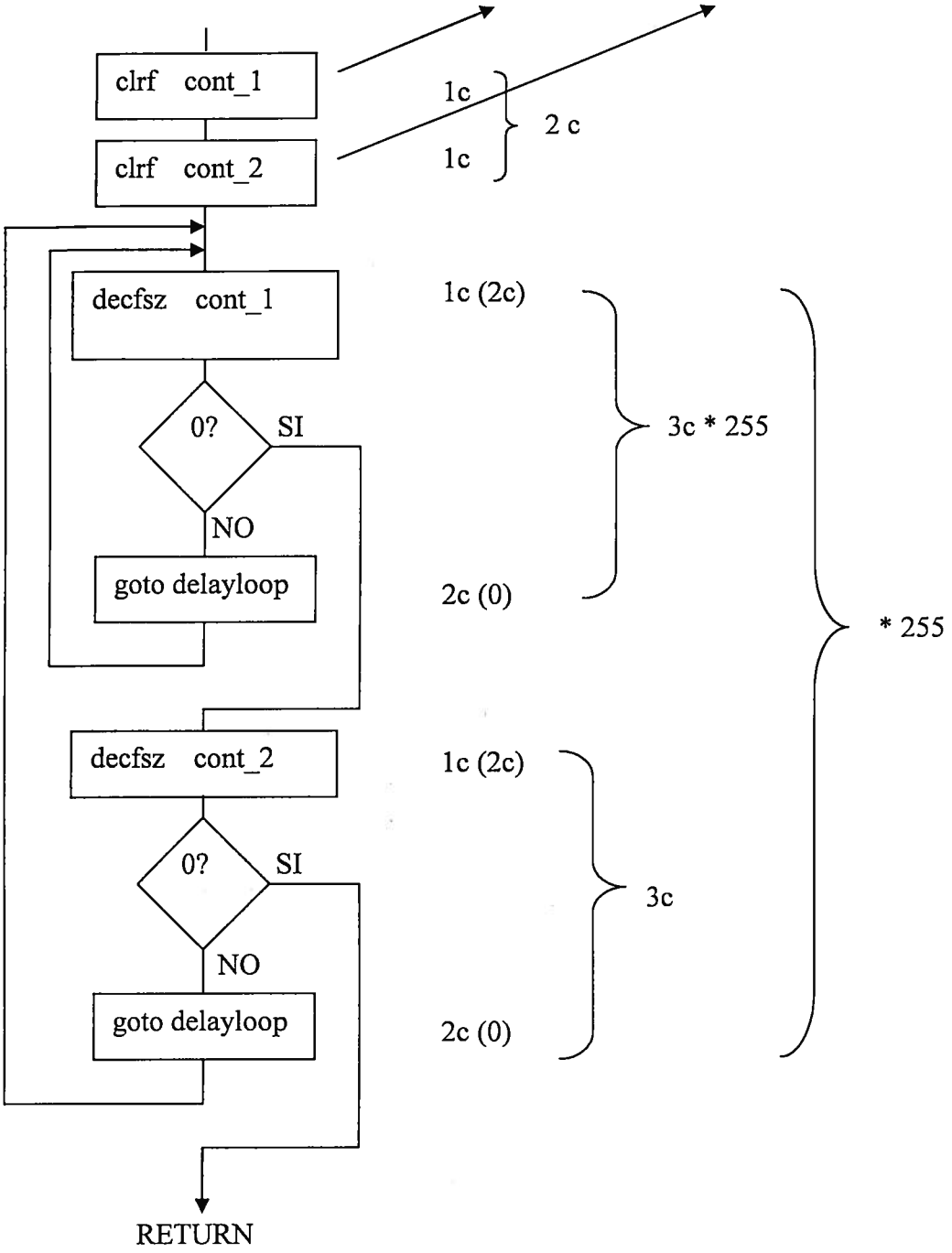
Il registro cont_2 verrà decrementato di uno ogni 256 decrementi di cont_1

Quando anche cont_2 avrà raggiunto lo zero, l'istruzione **return** determinerà l'uscita dalla routine di ritardo e il proseguimento del programma principale, a partire dall'istruzione successiva alla chiamata (call) della routine delay.

Il programma è:

```
        list            p=16f877a
        #include        <p16f877a.inc>
        Cont_1  EQU     020h
        Cont_2  EQU     021h

        banksel TRISC

        MOVLW   B'10111111'         ; solo PC0 è uscita
        MOVWF   TRISC

        banksel         PORTC

        BSF             PORTC,6     ; accendi il primo led
ciclo:

        CALL            Delay

        BTFSC           PORTC,6
        GOTO            spegni
        BSF             PORTC,6     ;accendi e torna a Loop
        GOTO            ciclo
spegni:
        BCF             PORTC,6     ;spegni e vai a Loop
        GOTO            ciclo

Delay:
        CLRF            Cont_1      ;dopo l'azzeramento i contatori
        CLRF            Cont_2      ;ripartono da 255
DelayLoop:
        DECFSZ          Cont_1,1
        GOTO            DelayLoop

        DECFSZ          Cont 1_2,1
        GOTO            DelayLoop   ;se cont_2 non è 0, riparti col primo contatore
                                    ;cont_2 si decrem di 1 ogni 256 decrem di cont_1
RETURN

END
```

Diagramma di flusso del ritardo (**azzerare i contatori significa farli ripartire da 255**)



n. cicli = 2 + { [(3 * 255 + 2) + 3] * 255}+ 2

clrf — dec+goto — dec senza goto — dec+goto cont_2 — dec senza goto cont_2

n. cicli = 2 + (770 * 255) + 2 ⟶ n. cicli = 196354

T = n. cicli * 0,5us = 196354 *0,5us = 98177us ⟶ **98,2ms**