

INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

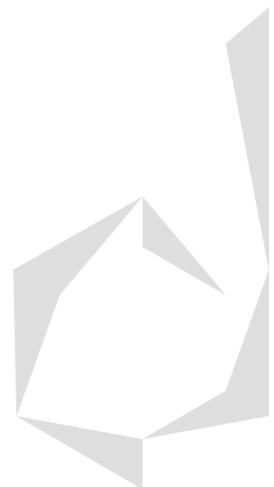
ENSAMBLADOR EN MICROCONTROLADORES

MPLAB X IDE v5.50

Ejercicios de Subrutinas,
Interrupciones y Tablas en MPLABX

Contenido

Ejercicios de Subrutinas en Ensamblador	2
13.- Nivel Superior de Pila un Subrutina de Tiempo:.....	2
14.- Subrutina de Tiempo de 1 Variable - Retardo de 10 ms:.....	4
15.- Subrutina de Tiempo de 2 Variables - Retardo de 10 ms:	9
16.- Rotación de 1 Bit por un Puerto - Retardo de 400 ms:.....	10
17.- Pulso PWM con Frecuencia de 1kHz y Tiempo Activo del 5%:	11
18.- Pulso PWM con Frecuencia de 1kHz y Tiempo Activo del 40%:	11
19.- Contador Hexadecimal de 0 a F con Display de 7 Segmentos:	12
20.- Contador de 0 a 99 en Displays de 7 Segmentos de Ánodo Común:	14
21.- Contador de 0 a 99 en Displays de 7 Segmentos de Cátodo Común:	15
Ejercicios de Interrupciones y Tablas.....	16
22.- Contador de 0 a 99 en Displays de 7 Segmentos de Ánodo Común:	16
23.- Contador de 0 a 99 en Displays de 7 Segmentos de Cátodo Común:	20
24.- Contador de 0 a 99 en Displays de 7 Segmentos Usando un ADC:	21
25.- Cronómetro de 6 Displays con Control de Dirección, Reset y Pausa:	23
26.- Control de Pantalla LCD:	29
27.- Melodía en Buzzer Pasivo:	42



Ejercicios de Subrutinas en Ensamblador

13.-Nivel Superior de Pila un Subrutina de Tiempo:

13.- Realiza una subrutina que guarde un valor del contador de programa (PC) en un nivel superior al nivel 0 de la Pila.

```
;13.- Realiza una subrutina que guarde un valor del contador de programa (PC)
;en un nivel superior al nivel 0 de la Pila.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder
;hacerlo debo poner la dirección de la carpeta que llega hasta donde se
;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:
INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\Configuracion_PIC.asm>
;Lo que hace la configuración es indicar el PIC que estoy usando, declarar las
;2 palabras de configuración, usar otra directiva INCLUDE para jalar el archivo
;PIC16F887.INC que incluye las 35 instrucciones del PIC junto con las directivas
;EQU (los nombres) de sus registros, la directiva ORG que indica la dirección de
;la memoria FLASH desde donde se empezará a guardar el código, el limpiado
;(poner en 0) de todos los bits de los puertos (A, B, C, D y E) y hacer que los
;puertos A, B y E (que son los únicos que pueden ser analógicos o digitales)
;sean todos entradas o salidas digitales.

;Ahora si ya puedo resolver el ejercicio:
;13.- Realiza una subrutina que guarde un valor del contador de programa (PC)
;en un nivel superior al nivel 0 de la Pila.

;TODAS LAS SUBROUTINAS LAS DEBO DECLARAR HASTA LA PARTE DE ABAJO EN MI PROGRAMA.
;UNA SUBROUTINA ES COMO UNA FUNCIÓN EN CUALQUIER OTRO LENGUAJE DE PROGRAMACIÓN,
;YA QUE SIRVE PARA ESCRIBIR CÓDIGO QUE SE REUTILIZARÁ VARIAS VECES DENTRO DE UN
;MISMO PROGRAMA, SE LE DEBE ASIGNAR UN NOMBRE A LA SUBROUTINA (O DIRECTIVA EQU)
;Y CUANDO LO QUIERA USAR SIMPLEMENTE DEBO USAR LA INSTRUCCIÓN CALL Y EL NOMBRE
;DE LA SUBROUTINA.
;AL UTILIZAR LA INSTRUCCIÓN CALL SE MODIFICA EL ESTADO DE LA PILA, QUE ES UN
;FRAGMENTO DE LA MEMORIA RAM DISEÑADO PARA GUARDAR VALORES DEL CONTADOR DE
;PROGRAMA (PC) PARA APUNTAR A DIRECCIONES DE LA MEMORIA FLASH, OSEA A LÍNEAS
;ESPECÍFICAS DE MI CÓDIGO, LA PILA EN EL PIC16F887 CONSTA DE 8 NIVELES, POR LO
;QUE SOLO PODREMOS DECLARAR SUBROUTINAS QUE ANIDEN 8 SUBROUTINAS DENTRO DE ELLAS,
;AUNQUE PODREMOS CREAR LAS SUBROUTINAS QUE QUERRAMOS SIEMPRE Y CUANDO SE RESPETE
;ESTA CONDICIÓN. LAS ACCIONES QUE HACE EL PROGRAMA CUANDO SE USA CALL SON:
;Almacena en un nivel de la Pila el valor que le sigue al PC que hay
;donde se usó la instrucción CALL, en este caso como CALL se usa en la
;línea 58 del código, se guardaría el valor 59 en el nivel 0 de la pila,
;pero como en ese nivel hay un comentario, se guarda en la pila el valor
;del PC donde se encuentra la siguiente línea de código, osea el 67 en la
;columna Location de la Pila.

;Sube el nivel de la Pila, que cuenta del 0 al 7, en este caso subirá al
;nivel 1.

;Carga en el PC el valor de donde se encuentra la subrutina que se quiere
;usar, en este caso como la subrutina se declaró hasta el final del código
;en la línea 167, ese valor se carga al PC.
;PARA SIMULAR EL COMPORTAMIENTO DE LA PILA DEBO USAR LA HERRAMIENTA STOPWATCH
;DE MPLABX Y SUS BREAKPOINTS, EN ESPECÍFICO SI QUIERO VER LOS NIVELES DE LA
;PILA QUE ALCANZA CADA SUBROUTINA DEBO PONER UN BREAKPOINT JUSTO EN SU
;INSTRUCCIÓN RETURN
;CALL k: Sirve para llamar una subrutina por su nombre y ejecutar
;una acción que se pueda repetir varias dentro del mismo código, de
;esta manera esa acción no la debo escribir varias veces.
INICIO: CALL SUBROUTINA ;Llama la subrutina que no tiene otra anidada.
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm). En los microcontroladores 1 ciclo de
```



```

;máquina equivale a 4 ciclos del oscilador (o señal de reloj). Si
;estamos utilizando el oscilador interno del PIC16F887 sin activar
;el divisor de reloj, la frecuencia del reloj es de 4MHz, por lo que
;1 cm = 4*(1/4X10^6) = 1 microsegundo = 1 us
NOP

;CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
CALL SUBROUTINA1 ;Llama la subrutina que tiene otra anidada.
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm).
NOP

;CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
CALL SUB_F ;Llama la subrutina que llega al nivel 7 de la pila.
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm).
NOP

;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO INICIO
;Todos los códigos en ensamblador deben tener al menos una
;instrucción GOTO cuya función sea ocasionar que el PIC repita su
;función indefinidamente.

;-----HASTA ABAJO DEL PROGRAMA SE DECLARAN TODAS LAS SUBROUTINAS-----
;ESTA ES UNA SUBROUTINA NO ANIDADA, QUE SOLO ALCANZARÁ A GUARDAR UN VALOR DE PC
;EN EL NIVEL 0 DE LA PILA Y HARÁ QUE BRINQUE AL NIVEL 1 CON LA INSTRUCCIÓN CALL,
;PARA QUE LUEGO AL LLEGAR A LA INSTRUCCIÓN RETURN REGRESE LA PILA AL NIVEL 0.
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm).
SUBROUTINA: NOP
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila
;y cargar en el PC el valor que esté guardado ahí, haciendo que el
;programa brinque a la parte del código después de la instrucción
;CALL que llamó esta subrutina, siguiendo así la ejecución normal
;que llevaba el programa.
RETURN

;ESTA ES UNA SUBROUTINA CON OTRA ANIDADA, QUE ALCANZARÁ A GUARDAR UN VALOR DE PC
;EN EL NIVEL 0 DE LA PILA Y HARÁ QUE BRINQUE AL NIVEL 1 CON LA INSTRUCCIÓN
;CALL, LUEGO LLAMARÁ LA SUBROUTINA ANIDADA Y ÉSTA OTRA ESCRIBIRÁ EN EL NIVEL 1
;DE LA PILA OTRO VALOR DE PC Y SUBIRÁ AL NIVEL 2 CON LA SEGUNDA INSTRUCCIÓN CALL,
;DESPUÉS AL LLEGAR A LA INSTRUCCIÓN RETURN DE LA SUBROUTINA ANIDADA, HARÁ QUE LA
;PILA REGRESE AL NIVEL 1 DE LA PILA Y AL LLEGAR A LA INSTRUCCIÓN RETURN DE LA
;PRIMERA SUBROUTINA REGRESARÁ AL NIVEL 0.
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm).
SUBROUTINA1: NOP
;CUANDO DENTRO DE UNA SUBROUTINA SE LLAME A OTRA, A LA SEGUNDA SUBROUTINA SE LE
;DICE QUE ESTÁ ANIDADA DENTRO DE LA PRIMERA Y AL ANIDAR UNA SUBROUTINA LO QUE
;ESTAMOS HACIENDO ES ESCRIBIR EN NIVELES SUPERIORES DE LA PILA, DEBEMOS TENER
;CUIDADO AL HACER ESTO PORQUE SI ANIDAMOS MÁS DE 8 SUBROUTINAS DENTRO DE UNA
;MISMA, ESTAREMOS DESBORDANDO LA PILA, APUNTANDO A UN NIVEL QUE NO EXISTE, YA
;QUE LA PILA DEL PIC16F887 CUENTA CON 8 NIVELES, PODEMOS VER EL NIVEL DE LA PILA
;QUE SE ESTÁ UTILIZANDO DENTRO DE MPLAB CON EL SIMULADOR ABRIENDO LA VENTANA:
;Window->Target Memory Views->Hardware Stack
;LA FLECHITA VERDE QUE APUNTA AL NIVEL DE PILA DONDE SE VA A ESCRIBIR EL
;SIGUIENTE VALOR DEL PC, SE LLAMA STACK POINTER Y SI ESTE LLEGA A APUNTAR UN
;NÚMERO MAYOR A 7, ES PORQUE EXISTEN MÁS DE 8 SUBROUTINAS ANIDADAS DENTRO DE UNA
;Y EL PROGRAMA FALLARÁ FATALMENTE YA QUE ESTARÁ APUNTANDO A UN NIVEL DE LA PILA
;QUE NO EXISTE.
;CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
CALL SUBROUTINA2
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila

