UNIVERSIDAD NACIONAL DE LANÚS

Trabajo Práctico Final 2021

“Programación en Assembler”

Carrera:	Licenciatura en Sistemas
Comisión:	8602-1 TM (Viernes)
Materia:	Arquitectura de Computadoras
Profesor Titular:	García Roberto
Ayudante:	Bianco Santiago
Estudiante:	Giudice Matías Damian
Fecha de entrega:	17/11/2021

Proyecto: Ejercicio 1 - PUNTO E

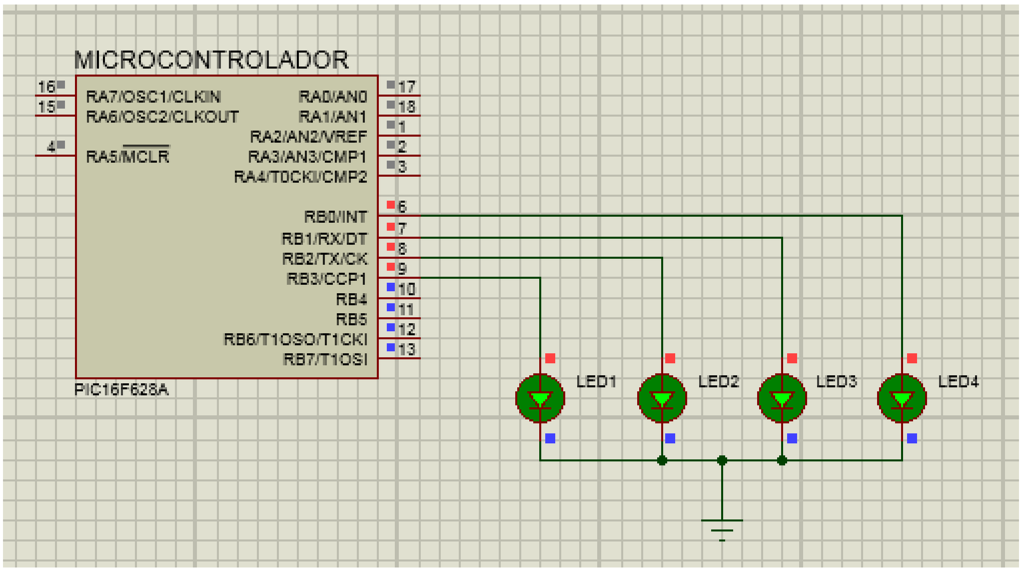
- PIC: 16F628A.
- Velocidad de reloj: Int 4 Mhz.
- Descripción: Se encenderán los leds de RB0 hasta el de RB3 y se apagarán todos los leds desde el RB3 al RB0, ambos con la misma demora, realizando todo el ciclo indefinidamente.