```

```

; y cargar en el PC el valor que esté guardado ahí, haciendo que el
; programa brinque a la parte del código después de la instrucción
; CALL que llamó esta subrutina, siguiendo así la ejecución normal
; que llevaba el programa.
RETURN

; ESTA ES LA SUBROUTINA QUE ESTÁ ANIDADA DENTRO DE LA SUBROUTINA 1 Y HACE QUE LA
; PILA SUBA AL NIVEL 2, AUNQUE NO ESCRIBE NADA DENTRO DE ÉL.
SUBROUTINA2: ;NOP: Esta instrucción no hace nada, solamente sirve para dejar
; pasar 1 ciclo de máquina (cm).
NOP
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return, lo que hace RETURN es bajar un nivel a la pila
; y cargar en el PC el valor que esté guardado ahí, haciendo que el
; programa brinque a la parte del código después de la instrucción
; CALL que llamó esta subrutina, siguiendo así la ejecución normal
; que llevaba el programa.
RETURN

; SUBROUTINA QUE LLEGA AL ÚLTIMO NIVEL DE LA PILA, QUE VA DEL 0 AL 7.
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F: CALL SUB_F1
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F1: CALL SUB_F2
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F2: CALL SUB_F3
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F3: CALL SUB_F4
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F4: CALL SUB_F5
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; CALL k: Sirve para ejecutar una subrutina, indicada por su nombre.
SUB_F5: CALL SUB_F6
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN
; NOP: Esta instrucción no hace nada, solamente sirve para dejar
; pasar 1 ciclo de máquina (cm).
SUB_F6: NOP
; RETURN: Todas las subrutinas siempre deben terminar con la
; instrucción return.
RETURN

; SI PONGO UNA SUBROUTINA MÁS ANIDADA, LO QUE PASARÁ ES QUE SE DESBORDARÁ LA PILA
; Y ESTO LO PODRÉ NOTAR PORQUE AL SIMULAR MI CÓDIGO EL STACK POINTER ESTARÁ
; APUNTANDO A UN NIVEL QUE NO EXISTE Y NINGÚN NIVEL DE LA PILA ESTARÁ MARCADO EN
; COLOR VERDE.

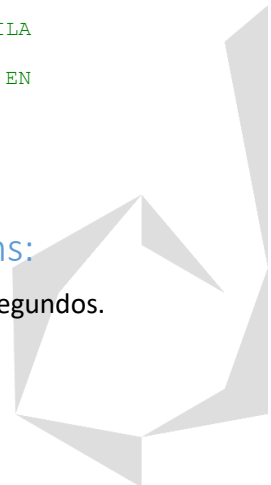
; Los programas en ensamblador deben acabar con la directiva END.
END

```

14.- Subrutina de Tiempo de 1 Variable - Retardo de 10 ms:

14.- Realiza una subrutina de tiempo de 1 variable que cree un retardo de 10 milisegundos.

;14.- Realiza una subrutina de tiempo de 1 variable que cree un retardo de



```

;10 milisegundos.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder
;hacerlo debo poner la dirección de la carpeta que llega hasta donde se
;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:
    INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\Configuracion_PIC.asm>
;Lo que hace la configuración es indicar el PIC que estoy usando, declarar las
;2 palabras de configuración, usar otra directiva INCLUDE para jalar el archivo
;P16F887.INC que incluye las 35 instrucciones del PIC junto con las directivas
;EQU (los nombres) de sus registros, la directiva ORG que indica la dirección de
;la memoria FLASH desde donde se empezará a guardar el código, el limpiado
;(poner en 0) de todos los bits de los puertos (A, B, C, D y E) y hacer que los
;puertos A, B y E (que son los únicos que pueden ser analógicos o digitales)
;sean todos entradas o salidas digitales.

;Ahora si ya puedo resolver el ejercicio:
;14.- Realiza una subrutina de tiempo de 1 variable que cree un retardo de
;10 milisegundos.

;ME DEBO IR AL BANCO 0 PARA PODER ACCEDER A SUS REGISTROS DE PROPÓSITO GENERAL
;Y AHÍ PODER GUARDAR EL VALOR DE LAS VARIABLES DE MIS SUBROUTINAS DE TIEMPO
INICIO:    BCF      STATUS, RP1 ;RP1 = 0
          BCF      STATUS, RP0 ;RP0 = 0
          ;Con esto ya estoy en el banco 0
          ;Banco 0: RP1 = 0, RP0 = 0
          ;Banco 1: RP1 = 0, RP0 = 1
          ;Banco 2: RP1 = 1, RP0 = 0
          ;Banco 3: RP1 = 1, RP0 = 1

;TODAS LAS SUBROUTINAS LAS DEBO DECLARAR HASTA LA PARTE DE ABAJO EN MI PROGRAMA.
;Una subrutina es como una función en cualquier otro lenguaje de programación,
;ya que sirve para escribir código que se reutilizará varias veces dentro de un
;mismo programa, se le debe asignar un nombre (directiva EQU) a la subrutina
;y cuando la quiera usar simplemente debo usar la instrucción CALL y el nombre
;de la subrutina. Las subrutinas afectan el estado de la Pila y solo se pueden
;anidar 6 subrutinas dentro de una misma (osea una subrutina de 7 niveles), sino
;la Pila se desbordará de sus 8 niveles y el programa fallará catastróficamente.

;PARA SIMULAR EL COMPORTAMIENTO DE LAS SUBROUTINAS DEBO USAR LA HERRAMIENTA
;STOPWATCH DE MPLABX Y SUS BREAKPOINTS, YA SEA PARA VER EL ESTADO DE LOS NIVELES
;EN LA PILA QUE ALCANZA CADA SUBROUTINA PONIENDO UN BREAKPOINT JUSTO EN LA
;INSTRUCCIÓN RETURN DE CADA UNA O PARA MEDIR EL TIEMPO DE CADA SUBROUTINA DE
;TIEMPO VIENDO LA VENTANA DEL STOPWATCH.

;LAS SUBROUTINAS DE TIEMPO SE USAN PARA DEJAR PASAR CIERTO TIEMPO DESPUÉS DE QUE
;SE EJECUTE UNA LÍNEA DE CÓDIGO, A ESTO SE LE LLAMA MANEJO DE TIEMPOS Y ES MUY
;IMPORTANTE EN LAS TAREAS DEL PIC. DEPENDIENDO DEL TIEMPO DE RETARDO QUE QUIERA
;ALCANZAR USARÉ SUBROUTINAS DE TIEMPO DE 1, 2 O 3 VARIABLES.

;SUBROUTINA DE TIEMPO DE 1 VARIABLE: Abarca un tiempo de retardo de 11 us
;(microsegundos) a 1,541 us = 1.541 ms (milisegundos).
;Las variables de mi subrutina se pueden colocar en los registros de
;propósito general que quiera, pero para tener un orden, las variables 1, 2
;o 3 de las subrutinas de tiempo las guardaré en los registros de propósito
;general que vayan de la dirección 0X60 a la 0X68 del banco 0 de la memoria
;RAM en el PIC y las declararé en forma decimal de la siguiente manera:
;.numero_decimal
;Se usa el siguiente registro de propósito general para guardar la variable:
;Registro 0X60 para guardar la variable 1.
;MOVLW K: Coloca directamente en el acumulador W un número binario
;cualquiera de 8 bits dado por el valor literal K. Como en este caso
;lo usaré para colocar una variable de mi subrutina de tiempo, se
;declara en forma decimal, poniendo .numero_decimal.
MOVLW .25 ;W = var1 = .25 = 25 decimal
;MOVWF F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F. En este el
;contenido del acumulador W se coloca en un registro de propósito
;general F que sirve para guardar un valor en la memoria RAM (que se