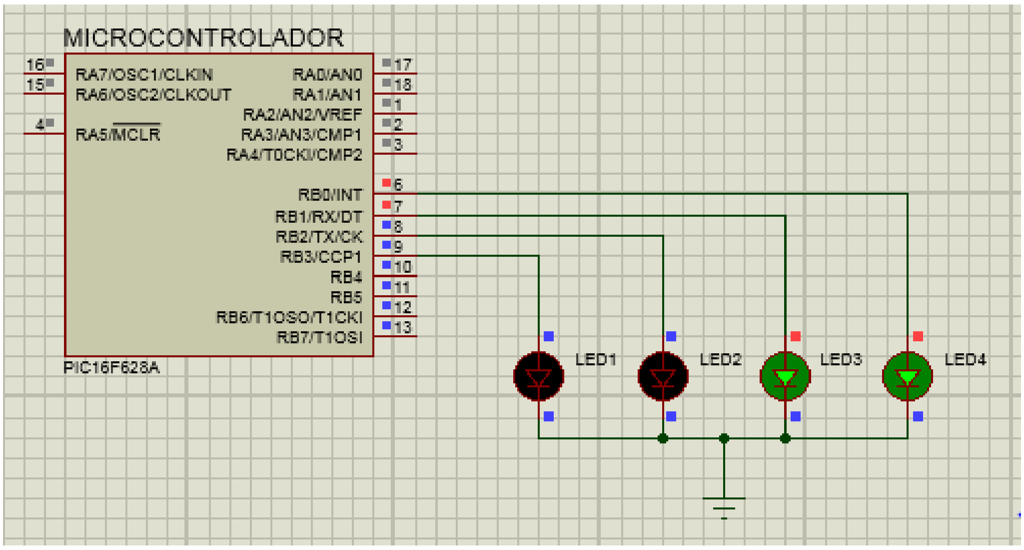
Código Assembler Comentado

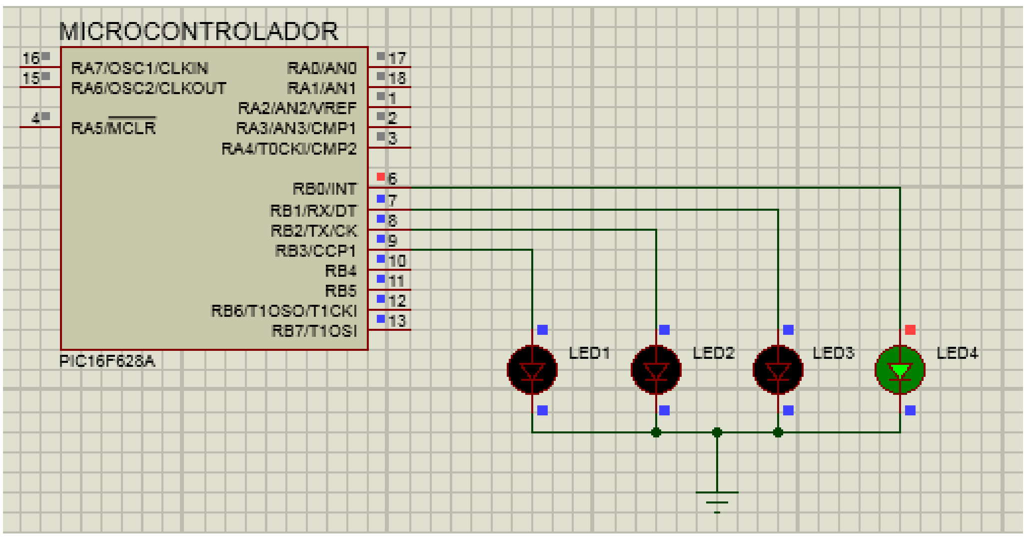
(Si no se llega a ver muy bien, también está el código fuente en la carpeta).

```
#INCLUDE <P16F628A.INC>                 ;Incluimos el PIC
    ORG 0                               ;Aca comienza el programa, se ejecuta
;---DEFINIMOS COSTANTES----------------------------------------
CONT3 EQU 0x20                          ;Posicion de memoria
CONT2 EQU 0x21                          ;Posicion de memoria
CONT1 EQU 0x22                          ;Posicion de memoria
;-------------------------------------------------------------
    BSF STATUS,RP0                      ;Moverse al banco 1
    MOVLW B'00000000'                   ;Configura el puerto B como salida
    MOVWF TRISB

    BCF STATUS,RP0                      ;Vuelve al banco 0
;---INICIO DEL PROGRAMA----------------------------------------
inicio                                  ;Inicio del programa
    MOVLW B'00000000'                   ;Pongo los puertos en 0, lo movemos a w
    MOVWF PORTB                         ;Lo que se guardo en w lo pasamos a PORTB
    BSF PORTB,0                         ;Por cada PORTB una salida individualmente la vamos prendiendo
    CALL demora_500ms                   ;Con una demora de 500ms y luego en ese orden la vamos apagando
    BSF PORTB,1
    CALL demora_500ms
    BSF PORTB,2
    CALL demora_500ms
    BSF PORTB,3
    CALL demora_500ms
    BCF PORTB,3
    CALL demora_500ms
    BCF PORTB,2
    CALL demora_500ms
    BCF PORTB,1
    CALL demora_500ms
    BCF PORTB,0
    CALL demora_500ms
    GOTO inicio                         ;Cuando termina toda la secuencia vuelve arriba, de esta forma quedaria un ciclo infinito
;-------------------------------------------------------------

demora_500ms
    movlw   .2                          ;Movemos a w el 2 en decimal
    movwf   CONT3                       ;Lo pasamos al contador (CONT3)

bucle_1 CALL    demora_250ms            ;Al entrar al bucle_1, llamamos a demora_250ms porque necesitamos un retardo de 250 y 1 microsegundo