```

```

;borrará cuando se reinicie el PIC), indicado en forma hexadecimal,
;poniendo 0Xnúmero_hexadecimal.
MOVWF 0X60 ;0X60 = W = var1 = 25 decimal
;CALL k: Sirve para llamar una subrutina por su nombre y ejecutar
;una acción que se pueda repetir varias dentro del mismo código, de
;esta manera esa acción no la debo escribir varias veces.
CALL ST1V ;Llama la Subrutina de Tiempo de 1 Variable (ST1V).

;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO INICIO
;Todos los códigos en ensamblador deben tener al menos una
;instrucción GOTO cuya función sea ocasionar que el PIC repita su
;función indefinidamente.

;-----HASTA ABAJO DEL PROGRAMA SE DECLARAN TODAS LAS SUBROUTINAS-----
;PARA CREAR UNA SUBROUTINA DE TIEMPO LO QUE SE HACE ES CREAR UN CONTADOR QUE
;CUENTE EL TIEMPO DE RETARDO DESEADO, DADO POR EL VALOR #cm EN LAS 3 ECUACIONES
;DE LAS SUBROUTINAS DE TIEMPO DE 1, 2 O 3 VARIABLES, DEPENDIENDO DEL TIEMPO QUE
;SE QUIERA ALCANZAR SE USARÁ UNA SUBROUTINA EN ESPECÍFICO, YA QUE MIENTRAS MAYOR
;SEA EL TIEMPO DE RETARDO, MAYOR DEBERÁ SER EL NÚMERO DE VARIABLES EN LA
;SUBROUTINA.
;EN LAS ECUACIONES DE LAS SUBROUTINAS DE TIEMPO DE 1, 2 Y 3 VARIABLES SE
;CONSIDERA QUE EN LOS MICROCONTROLADORES UN CICLO DE MÁQUINA (cm) EQUIVALE A
;4 CICLOS DEL OSCILADOR Y SI ESTAMOS UTILIZANDO EL OSCILADOR INTERNO DEL
;PIC16F887 SIN UTILIZAR EL DIVISOR DE RELOJ, LA FRECUENCIA DEL RELOJ ES DE 4MHZ,
;POR LO QUE:
;1 cm = 4*(1/4X10^6) = 1 microsegundo = 1 us
;*****
;SUBROUTINA DE TIEMPO DE 1 VARIABLE: Puede abarcar tiempos de retardo de
;11 us (microsegundos) a 1.541 ms (milisegundos) ya que var1 en su ecuación
;solo puede adoptar valores de 1 a 256.
;SE USA EL REGISTRO DE PROPÓSITO GENERAL 0X60.
;Ecuación: En la ecuación, #cm es el tiempo de retardo que quiero obtener y
;lo debo introducir en microsegundos (us).
;#cm = 5 + (var1)(#NOPS + 3)
;-----Bucle con var1 guardada en el registro 0X60-----
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm). Se puede declarar un número
;cualquiera de NOPS pero en el curso dentro de las subrutinas de
;tiempo siempre usaremos #NOPS = 3 para que sea un dato conocido.
ST1V: NOP
NOP
NOP ;#NOPS = 3
;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;UN BUCLE PARECIDO AL CICLO FOR Y SE USA PARA CREAR UN CONTADOR.
;DECFSZ F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincaré la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.
;Dura 1 ciclo de máquina su ejecución.

;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;Dura 2 ciclos de máquina su ejecución.
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 1 VARIABLE.
DECFSZ 0X60,F
;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO ST1V
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila
;y cargar en el PC el valor que esté guardado ahí, haciendo que el

```



```

;programa brinque a la parte del código después de la instrucción
;CALL que llamó esta subrutina, siguiendo así la ejecución normal
;que llevaba el programa.
;-----Bucle con var1 guardada en el registro 0X60-----
RETURN
;*****
;SUBROUTINA DE TIEMPO DE 2 VARIABLES: Puede abarcar tiempos de retardo de
;17 us (microsegundos) a 394.247 ms (milisegundos) ya que var1 y var2 en su
;ecuación solo puede adoptar valores de 1 a 256.
;LA SUBROUTINA DE TIEMPO DE 2 VARIABLES SE CREA ANIDANDO LA DE 1 VARIABLE DENTRO
;DE OTRA Y SE USAN LOS REGISTROS DE PROPÓSITO GENERAL 0X61, 0X62 y 0X63.
;Ecuación: En la ecuación, #cm es el tiempo de retardo que quiero obtener y
;lo debo introducir en microsegundos (us).
;#cm = 7 + var2 * (4 + (nops+3) * var1)
;-----Bucle con var2 guardada en el registro 0X61-----
;MOVWF F, D: Lee el contenido de un registro de la RAM indicado
;por la dirección F y lo coloca en el mismo registro F o en el
;acumulador W, dependiendo de la directiva EQU que ponga en donde
;dice D.
ST2V: MOVWF 0X62,W ;W = 0X62 = var1
;MOVWF F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F. En este caso el
;contenido del acumulador W se coloca en un registro de propósito
;general F que sirve para guardar un valor en la memoria RAM (que se
;borrará cuando se reinicie el PIC), indicado en forma hexadecimal,
;poniendo 0Xnúmero hexadecimal para la subrutina de tiempo.
MOVWF 0X63 ;Guarda una copia de la variable 1
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm). Se puede declarar un número
;cualquiera de NOPS pero en el curso dentro de las subrutinas de
;tiempo siempre usaremos #NOPS = 3 para que sea un dato conocido.
;-----Bucle con la copia de var1 guardada en el registro 0X63-----
ST2V_V1: NOP
NOP
NOP ;#NOPS = 3
;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;UN BUCLE PARECIDO AL CICLO FOR Y SE USA PARA CREAR UN CONTADOR.
;DECFSZ F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincaré la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.

;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 1 VARIABLE.
DECFSZ 0X63,F ;Haz un bucle con la copia de var1
;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO ST2V_V1
;-----Bucle con la copia de var1 guardada en el registro 0X63-----
;DECFSZ F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincaré la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.

;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 2 VARIABLES.
DECFSZ 0X61,F ;Haz un bucle con var2
;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.

```




```

GOTO      ST2V
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila
;y cargar en el PC el valor que esté guardado ahí, haciendo que el
;programa brinque a la parte del código después de la instrucción
;CALL que llamó esta subrutina, siguiendo así la ejecución normal
;que llevaba el programa.
;-----Bucle con var2 guardada en el registro 0X61-----
RETURN
;*****
;SUBROUTINA DE TIEMPO DE 3 VARIABLES: Puede abarcar tiempos de retardo de
;23 us (microsegundos) a 100.926 s (segundos) = 1.6821 min (minutos) ya que
;var1, var2 y var3 pueden adoptar valores de 1 a 256 en su ecuación.
;LA SUBROUTINA DE TIEMPO DE 3 VARIABLES SE CREA ANIDANDO LA DE 2 VARIABLES y DE 1
;VARIABLE DENTRO DE OTRA Y SE USAN LOS REGISTROS DE PROPÓSITO GENERAL 0X64, 0X65
;0X66, 0X67 y 0X68.
;Ecuación: En la ecuación, #cm es el tiempo de retardo que quiero obtener y
;lo debo introducir en microsegundos (us).
;#cm = 9 + var1 * (var3 * (var2 * (nops+3) + 4) + 4)
;-----Bucle con var3 guardada en el registro 0X64-----
;MOVF    F, D: Lee el contenido de un registro de la RAM indicado
;por la dirección F y lo coloca en el mismo registro F o en el
;acumulador W, dependiendo de la directiva EQU que ponga en donde
;dice D.
ST3V:      MOVF      0X66,W ;W = 0X66 = var1
;MOVWF    F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F. En este caso el
;contenido del acumulador W se coloca en un registro de propósito
;general F que sirve para guardar un valor en la memoria RAM (que se
;borrará cuando se reinicie el PIC), indicado en forma hexadecimal,
;poniendo 0Xnúmero_hexadecimal para la subrutina de tiempo.
MOVWF      0X67 ;0X67 = W = var1, copia de la variable 1
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm). Se puede declarar un número
;cualquiera de NOPS pero en el curso dentro de las subrutinas de
;tiempo siempre usaremos #NOPS = 3 para que sea un dato conocido.
;-----Bucle con la copia de var1 guardada en el registro 0X67-----
;MOVF    F, D: Lee el contenido de un registro de la RAM indicado
;por la dirección F y lo coloca en el mismo registro F o en el
;acumulador W, dependiendo de la directiva EQU que ponga en donde
;dice D.
ST3V_V1:   MOVF      0X65,W ;W = 0X65 = var2
;MOVWF    F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F. En este caso el
;contenido del acumulador W se coloca en un registro de propósito
;general F que sirve para guardar un valor en la memoria RAM (que se
;borrará cuando se reinicie el PIC), indicado en forma hexadecimal,
;poniendo 0Xnúmero_hexadecimal para la subrutina de tiempo.
MOVWF      0X68 ;0X68 = W = var2, copia de la variable 2
;NOP: Esta instrucción no hace nada, solamente sirve para dejar
;pasar 1 ciclo de máquina (cm). Se puede declarar un número
;cualquiera de NOPS pero en el curso dentro de las subrutinas de
;tiempo siempre usaremos #NOPS = 3 para que sea un dato conocido.
ST3V_V2:   NOP
NOP
NOP ;#NOPS = 3
;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;UN BUCLE PARECIDO AL CICLO FOR Y SE USA PARA CREAR UN CONTADOR.
;DECFSZ    F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincaré la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.

;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 1 VARIABLE.
DECFSZ     0X68,F ;Haz un bucle con la copia de var2

```



```

;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO ST3V_V2

;DECFSZ F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincará la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.

;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 2 VARIABLES.
DECFSZ 0X67,F ;Haz un bucle con la copia de var1
;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO ST3V_V1
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila
;y cargar en el PC el valor que esté guardado ahí, haciendo que el
;programa brinque a la parte del código después de la instrucción
;CALL que llamó esta subrutina, siguiendo así la ejecución normal
;que llevaba el programa.

;DECFSZ F, D: Esta operación decrementa (le resta 1) a lo que haya en
;la dirección F del registro de RAM indicado y el resultado lo guarda en
;el acumulador W o en el mismo registro F, además dependiendo de si el
;resultado del decremento es cero o no, se brincará la siguiente instrucción
;o la ejecutará normalmente:
;Si el resultado del decremento NO es cero:
;Sigue la ejecución normal del código.
;-----Bucle con la copia de var1 guardada en el registro 0X67-----
;Si el resultado del decremento es cero (0X00):
;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
;en 1 al contador de programa o PC).
;HACE QUE EL PROGRAMA SALGA DE LA SUBROUTINA DE TIEMPO DE 2 VARIABLES.
DECFSZ 0X64,F ;Haz un bucle con var3
;GOTO k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por una directiva EQU que tenga el nombre de la
;parte del código a donde quiero que brinque el programa.
GOTO ST3V
;RETURN: Todas las subrutinas siempre deben terminar con la
;instrucción return, lo que hace RETURN es bajar un nivel a la pila
;y cargar en el PC el valor que esté guardado ahí, haciendo que el
;programa brinque a la parte del código después de la instrucción
;CALL que llamó esta subrutina, siguiendo así la ejecución normal
;que llevaba el programa.
;-----Bucle con var3 guardada en el registro 0X64-----
RETURN
;*****

;Los programas en ensamblador deben acabar con la directiva END.
END

```

15.- Subrutina de Tiempo de 2 Variables - Retardo de 10 ms:

15.- Realiza una subrutina de tiempo de 2 variables.

;15.- Realiza una subrutina de tiempo de 2 variables que cree un retardo de 17 us (microsegundos) a 394.247 ms (milisegundos) ya que var1 y var2 en su ecuación solo puede adoptar valores de 1 a 256.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder hacerlo debo poner la dirección de la carpeta que llega hasta donde se



```

;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:
    INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\Configuracion_PIC.asm>
    CLRF      STATUS
    ADDLW     0XEE
    ADDLW     0XEE
    ADDLW     0XEE
;*****
    MOVLW     .248
    MOVWF     0X62
    MOVLW     .67
    MOVWF     0X61
    CALL      ST2V
    MOVLW     .4
    MOVWF     0X60
    CALL      ST1V
;*****
    ADDLW     0XEE
    ADDLW     0XEE
    ADDLW     0XEE
    GOTO      $
;*****
ST1V:    NOP
        NOP
        NOP
        DECFSZ    0X60,F
        GOTO      ST1V
        RETURN
;*****
;SUBROUTINA DE TIEMPO DE DOS VARIABLES
ST2V:    MOVF     0X62,W
        MOVWF     0X63
DECRE2V  NOP
        NOP
        NOP
        DECFSZ    0X63,F
        GOTO      DECRE2V
        DECFSZ    0X61,F
        GOTO      ST2V
        RETURN
;*****
        END

```

16.- Rotación de 1 Bit por un Puerto - Retardo de 400 ms:

16.- Rota un 1 por el puerto C con espacios de tiempo de 400 milisegundos.

;16.- Rota un 1 por el puerto C con espacios de tiempo de 400 milisegundos.

```

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder
;hacerlo debo poner la dirección de la carpeta que llega hasta donde se
;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:

```

```

    INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\Configuracion_PIC.asm>
    CLRF      TRISA
    CLRF      STATUS      ;B0
    BSF       PORTA,0
ROTA:    CALL    T400MS
        RLF      PORTA,F
        BTFSC    STATUS,C
        RLF      PORTA,F
        GOTO      ROTA
T400MS:  MOVLW     .199
        MOVWF     0X64
        MOVLW     .5

```



```

MOVWF    0X65
MOVLW    .59
MOVWF    0X66
CALL     ST3V
RETURN
INCLUDE   <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\SubRutinaTiempo.asm>
END

```

17.- Pulso PWM con Frecuencia de 1kHz y Tiempo Activo del 5%:

17.- Crea un pulso PWM que tenga una frecuencia de 1 kHz que tenga un tiempo activo del 5%.

```

;17.- PWM con frecuencia de 1kHz con ciclo de trabajo de 5%.
;El periodo de 1 milisegundo es de la frecuencia de 1kHz, el 5 por ciento de
;esto es de 50 microsegundos.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder
;hacerlo debo poner la dirección de la carpeta que llega hasta donde se
;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:
INCLUDE   <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\Configuracion_PIC.asm>
CLRF     TRISA
CLRF     STATUS          ;B0
CICLO:   BSF      PORTA,0
MOVLW    .2
MOVWF    0X64
MOVLW    .2
MOVWF    0X65
MOVLW    .1
MOVWF    0X66
CALL     ST3V            ;T50US;49US;
BCF      PORTA,0
MOVLW    .157
MOVWF    0X60
CALL     ST1V            ;T950US;947
GOTO     CICLO
;Banco 1. PA, PB Y PE = DIGITALES.
INCLUDE   <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\SubRutinaTiempo.asm>
END

```

18.- Pulso PWM con Frecuencia de 1kHz y Tiempo Activo del 40%:

18.- Crea un pulso PWM que tenga una frecuencia de 1 kHz que tenga un tiempo activo del 40%.

```

;18.- PWM con frecuencia de 1kHz con ciclo de trabajo de 40%.
;El periodo de 1 milisegundo es de la frecuencia de 1kHz, el 5 por ciento de
;esto es de 50 microsegundos.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder
;hacerlo debo poner la dirección de la carpeta que llega hasta donde se
;encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con
;todo y el nombre del archivo que es Configuracion_PIC.asm de la siguiente
;manera:
INCLUDE   <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\Configuracion_PIC.asm>
CLRF     TRISA
CLRF     STATUS          ;B0
CICLO:   BSF      PORTA,0
MOVLW    .4
MOVWF    0X62
MOVLW    .14
MOVWF    0X61
CALL     ST2V            ;T400US;399US;
BCF      PORTA,0

```



```

        MOVLW    .1
        MOVWF    0X62
        MOVLW    .59
        MOVWF    0X61
        CALL     ST2V            ;T600US;597US
        GOTO     CICLO
;Banco 1. PA, PB Y PE = DIGITALES.
INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\SubRutinaTiempo.asm>
END

```

19.- Contador Hexadecimal de 0 a F con Display de 7 Segmentos:

19.- Crea un contador hexadecimal que cuente de 0 a F y muestre el resultado en un display de 7 segmentos.

;19.- Crea un contador hexadecimal que cuente de 0 a F y muestre el resultado en un display de 7 segmentos.

;La configuración del PIC será jalada por una instrucción INCLUDE, para poder hacerlo debo poner la dirección de la carpeta que llega hasta donde se encuentra el archivo dentro de este proyecto de MPLABX en mi computadora, con todo y el nombre del archivo que es Configuración_PIC.asm de la siguiente manera:

```

INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB
(Ensamblador)\1.- Ejercicios PIC16F887\Configuración_PIC.asm>
GOTO INICIO

```

;En el display de 7 segmentos hay 8 leds que se pueden encender para obtener un símbolo, estos son: DP, G, F, E, D, C, B y A.
;El orden en el que se colocó arriba es como se declarará en la tabla de abajo, se puede colocar en forma hexadecimal o en forma binaria, como el display se quiere que sea de ánodo común los leds se encenderían con 0 lógico pero como en la placa son de cátodo común se encienden con 1 lógico.

```

        ADDWF    F,D    (C, DC, Z): Suma el contenido del acumulador W con el
        ;contenido de un registro de RAM indicado por la dirección F y el
        ;resultado lo guarda en el acumulador W o en el mismo registro F.
;EL REGISTRO PCL ES EL CONTADOR DEL PROGRAMA Y SE USA PARA ACCEDER A UNA
;FILAS ESPECÍFICA DE LA TABLA.

```

```

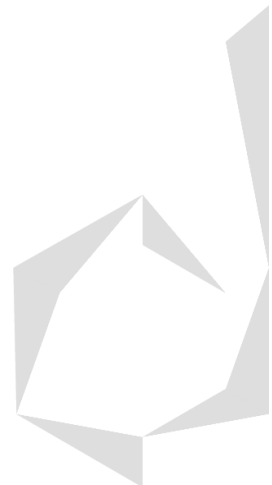
SIETESEG:    ADDWF    PCL,F ;PCL = PCL + W
;El símbolo se pondrá en forma binaria de la sig manera:
;B'DP + G + F + E + D + C + B + A'
;Y alado se pondrá en su forma hexadecimal
;RETLW K: Siempre se debe declarar dentro de una subrutina y lo que
;hace es cargar en el acumulador W el valor literal K declarado y
;luego saca al programa de la subrutina donde se encuentre.
;RETLW = MOVLW + RETURN. Por lo que tarda 2 ciclos de máquina en
;ejecutarse.

```

```

;0 en el display de ánodo común = 1100 0000 = 0XC0
;0 en el display de cátodo común = 0011 1111 = 0X3F
RETLW    B'11000000' ;W = 0X3F = 0 disp7seg
;1 en el display de ánodo común = 1111 1001 = 0XF9
;1 en el display de cátodo común = 0000 0110 = 0X06
RETLW    B'11111001' ;W = 0X06 = 1 disp7seg
;2 en el display de ánodo común = 1010 0100 = 0XA4
;2 en el display de cátodo común = 0101 1011 = 0X5B
RETLW    B'10100100' ;W = 0X5B = 2 disp7seg
;3 en el display de ánodo común = 1011 0000 = 0XB0
;3 en el display de cátodo común = 0100 1111 = 0X4F
RETLW    B'10110000' ;W = 0X4F = 3 disp7seg
;4 en el display de ánodo común = 1001 1001 = 0X99
;4 en el display de cátodo común = 0110 0110 = 0X66
RETLW    B'10011001' ;W = 0X66 = 4 disp7seg
;5 en el display de ánodo común = 1001 0010 = 0X92
;5 en el display de cátodo común = 0110 1101 = 0X6D
RETLW    B'10010010' ;W = 0X6D = 5 disp7seg
;6 en el display de ánodo común = 1000 0010 = 0X82
;6 en el display de cátodo común = 0111 1101 = 0X7D
RETLW    B'10000010' ;W = 0X7D = 6 disp7seg