    decfsz  CONT3,f                     ;Luego decrementamos 2 veces al contador (CONT3) con un 1 por cada ciclo, si llega a 0 se saltea el
                                        ;goto y va al return
    goto    bucle_1                     ;Mientras no llegue a 0, se vuelve a repetir el bucle_1
    return                              ;Cuando llega a 0, devulve el return

demora_250ms
    movlw   .250                        ;Movemos a w el 250 en decimal
    movwf   CONT2                       ;Lo pasamos al contador (CONT2)

bucle_2 CALL    demora_1ms              ;Al entrar al bucle_2, llamamos a demora_1ms porque necesitamos un retardo de 1 milisegundo
    decfsz  CONT2,f                     ;Le vamos decrementando al contador (CONT2) un 1 por cada ciclo, si llega a 0 se saltea el goto y va
                                        ;al return
    goto    bucle_2                     ;Mientras no llegue a 0, se vuelve a repetir el bucle_2
    return                              ;Cuando llega a 0, devulve el return
;-------------------------------------------------------------
;Consegimos 4 microsegundos
;Si esos 4 microsegundos lo multiplicamos por 250 nos da 1000 microsegundos que es 1 milisegundo
;-------------------------------------------------------------
demora_1ms
    movlw   .250                        ;Movemos a w el 250 en decimal
    movwf   CONT1                       ;Lo pasamos al contador (CONT1)

bucle_3 nop                             ;NOP es como si fuera una linea vacia(salta a la otra linea)
    decfsz  CONT1,f                     ;Le vamos decrementando al contador (CONT1) un 1 por cada ciclo, si llega a 0 se saltea el goto y va
                                        ;al return
    goto    bucle_3                     ;Mientras no llegue a 0, se vuelve a repetir el bucle_3
    return                              ;Cuando llega a 0, devulve el return
    END                                 ;Fin del programa
```

Capturas de pantalla (Proteus)







Proyecto: Ejercicio 2 - PUNTO D

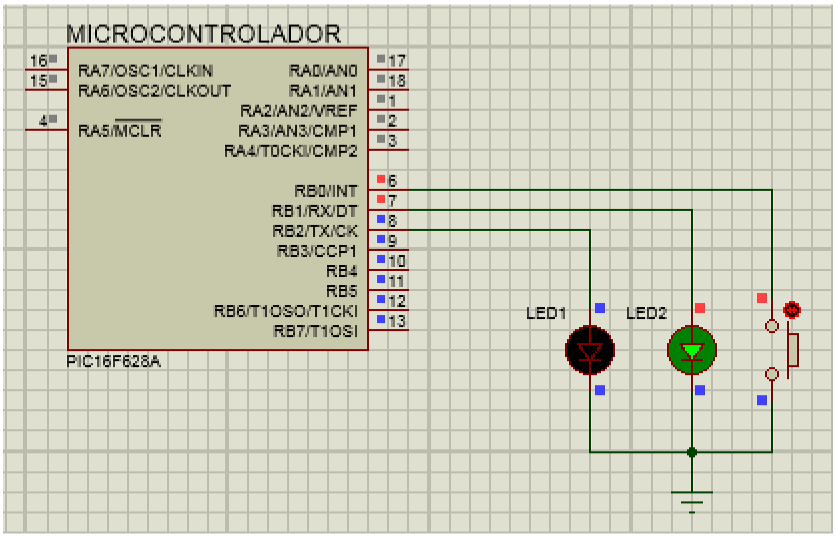
- PIC: 16F628A.
- Velocidad de reloj: Int 4 Mhz.
- Descripción: Tendrá una secuencia a medida que se presiona el botón: prender RB1, prender RB2, apagar RB1, apagar RB2. Con interrupción por RB0.