```



```

;7 en el display de ánodo común = 1111 1000 = 0XF8
;7 en el display de cátodo común = 0000 0111 = 0X07
RETLW      B'11111000' ;W = 0X07 = 7 disp7seg
;8 en el display de ánodo común = 1000 0000 = 0X80
;8 en el display de cátodo común = 0111 1111 = 0X7F
RETLW      B'10000000' ;W = 0X7F = 8 disp7seg
;9 en el display de ánodo común = 1001 1000 = 0X98
;9 en el display de cátodo común = 0110 0111 = 0X67
RETLW      B'10011000' ;W = 0X67 = 9 disp7seg
;A en el display de ánodo común = 1000 1000 = 0X88
RETLW      B'10001000' ;W = 0X88 = A disp7seg
;b en el display de ánodo común = 1000 0011 = 0X83
RETLW      B'10000011' ;W = 0X83 = b disp7seg
;C en el display de ánodo común = 1100 0110 = 0XC6
RETLW      B'11000110' ;W = 0XC6 = C disp7seg
;d en el display de ánodo común = 1010 0001 = 0XA1
RETLW      B'10100001' ;W = 0XA1 = d disp7seg
;E en el display de ánodo común = 1000 0110 = 0X86
RETLW      B'10000110' ;W = 0X86 = E disp7seg
;F en el display de ánodo común = 1000 0110 = 0X8E
RETLW      B'10000110' ;W = 0X8E = F disp7seg

INICIO:      ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
              ;indicado. Siempre levanta la bandera ceros, Z = 1.
              CLRF      TRISD
              ;Hace todos los pines del puerto D sean salidas.
              CLRF      STATUS
              ;Con esto ya estoy en el banco 0
              ;Banco 0: RP1 = 0, RP0 = 0
              ;Banco 1: RP1 = 0, RP0 = 1
              ;Banco 2: RP1 = 1, RP0 = 0
              ;Banco 3: RP1 = 1, RP0 = 1
              ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
              ;indicado, como en este caso se usa un registro de propósito
              ;específico (los puertos) en vez de poner un número hexadecimal
              ;para indicar el registro F, pondré su directiva EQU (osea su
              ;nombre). Esta instrucción siempre levanta la bandera ceros, Z = 1.
              CLRF      0X20 ; 0X20 = B'00000000' = 0X00
              ;Limpia el registro general 20 de la RAM

              ;MOVWF  F, D: Lee el contenido de un registro de la RAM indicado
              ;por la dirección F y lo coloca en el mismo registro F o en el
              ;acumulador W, dependiendo de la directiva EQU que ponga en donde
              ;dice D.
SACADATO:    MOVWF      0X20,W ;W = 0X20 = B'00000000'
              ;CALL   k: Sirve para llamar una subrutina por su nombre y ejecutar
              ;una acción que se pueda repetir varias dentro del mismo código, de
              ;esta manera esa acción no la debo escribir varias veces.
              CALL      SIETESEG
              ;MOVWF  F: Lee el contenido del acumulador W y lo coloca en un
              ;registro de la RAM indicado por la dirección F.
              MOVWF      PORTD
              ;LLAMA LA SUBROUTINA DE TIEMPO QUE TARDA 1 SEGUNDO
              CALL      T1S
              ;INCF   F, D (Z): Le suma 1 (incrementa) a lo que haya almacenado
              ;en la dirección F del registro de RAM indicado y el resultado lo
              ;coloca en el mismo registro F o en el acumulador W, dependiendo de
              ;la directiva EQU que ponga en donde dice D.
              ;Solo afecta a la bandera Z = ceros.
              ;Si se incrementa en 1 al valor máximo que puede adoptar un número
              ;binario, lo empujará a volverse cero y se repetirá el conteo.
              INCF      0X20,F ;0X20 = 0X20 + 1

              ;AHORA SE COLOCARÁ UNA MÁSCARA AND CON EL VALOR 0F PARA QUE SE PUEDA HACER UN
              ;CONTEO SOLAMENTE CON LOS SEGUNDOS BITS DEL NÚMERO BINARIO DE 8 BITS EN EL
              ;REGISTRO, YA QUE ESTOS SON LOS QUE DARÁN LOS DIGITOS DE 0 A F.
              ;MOVLW  K: Coloca directamente en el acumulador W un número binario
              ;cualquiera de 8 bits dado por el valor literal K indicado en
              ;hexadecimal, poniendo 0Xnúmero_hexadecimal.
              MOVLW      0X0F ;W = 0F = B'0000 1111'
              ;ANDWF  F, D (Z): Realiza la función lógica AND entre lo que haya

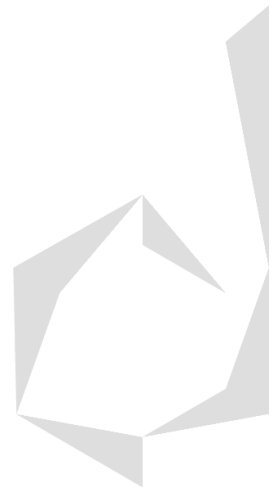
```




```

ADDWF      0X23,F
MOVF       0X23,W
XORLW      0XEE
BTFSC      STATUS,Z
GOTO       R1C1
MOVF       0X23,W
XORLW      0XDE
BTFSC      STATUS,Z
GOTO       R1C2
MOVF       0X23,W
XORLW      0XBE
BTFSC      STATUS,Z
GOTO       R1C3
MOVF       0X23,W
XORLW      0X7E
BTFSC      STATUS,Z
GOTO       R1C4
MOVF       0X23,W
XORLW      0XED
BTFSC      STATUS,Z
GOTO       R2C1
MOVF       0X23,W
XORLW      0XDD
BTFSC      STATUS,Z
GOTO       R2C2
MOVF       0X23,W
XORLW      0XBD
BTFSC      STATUS,Z
GOTO       R2C3
MOVF       0X23,W
XORLW      0X7D
BTFSC      STATUS,Z
GOTO       R2C4
MOVF       0X23,W
XORLW      0XEB
BTFSC      STATUS,Z
GOTO       R3C1
MOVF       0X23,W
XORLW      0XDB
BTFSC      STATUS,Z
GOTO       R3C2
MOVF       0X23,W
XORLW      0XBB
BTFSC      STATUS,Z
GOTO       R3C3
MOVF       0X23,W
XORLW      0X7B
BTFSC      STATUS,Z
GOTO       R3C4
MOVF       0X23,W
XORLW      0XE7
BTFSC      STATUS,Z
GOTO       R4C1
MOVF       0X23,W
XORLW      0XD7
BTFSC      STATUS,Z
GOTO       R4C2
MOVF       0X23,W
XORLW      0XB7
BTFSC      STATUS,Z
GOTO       R4C3
MOVF       0X23,W
XORLW      0X77
BTFSC      STATUS,Z
GOTO       R4C4
GOTO       REGRE
;Introducir valores en la tabla a través de sus renglones y columnas.
R1C1:      MOVLW      0X00
           GOTO       SACAV
R2C1:      MOVLW      0X01
           GOTO       SACAV

```



23.- Contador de 0 a 99 en dos displays de 7 segmentos de ánodo común con una frecuencia de 1 Hz a través de interrupciones y una tabla que contiene los tiempos de las subrutinas.

[illegible]


```

        GOTO          SIETE0
        BSF           PORTA,7
        BTFSS         0X25,6
        GOTO          SEIS0
        BSF           PORTA,6
        GOTO          INIADC
SIETE0:   BCF          PORTA,7
        GOTO          INIADC
SEIS0:    BCF          PORTA,6
INIADC:   BCF          PIR1,ADIF
        BSF          ADCON0,1      ;INICIA LA CAD
        INCLUDE      <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\RecuperaInterrupcion.asm>
;0000000000000000000000000000000000000000000000000000000
T25MS:    MOVLW       .88
        MOVWF        0X61
        MOVLW       .35
        MOVWF        0X62
        CALL         ST2V
        RETURN
;0000000000000000000000000000000000000000000000000000000
        INCLUDE      <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\SubRutinaTiempo.asm>
        END

```

25.- Cronómetro de 6 Displays con Control de Dirección, Reset y Pausa:

25.-Cronómetro en seis displays de 7 segmentos y un controlador que tiene las siguientes funciones:

Se cuenta con 6 displays de 7 segmentos, el tiempo que debe tardar cada valor desplegándose es de $1/100 \text{ s} = 10 \text{ milisegundos} = 10,000 \text{ us}$. Existen 3 push buttons, cada uno con su respectivo antirrebote.