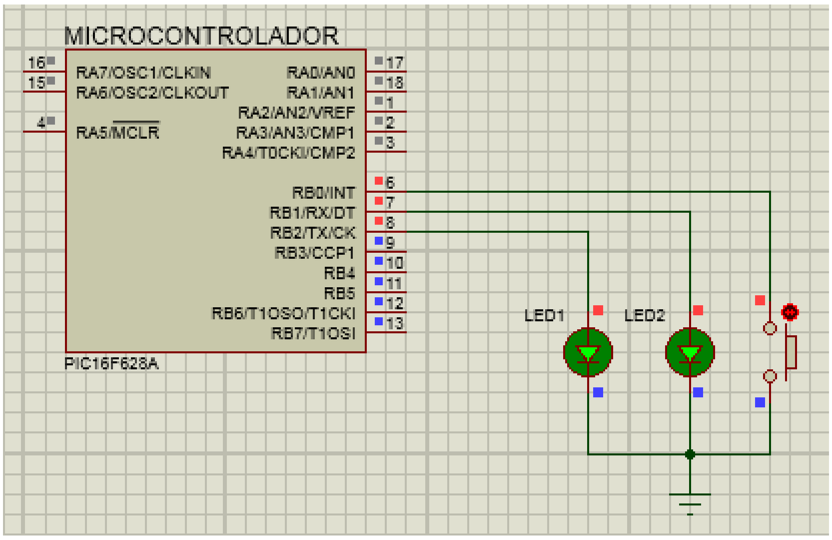
Código Assembler Comentado

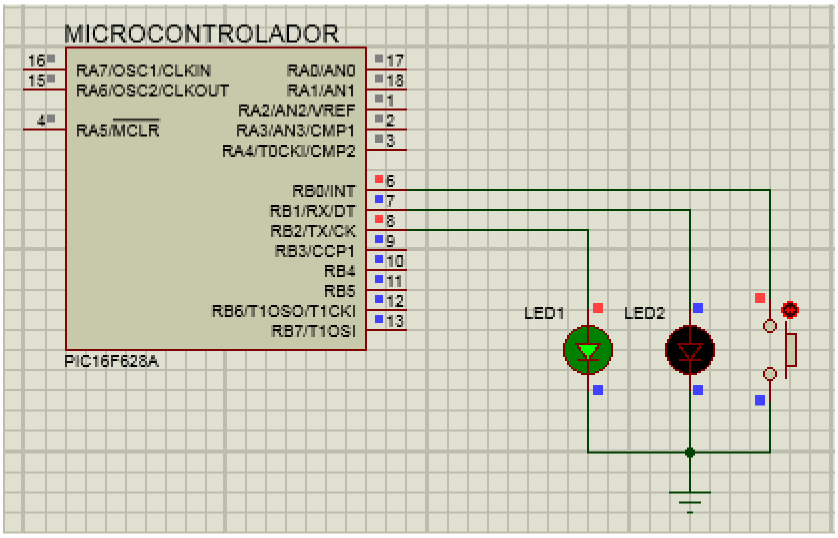
(Si no se llega a ver muy bien, también está el código fuente en la carpeta).

```
#INCLUDE <P16F628A.INC>                 ;Incluimos el PIC
    LIST P=PIC16F628A

    ORG     0x00                        ;Aca comienza el programa, se ejecuta
        GOTO    inicio                  ;Se va a mover a la etiqueta inicio
    ORG     0x04                        ;Cuando sucede una interrupcion viene a esta linea
        BTFSC   INTCON,1                ;Verifica si la flag de la interrupcion esta levantada por pulsacion de boton
                                        ;(en este caso siempre va a estar levantada)
                                        ;Al estar levantada la flag por la interrupcion por RB0 por eso el 1
        GOTO    main                    ;Se va a mover a la etiqueta main
        GOTO    end_int                 ;Se va a mover a la etiqueta end_int

main
    BTFSC   PORTB,1     ;B'00000001'    ;Verificamos los bit de los leds
    GOTO    dos                         ;En el caso que si alguno esta encendido, pasa a la siguiente etiqueta (dos)
    BTFSC   PORTB,2
    GOTO    dos
    BSF     PORTB,1                     ;En el primer caso si estan apagados ambos, se enciende el primer led
    GOTO    end_int                     ;En el caso que se encienda el led, se va a la etiqueta end_int

dos
    BTFSS   PORTB,1     ;B'00000011'    ;Verifica si el led2 esta apagado y el led1 encendido
    GOTO    tres                        ;En el caso que esto este de otra forma, pasa a la siguiente etiqueta (tres)
    BTFSC   PORTB,2
    GOTO    tres
    BSF     PORTB,2                     ;Si sucede que el led2 esta apagado y el led1 encendido, se enciende el led2
    GOTO    end_int                     ;En el caso que se encienda el led, se va a la etiqueta end_int

tres
    BTFSS   PORTB,1     ;B'00000111'    ;Verifica si ambos estan encendidos
    GOTO    cuatro                      ;En caso que no esten los ambos encendidos, significa que hay uno apagado, entonces hay que apagar el
                                        ;otro, pasa a la siguiente etiqueta (cuatro)
    BTFSS   PORTB,2
    GOTO    cuatro
    BCF     PORTB,1
    GOTO    end_int                     ;En el caso que se encienda el led1, se va a la etiqueta end_int

cuatro
    BCF     PORTB,2
    GOTO    end_int                     ;En el caso que se encienda el led1, se va a la etiqueta end_int

end_int
    BCF     INTCON,1                    ;Baja la flag y termina aca hasta que se levante otra interrupcion
    RETFIE                              ;Vuelve a la linea GOTO $


inicio
    BSF     STATUS,RP0                  ;Nos movemos al banco 1
    BCF     OPTION_REG,7                ;Si esta en 0 esta activado y 1 desactivado
    MOVLW   B'00000001'                 ;Seteamos el puerto RB0 en modo entrada y todos los demas en salida y lo guardamos en w
    MOVWF   TRISB                       ;Lo que estaba en w, ahora lo guardamos en el puerto TRISB
    BCF     OPTION_REG,7                ;Si esta en 0 esta activado y 1 desactivado
    BCF     STATUS,RP0                  ;Volvemos al banco 0

    MOVLW   B'00000000'
    MOVWF   PORTB

    BSF     INTCON,7                    ;Activamos las interrupciones globales
    BSF     INTCON,4                    ;Activamos las interrupciones por RB0
                                        ;Si sucede una interrupcion se utiliza el codigo 0x04
    GOTO    $                           ;El programa se queda esperando aca (el signo hace que se quede en esta linea, en forma de bucle)

    END                                 ;Fin del programa
```

Capturas de pantalla (Proteus)







Proyecto: Ejercicio 3 - PUNTO C

- PIC: 16F628A.
- Velocidad de reloj: Int 4 Mhz.
- Descripción: Tendrá un contador de 0 a 15 en hexadecimal con una secuencia automática, cuando llega a 15 el contador comienza a contar de forma descendente hasta 0. El resultado se mostrará en el display. La secuencia se repite indefinidamente. También se puede pausar o reanudar el proceso con el pulsador RB0, el paso entre número y número es de 1 segundo y se realiza mediante el uso del Timer 0 y la interrupción por RB0.