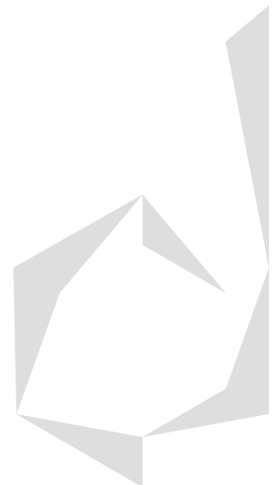
- **Push button A:** Reset.
- **Push button B:** Dirección de conteo.
 - Si está presionado el conteo será descendente.
 - Si no está presionado se hace un conteo ascendente.
- **Push button C:** Se acciona cada vez que el programa termina de desplegar las 6 cifras.
 - Si está presionado el contador se pondrá en pausa y se quedará desplegado el valor que se quedó antes de oprimir el botón, saldrá de este estado de pausa cuando se presione el botón de nuevo.
 - Si no está presionado pregunta si se ha presionado el botón 2 para hacer el conteo ascendente o descendente.

```

;24.-Cronómetro en seis displays de 7 segmentos y un controlador que tiene las
;siguientes funciones:
;Se cuenta con 6 displays de 7 segmentos, el tiempo que debe tardar cada valor
;desplegándose es de 1/100 s = 10 milisegundos = 10,000 us.
;Existen 3 push buttons, cada uno con su respectivo antirrebote.
;Push button A: Reset.
;Push button B:
;Si está presionado el conteo será descendente.
;Si no está presionado se hace un conteo ascendente.
;Push button C: Se acciona cada vez que el programa termina de desplegar
;las 6 cifras.
;Si está presionado el contador se pondrá en pausa y se quedará
;desplegado el valor que se quedó antes de oprimir el botón, saldrá de
;este estado de pausa cuando se presione el botón de nuevo.

;Si no está presionado pregunta si se ha presionado el botón 2 para
;hacer el conteo ascendente o descendente.
UNIS      EQU          0X20 ;UNIDAD ANTES DEL PUNTO
DECI      EQU          0X21 ;DECIMA ANTES DEL PUNTO

```




```

UNID      EQU      0X22 ;UNIDAD DESPUES DEL PUNTO
DECE      EQU      0X23 ;DECENA DESPUES DEL PUNTO
CENT      EQU      0X24 ;CENTENA DESPUES DEL PUNTO
UMIL      EQU      0X25 ;MILLARES DESPUES DEL PUNTO
CONT60    EQU      0X26 ;CONTADOR

        INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB
(Ensamblador)\1.- Ejercicios PIC16F887\Configuracion_PIC.asm>
;Lo que hace la configuración es indicar el PIC que estoy usando, declarar las
;2 palabras de configuración, usar otra directiva INCLUDE para jalar el archivo
;P16F887.INC que incluye las 35 instrucciones del PIC junto con las directivas
;EQU (los nombres) de sus registros, la directiva ORG que indica la dirección de
;la memoria FLASH desde donde se empezará a guardar el código, el limpiado
;(poner en 0) de todos los bits de los puertos (A, B, C, D y E) y hacer que los
;puertos A, B y E (que son los únicos que pueden ser analógicos o digitales)
;sean todos entradas o salidas digitales.
        GOTO      INICIO ;SALTA A EJECUTAR EL PRROGRAMA PRINCIPAL
        INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB
(Ensamblador)\1.- Ejercicios PIC16F887\TablaDisplay7Segmentos.asm>
;Tabla que enciende los leds DP, G, F, E, D, C, B y A de los displays de 7 segmentos.
        INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB
(Ensamblador)\1.- Ejercicios PIC16F887\SubRutinaTiempo.asm>
;Subrutinas de tiempo de 1, 2 y 3 variables.

INICIO:   CLRF      TRISD ;El puerto D enciende los leds de los displays
          CLRF      TRISA ;El puerto A enciende cada display
          ;Hace que todos los pines del puerto A y D sean salidas.
          CLRF      STATUS ;SE CAMBIA AL BANCO 0.
          CALL      REBOTE
          BTFSC     PORTB,0
          GOTO      INVERSA
          CALL      REBOTE

;Limpieza de los registros de propósito general 0X20 a 0X25 que guardarán el
;valor binario que encenderá cada led del display para mostrar un dígito.
LIMPIATODO: CLRF      UMIL
LIMPIACENT: CLRF      CENT
LIMPIADECE: CLRF      DECE
LIMPIAUNID: CLRF      UNID
LIMPIADECI: CLRF      DECI
LIMPIAUNIS: CLRF      UNIS

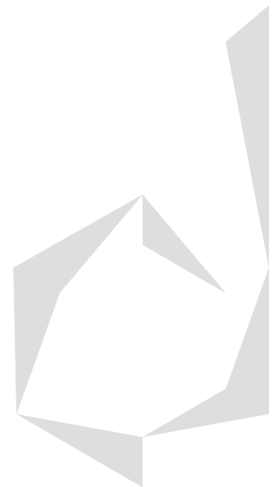
RECARGA60: BTFSC     PORTB,0
          GOTO      INVERSA
          ;Esto sirve para poner pausa si es que ha sido presionado el
          ;boton 2.
          MOVLW     .1
          MOVWF     CONT60

SACADATO: MOVF      UNIS,W
          CALL      SIETESEG
          MOVWF     PORTD
          BSF       PORTA,0 ;.UNIDADES DECIMALES
          CALL      T1600U
          CALL      T45U
          NOP
          BCF       PORTA,0

          MOVF      DECI,W
          CALL      SIETESEG
          MOVWF     PORTD
          BSF       PORTA,1 ;.DECENAS DECIMALES
          CALL      T1600U
          CALL      T45U
          NOP
          BCF       PORTA,1

          MOVF      UNID,W
          CALL      SIETESEGP
          MOVWF     PORTD

```



```

BSF          PORTA,2      ;UNIDADES
CALL         T1600U
CALL         T45U
NOP
BCF          PORTA,2

MOVF         DECE,W
CALL         SIETESEG
MOVWF        PORTD
BSF          PORTA,3      ;DECENAS
CALL         T1600U
CALL         T45U
BCF          PORTA,3

MOVF         CENT,W
CALL         SIETESEG
MOVWF        PORTD
BSF          PORTA,4      ;CENTENAS
CALL         T1600U
CALL         T45U
BCF          PORTA,4

MOVF         UMIL,W
CALL         SIETESEG
MOVWF        PORTD
BSF          PORTA,5      ;MILLARES
CALL         T1600U
CALL         T45U
BCF          PORTA,5

BTFSC        PORTB,1
CALL         PAUSA1
BTFSC        PORTB,0
GOTO         INVERSA
DECFSZ       CONT60, F
GOTO         SACADATO

CONTADOR_UNIS: INCF          UNIS,F
                MOVLW        .10
                XORWF        UNIS,W
                ;MÁSCARA XOR USADA PARA LIMITAR EL CONTEO HASTA 10 DECIMAL
                ;Si se levanta la bandera ceros al usarse una máscara XOR es
                ;porque los dos números comparados son iguales.
                BTFSS        STATUS,Z
                GOTO         RECARGA60

CONTADOR_DECI: INCF          DECI,F
                MOVLW        .10
                XORWF        DECI,W
                BTFSS        STATUS,Z
                GOTO         LIMPIAUNIS

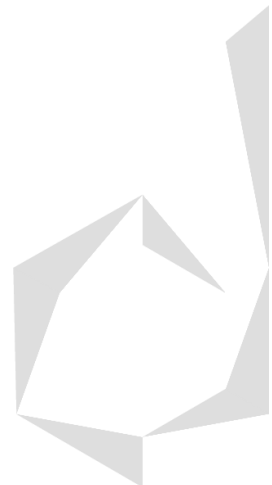
CONTADOR_UNID: INCF          UNID,F
                MOVLW        .10
                XORWF        UNID,W
                BTFSS        STATUS,Z
                GOTO         LIMPIADECI

CONTADOR_DECE: INCF          DECE,F
                MOVLW        .10
                XORWF        DECE,W
                BTFSS        STATUS,Z
                GOTO         LIMPIAUNID

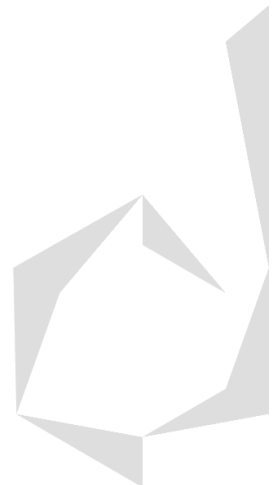
CONTADOR_CENT: INCF          CENT,F
                MOVLW        .10
                XORWF        CENT,W
                BTFSS        STATUS,Z
                GOTO         LIMPIADECE

CONTADOR_UMIL: INCF          UMIL,F

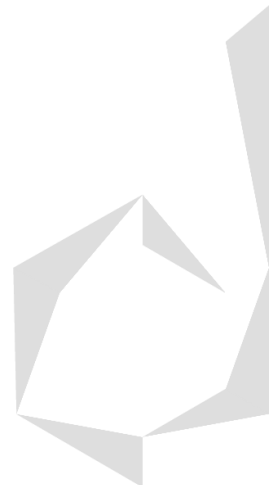
```



	MOVLW	.10	
	XORWF	UMIL,W	
	BTFS	STATUS,Z	
	GOTO	LIMPIACENT	
	GOTO	LIMPIATODO	
INVERSA:	CALL	RECARGA	
	BTFS	PORTB,0	
	GOTO	\$-1	
	CALL	REBOTE	
LIMPIATODOI:	MOVLW	.10	
	MOVWF	UMIL	
LIMPIACENTI:	MOVLW	.10	
	MOVWF	CENT	
LIMPIADECEI:	MOVLW	.10	
	MOVWF	DECE	
LIMPIAUNIDI:	MOVLW	.10	
	MOVWF	UNID	
LIMPIADECII:	MOVLW	.10	
	MOVWF	DECI	
LIMPIAUNISI:	MOVLW	.10	
	MOVWF	UNIS	
RECARGA60I:	BTFS	PORTB,0	
	GOTO	INVERSA	
		;Esto sirve para poner pausa si es que ha sido presionado el	
		;boton 2.	
	MOVLW	.1	
	MOVWF	CONT60	
SACADATOI:	MOVF	UNIS,W	
	CALL	SIETESEGI	
	MOVWF	PORTD	
	BSF	PORTA,0	; .UNIDADES DECIMALES
	CALL	T1600U	
	CALL	T45U	
	NOP		
	BCF	PORTA,0	
	MOVF	DECI,W	
	CALL	SIETESEGI	
	MOVWF	PORTD	
	BSF	PORTA,1	; .DECENAS DECIMALES
	CALL	T1600U	
	CALL	T45U	
	NOP		
	BCF	PORTA,1	
	MOVF	UNID,W	
	CALL	SIETESEGPI	
	MOVWF	PORTD	
	BSF	PORTA,2	; UNIDADES
	CALL	T1600U	
	CALL	T45U	
	NOP		
	BCF	PORTA,2	
	MOVF	DECE,W	
	CALL	SIETESEGI	
	MOVWF	PORTD	
	BSF	PORTA,3	; DECENAS
	CALL	T1600U	
	CALL	T45U	
	BCF	PORTA,3	
	MOVF	CENT,W	
	CALL	SIETESEGI	
	MOVWF	PORTD	
	BSF	PORTA,4	; CENTENAS
	CALL	T1600U	



	CALL	T45U	
	BCF	PORTA,4	
	MOVF	UMIL,W	
	CALL	SIETESEGI	
	MOVWF	PORTD	
	BSF	PORTA,5	;MILLARES
	CALL	T1600U	
	CALL	T45U	
	BCF	PORTA,5	
	BTFSC	PORTB,1	
	CALL	PAUSA1	
	BTFSC	PORTB,0	
	GOTO	RECARGA60I	
	DECFSZ	CONT60, F	
	GOTO	SACADATO	
CONT_INV_UNIS:	DECF	UNIS,F	
	MOVLW	.0	
	XORWF	UNIS,W	
	BTFSS	STATUS,Z	
	GOTO	RECARGA60I	
CONT_INV_DECI:	DECF	DECI,F	
	MOVLW	.0	
	XORWF	DECI,W	
	BTFSS	STATUS,Z	
	GOTO	LIMPIAUNISI	
CONT_INV_UNID:	DECF	UNID,F	
	MOVLW	.0	
	XORWF	UNID,W	
	BTFSS	STATUS,Z	
	GOTO	LIMPIADECII	
CONT_INV_DECE:	DECF	DECE,F	
	MOVLW	.0	
	XORWF	DECE,W	
	BTFSS	STATUS,Z	
	GOTO	LIMPIAUNIDI	
CONT_INV_CENT:	DECF	CENT,F	
	MOVLW	.0	
	XORWF	CENT,W	
	BTFSS	STATUS,Z	
	GOTO	LIMPIADECEI	
CONT_INV_UMIL:	DECF	UMIL,F	
	MOVLW	.0	
	XORWF	UMIL,W	
	BTFSS	STATUS,Z	
	GOTO	LIMPIACENTI	
	GOTO	LIMPIATODOI	
PAUSA1:	BTFSC	PORTB,1	
	GOTO	\$-1	
	CALL	REBOTE	
SACADATOP:	MOVF	UNIS,W	
	CALL	SIETESEG	
	MOVWF	PORTD	
	BSF	PORTA,0	; .UNIDADES
	CALL	T1600U	
	CALL	T45U	
	NOP		
	BCF	PORTA,0	
	MOVF	DECI,W	
	CALL	SIETESEG	
	MOVWF	PORTD	




```

REBOTE:      MOVLW      .176
             MOVWF      0X61
             MOVLW      .23
             MOVWF      0X62
             CALL       ST2V
             NOP
             RETURN

RECARGA:     BTFSC      PORTB,0
             GOTO       $-1
             ;Esto sirve para poner pausa si es que ha sido presionado el
             ;boton 2.
             MOVLW      .1
             MOVWF      CONT60
             RETURN
             END

```

26.- Control de Pantalla LCD:

26.-Control de pantalla LCD.

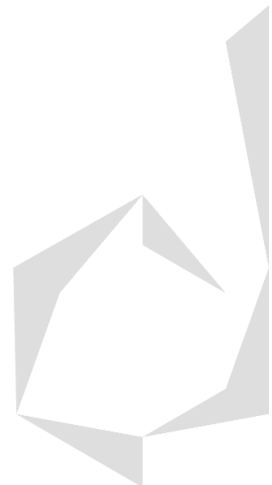
```

;26.- Control de Pantalla LCD.
PROCESSOR    16F887

;Se deben declarar 2 palabras de configuración de 14 bits para setear el PIC
;usando los registros 2007 y 2008 hexadecimales (que no pertenecen a la RAM).
__CONFIG 0X2007, 0X23E4
;La primera palabra de configuración que se guarda en el registro 0X2007 y sus
;bits significan:
;1:  Apaga el modo DEBUG que revisa el código línea por línea (bit 13).
;0:  Apaga el Modo de Baja Tensión y habilita el Puerto B como entradas y
;salidas digitales o analógicas (bit 12).
;1:  Activa el modo Reloj a Prueba de Fallas que monitorea si el oscilador
;funciona bien (bit 11).
;0:  Desactiva el divisor de reloj, dejando la frecuencia default del
;oscilador interno en el PIC que es de 4MHz (bit 10).
;1:  Activa el Brown-Out todo el tiempo, que reiniciará al PIC si el valor de
;voltaje en el oscilador baja de cierto rango (bits 9 y 8).
;1:  Apaga el modo de protección de escritura en la memoria RAM (bit 7).
;1:  Apaga el modo de protección de escritura en la memoria FLASH (bit 6).
;1:  Hace que el pin RE3 del puerto E funcione como reset, reiniciando el PIC
;cuando le ingrese un 1 lógico de una señal digital (bit 5).
;0:  Enciende el Power-up Timer, hace que el PIC tarde 75 milisegundos en
;encenderse para proteger al microcontrolador de las variaciones que vienen de
;la fuente de alimentación (bit 4).
;0:  Apaga el Watchdog (bit 3).
;100: Elige el oscilador tipo INTOSCIO, que usa el oscilador interno incluido
;en el PIC de 4 MHz y configura los pines RA6 y RA7 para que ambos sean
;entradas/salidas analógicas o digitales (bits 2, 1 y 0).
;Por lo tanto la palabra de configuración es 10 1011 1110 0100 = 2BE4
__CONFIG 0X2008, 0X3FFF
;La segunda palabra de configuración que se guarda en el registro 0X2008 y sus
;bits significan:
;111: Siempre estarán de esta manera, no se les debe cambiar (bits 13,
;12 y 11).
;11:  Apaga el modo de protección de escritura en la memoria FLASH (bits
;10 y 9).
;1:  Hace que el Brown-Out reinicie el PIC cuando la señal de reloj baje
;de 4V (bit 8).
;1111 1111: Siempre estarán de esta manera, no se les debe cambiar (bits 7, 6,
;5, 4, 3, 2, 1 y 0).
;Por lo tanto la palabra de configuración es 11 1111 1111 1111 = 3FFF

;La directiva INCLUDE sirve para abrir un archivo de texto plano, copiar todo
;su contenido y pegarlo en el programa, en este caso se usa para añadir el
;archivo P16F887.INC que incluye las 35 instrucciones del PIC16F887 a mi
;programa junto con sus directivas EQU para que las pueda usar.
INCLUDE <P16F887.INC>
ORG      0X0000
W_R      EQU      0X70
ST_R      EQU      0X71
PC_R      EQU      0X72
CENTI      EQU      0X25
DECI      EQU      0X20
UNID      EQU      0X21
DECE      EQU      0X22
CENT      EQU      0X23
MILL      EQU      0X24
TOR      EQU      0X26
V_ALTO      EQU      0X27
V_BAJA      EQU      0X28
GOTO      INICIO
NOP
NOP