Código Assembler Comentado

(Si no se llega a ver muy bien, también está el código fuente en la carpeta).

```
#INCLUDE <P16F628A.INC>                 ;Incluimos el PIC
    LIST P=PIC16F628A
    __CONFIG    3F10
;---DEFINIMOS COSTANTES-----------------------------------
NUMERO     EQU  0X20                    ;Posicion de memoria
CONT_TMR0  EQU  0X21                    ;Posicion de memoria
FLAG_RESTA EQU  0X22                    ;Posicion de memoria
;-------------------------------------------------------------
    ORG     0x00                        ;Aca comienza el programa, se ejecuta
        GOTO    inicio                  ;Se va a mover a la etiqueta inicio
    ORG     0x04                        ;Cuando sucede una interrupcion viene a esta linea
        BCF     INTCON,7                ;Desabilita las interrupciones globales
        BTFSS   INTCON,1                ;Se verifica si la flag de la pulsacion del boton esta activa, en ese caso va a la etiqueta main
        GOTO    main                    ;Se va a mover a la etiqueta main
        BCF     INTCON,1                ;Baja la flag y termina aca hasta que se levante otra interrupcion

        BSF     STATUS,5                ;Nos movemos al banco 1
        MOVLW   0x20                    ;Cargamos el acumulador con 0x20
        XORWF   OPTION_REG,F            ;El contenido del acumulador con el registro f
        BCF     STATUS,5                ;Volvemos al banco 0
        GOTO    end_int                 ;Se va a mover a la etiqueta end_int

inicio
    MOVLW   .61                         ;Cargamos a w el 61 en decimal
    MOVWF   TMR0                        ;Lo que tenemos en w lo pasamos a TMR0
    BSF     STATUS,5                    ;Nos movemos al banco 1
;-------------------------------------------------------------
;Luego activamos los registros del 7 al 0
;-------------------------------------------------------------
    BCF     OPTION_REG,7                ;NOT_RBPU
    BCF     OPTION_REG,5                ;Es el clock para el TMR0
    BCF     OPTION_REG,3                ;Encargado del prescaler, al estar en 0, se le asigna a TMR0
;-------------------------------------------------------------
;Se configura el TMR0 a 50 milisegundos, usando la formula, eso quiere decir que cada 50ms, el TMR0 aumenta
;-------------------------------------------------------------
    BSF     OPTION_REG,2                ;Controla el Rate, esta en 1, el valor es 256, este valor se usa para calcular el tiempo (con formula)
    BSF     OPTION_REG,1                ;Controla el Rate, esta en 1, el valor es 256, este valor se usa para calcular el tiempo (con formula)
    BSF     OPTION_REG,0                ;Controla el Rate, esta en 1, el valor es 256, este valor se usa para calcular el tiempo (con formula)
    BSF     INTCON,7                    ;Activamos las interrupciones globales
    BSF     INTCON,5                    ;Activamos las interrupciones por TMR0
    BSF     INTCON,4                    ;Activamos las interrupciones por RB0
    MOVLW   B'00000001'                 ;Seteamos el RB0 como entrada, y lo cargamos en w
    MOVWF   TRISB                       ;Lo que tenemos en w, lo movemos a TRISB
    BCF     STATUS,5                    ;Volvemos al banco 0
```

```
    CLRF    NUMERO                      ;El contenido del registro f se pone en ceros: 0x00, en la posicion de memoria de la variable NUMERO
    CLRF    CONT_TMR0                   ;El contenido del registro f se pone en ceros: 0x00, en la posicion de memoria de la variable CONT_TMR0
    CLRF    FLAG_RESTA                  ;El contenido del registro f se pone en ceros: 0x00, en la posicion de memoria de la variable FLAG_RESTA
    CLRW                                ;El contenido del registro w se pone en ceros: 0x00
    CALL    tabla_display               ;Llama a tabla_display
    MOVWF   PORTB                       ;Al retornar, lo guardamos en el PORTB, seria que cuando lo iniciamos se marque un 0

    GOTO    $                           ;El programa se queda esperando aca (el signo hace que se quede en esta linea, en forma de bucle)

main
    INCF    CONT_TMR0,1                 ;Incrementa el contador del Timer
    MOVF    CONT_TMR0,W                 ;Lo que se va incrementando en la anterio linea, lo guarda en w
    XORLW   .20                         ;Verifica dos valores y si son iguales es 0, si son diferentes es 1
    BTFSS   STATUS,Z                    ;Levanta el flag
    GOTO    salir_timer                 ;Se va a mover a la etiqueta salir_timer
    BTFSC   FLAG_RESTA,0                ;Si la verificacion resulta falsa viene aca, si esta en 0 es suma, si esta en 1 es resta, se verifica el
                                        ;primer bit y en caso de ser suma, salta a suma o en caso de ser resta, salta a resta
    GOTO    resta                       ;Se va a mover a la etiqueta resta

suma
    INCF    NUMERO,1                    ;Incrementamos el NUMERO
    BTFSS   NUMERO,4                    ;Verificamos el 4to bit, si es 1 es porque llego a 16 y no se va a mostrar el mostrar_display, mientras eso
                                        ;no suceda va a ir a mostrar_display
    GOTO    mostrar_display             ;Se va a mover a la etiqueta mostrar_display
    COMF    FLAG_RESTA,F                ;Complemento a 1
    DECF    NUMERO,1                    ;Decrementa en 1

resta
    DECF    NUMERO,1                    ;Decrementa en 1
    BTFSC   STATUS,Z                    ;Comprueba si llega a 0, la flag z salta y se pone en 1, en caso de dar 0 salta a la siguiente linea FLAG_RESTA
    COMF    FLAG_RESTA,F                ;Vuelve hacer el complemento a 1 a la FLAG_RESTA para que que ponga todo en 0

mostrar_display
    CLRF    CONT_TMR0                   ;Limpia el TMR0, para que vuelva a empezar a contar hasta el segundo
    MOVF    NUMERO,W                    ;Mueve a w lo que haya en NUMERO
    CALL    tabla_display               ;Llama a tabla_display
    MOVWF   PORTB                       ;Al retornar, lo guardamos en el PORTB

salir_timer
    BCF     INTCON,T0IF                 ;Bajamos la flag del TMR0
    MOVLW   .61                         ;Lo volvemos a cargar
    MOVWF   TMR0                        ;Lo pasamos a TMR0

 end_int
     RETFIE                             ;Vuelve a la linea GOTO $
 ;-------------------------------------------------------------
 ;Tabla de Valores
 ;-------------------------------------------------------------
 tabla_display
     ADDWF   PCL,1                      ;Suma w + PCL + 1 (saltaria a la linea del valor que da)
     RETLW   b'01111110';0              ;Retorna el literal que esta guardado en w, y asi con los demas
     RETLW   b'00001100';1
     RETLW   b'10110110';2
     RETLW   b'10011110';3
     RETLW   b'11001100';4
     RETLW   b'11011010';5
     RETLW   b'11111010';6
     RETLW   b'00001110';7
     RETLW   b'11111110';8
     RETLW   b'11011110';9
     RETLW   b'11101110';A
     RETLW   b'11111000';B
     RETLW   b'01110010';C
     RETLW   b'10111100';D
     RETLW   b'11110010';E
     RETLW   b'11100010';F

     END                                ;Fin del programa
```

Capturas de pantalla (Proteus)