```

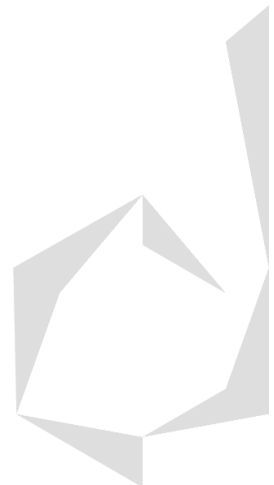


```

NOP
INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\RecuperaInterrupcion.asm>
GOTO RSI
INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios
PIC16F887\TablaDisplay7Segmentos.asm>
;Tabla que enciende los leds DP, G, F, E, D, C, B y A de los displays de 7 segmentos.
INCLUDE <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.- Ejercicios PIC16F887\SubRutinaTiempo.asm>
;Subrutinas de tiempo de 1, 2 y 3 variables.

INICIO: CLRf PORTD
CLRf PORTC
CLRf PORTA
BSF STATUS,RP0
BSF STATUS,RP1
;BANCO 3
MOVLW B'11111111' ;PE -> ANALOGICO
MOVWF ANSEL ;PA = AN, PA3 = AN = VREF+;
CLRf ANSELH ;PB DIGITAL.
;BANCO 1
BCF STATUS,RP1 ;B1.
CLRf TRISD
MOVLW B'00110110'
MOVWF PORTC
;CONFIGURACION PARA INTERRUPCIONES POR TECLADO MATRICIAL:
MOVLW 0XFF
MOVWF IOCB ;TODOS LOS PINES DEL PORTB SON SENSIBLES AL CAMBIO
MOVLW 0X0F
MOVWF TRISB
BCF OPTION_REG,7 ;ACTIVACION DE LAS RESISTENCIAS DE ELEVACIÓN 14
MOVLW 0XFF
MOVWF WPUB ;TODAS LAS RESISTENCIAS
MOVLW B'00010000' ;AJUSTE_IQZ,VREF-=VSS,VREF+=AN3
MOVWF ADCON1
;BANCO 0
CLRf CENTI
CLRf DECI
CLRf UNID
CLRf DECE
CLRf CENT
CLRf MILL
BCF STATUS,RP0 ;B0
;CONFIGURACIÓN DEL PERIFÉRICO QUE INTERRUMPIR:
MOVf PORTB,W
BCf INTCON,RBIF ;BANDERA ABAJO INT ON CHANGE
BSF INTCON,RBIE ;PERMISO PARA QUE INT ON CHANGE SEA FUENTE DE INTERRUP
BSF INTCON,GIE ;PERMISO GLOBAL DE INT
;ROUTINA DE SERVICIO DE INTERRUPCION:
RSI: CLRf PORTD
CALL T25MS
BTFSS PORTB,0
GOTO REN1
BTFSS PORTB,1
GOTO REN2
BTFSS PORTB,2
GOTO REN3
BTFSS PORTB,3
GOTO REN4
BTFSS PORTB,4
GOTO REN4COL1
BTFSS PORTB,5
GOTO REN4COL2
BTFSS PORTB,6
GOTO REN4COL3
BTFSS PORTB,7
GOTO $-1
GOTO REGRESA
REN3: CALL CAMBIO_PB
BTFSS PORTB,4
GOTO REN3COL1
BTFSS PORTB,5
GOTO REN3COL2
BTFSS PORTB,6
GOTO REN3COL3
BTFSS PORTB,7
GOTO $-1
GOTO REGRESA
REN2: CALL CAMBIO_PB
BTFSS PORTB,4
GOTO REN2COL1
BTFSS PORTB,5
GOTO REN2COL2
BTFSS PORTB,6
GOTO REN2COL3
BTFSS PORTB,7
GOTO $-1
GOTO REGRESA
REN1: CALL CAMBIO_PB
BTFSS PORTB,4
GOTO REN1COL1
BTFSS PORTB,5
GOTO REN1COL2
BTFSS PORTB,6
GOTO REN1COL3
GOTO $-1
; POT 2

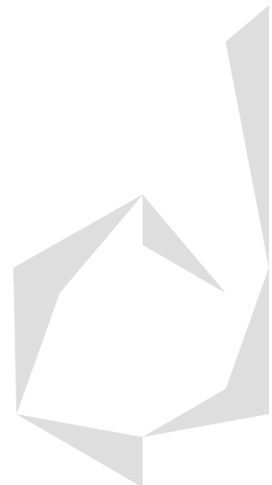
```



```

REN1COL4:CLRF    CENTI
           CLRF    DECI
           CLRF    UNID
           CLRF    DECE
           CLRF    CENT
           CLRF    MILL
           MOVLW    B'11011001'
           MOVWF    ADCON0 ;OSC=RC,CHS=AN6,DONE,ADON.
           CALL     T_ESTA
OTRA14:    BSF     ADCON0,1
           BTFSC    ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
           GOTO     $-1
           CLRF     V_ALTO
           CLRF     V_BAJO
           MOVF     ADRESH,W ; RESPALDO EL DATO
           MOVWF    V_ALTO ;
           CALL     CONVERSION
; DESPLEGANDO VALORES EN LCD
           BCF     PORTC,0 ; MODO INSTRUCCION
           MOVLW    0X01 ; LIMPIA DISPLAY
           MOVWF    PORTD
           CALL     COMANDO
           MOVLW    0X0C ; SELECCIONA LA PRIMER LINEA
           MOVWF    PORTD
           CALL     COMANDO ; DA DE ALTA EL COMANDO
           MOVLW    0X3C ; SE CONFIGURA EL CURSOR
           MOVWF    PORTD
           CALL     COMANDO
           BSF     PORTC,0 ; MODO DATO
           CALL     POT_2
           CALL     LINEA2
           MOVF     MILL,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVF     CENT,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVLW    '.'
           MOVWF    PORTD
           CALL     ENVIAR
           MOVF     DECE,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVF     UNID,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVF     DECI,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVF     CENTI,W
           CALL     LCD
           MOVWF    PORTD
           CALL     ENVIAR
           MOVLW    'v'
           MOVWF    PORTD
           CALL     ENVIAR
           CALL     T2TAD
           BTFSS    PORTB,7
           GOTO     $-1
           CLRF     PORTD
           GOTO     REGRESA
REN4COL3: BTFSS    PORTB,6
           GOTO     $-1
           GOTO     REGRESA
REN3COL3: BTFSS    PORTB,6
           GOTO     $-1
           GOTO     REGRESA
;FUENTE 3
REN2COL3:CLRF    CENTI
           CLRF    DECI
           CLRF    UNID
           CLRF    DECE
           CLRF    CENT
           CLRF    MILL
           MOVLW    B'11011101'
           MOVWF    ADCON0 ;OSC=RC,CHS=AN4,DONE,ADON.
           CALL     T_ESTA
OTRA23:    BSF     ADCON0,1
           BTFSC    ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
           GOTO     $-1
           CLRF     V_ALTO
           CLRF     V_BAJO
           MOVF     ADRESH,W ; RESPALDO EL DATO
           MOVWF    V_ALTO
           CALL     CONVERSION
; DESPLEGANDO VALORES EN LCD
           BCF     PORTC,0 ; MODO INSTRUCCION
           MOVLW    0X01 ; LIMPIA DISPLAY
           MOVWF    PORTD
           CALL     COMANDO

```

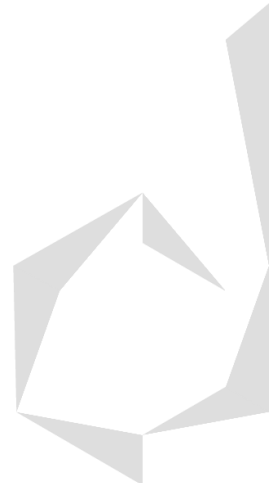



```

MOVLW    0X0C ; SELECCIONA LA PRIMER LINEA
MOVWF    PORTD
CALL     COMANDO ; DA DE ALTA EL COMANDO
MOVLW    0X3C ; SE CONFIGURA EL CURSOR
MOVWF    PORTD
CALL     COMANDO
BSF      PORTC,0 ; MODO DATO
CALL     FUENTE_3
CALL     LINEA2
MOVF     MILL,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     CENT,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVLW    '.'
MOVWF    PORTD
CALL     ENVIAR
MOVF     DECE,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     UNID,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     DECI,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     CENTI,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVLW    ' '
MOVWF    PORTD
CALL     ENVIAR
MOVLW    'v'
MOVWF    PORTD
CALL     ENVIAR
CALL     T2TAD
BTFSS    PORTB,6
GOTO     $-1
CLRF     PORTD
GOTO     REGRESA

; POT 1
REN1COL3:CLRF    CENTI
CLRF     DECI
CLRF     UNID
CLRF     DECE
CLRF     CENT
CLRF     MILL
MOVLW    B'11010101'
MOVWF    ADCON0 ; OSC=RC, CHS=AN5, DONE, ADON.
CALL     T_ESTA
BSF      ADCON0,1
BTFSC    ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
GOTO     $-1
CLRF     V_ALTO
CLRF     V_BAJO
MOVF     ADRESH,W ; RESPALDO EL DATO
MOVWF    V_ALTO
CALL     _CONVERSION
; DESPLEGANDO VALORES EN LCD
BCF      PORTC,0 ; MODO INSTRUCCION
MOVLW    0X01 ; LIMPIA DISPLAY
MOVWF    PORTD
CALL     COMANDO
MOVLW    0X0C ; SELECCIONA LA PRIMER LINEA
MOVWF    PORTD
CALL     COMANDO ; DA DE ALTA EL COMANDO
MOVLW    0X3C ; SE CONFIGURA EL CURSOR
MOVWF    PORTD
CALL     COMANDO
BSF      PORTC,0 ; MODO DATO
CALL     POT_1
CALL     LINEA2
MOVF     MILL,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     CENT,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVLW    '.'
MOVWF    PORTD
CALL     ENVIAR
MOVF     DECE,W
CALL     LCD
MOVWF    PORTD
CALL     ENVIAR
MOVF     UNID,W

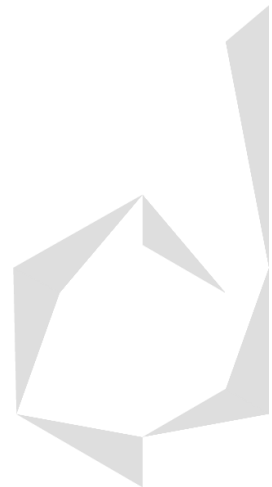
```



```

CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF DECI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENTI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'V'
MOVWF PORTD
CALL ENVIAR
CALL T2TAD
BTFS PORTB,6
GOTO $-1
CLRF PORTD
GOTO REGRESA
REN4COL2:BTFS PORTB,5
GOTO $-1
GOTO REGRESA
;PWM CON POT2
REN3COL2:MOVLW B'11011001'
MOVWF ADCON0 ;OSC=RC,CHS=AN6,DONE,ADON.
CALL T_ESTA
BSF ADCON0,1
BTFS ADCON0,1 ;YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
BCF STATUS,RP1 ;B1.
MOVLW .49 ;Fpwm = 5 KHz.
MOVWF FR2
BCF STATUS,RP0 ;B0
BSF CCP2CON,3
BSF CCP2CON,2 ;TIMER 2 COMO PWM.
MOVF ADRESH,W
MOVWF CCPR2L ;AJUSTANDO EL PERIODO DE CICLO UTIL
;MOVF ADRESL,W
BCF PIR1,TMR2IF ;BAJAMOS BANDERA
BCF T2CON,1
BCF T2CON,0 ;PRESCALER = 1
BSF T2CON,TMR2ON;HABILITA TIMER2
BTFS PIR1,TMR2IF ;ESPERAR A QUE SE BAJE LA BANDERA
GOTO $-1
BSF STATUS,RP0 ;B1
MOVLW B'11110100'
MOVWF TRISC ;PONER LOS BITS DE SALIDA RC1 Y RC2
BTFS PORTB,5
GOTO $-1
CLRF PORTD
GOTO REGRESA
; FUENTE 2
REN2COL2:CLRF CENTI
CLRF DECI
CLRF UNID
CLRF DECE
CLRF CENT
CLRF MILL
MOVLW B'11010001'
MOVWF ADCON0 ;OSC=RC,CHS=AN4,DONE,ADON.
CALL T_ESTA
BSF ADCON0,1
BTFS ADCON0,1 ;YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
CLRF V_ALTO
CLRF V_BAJO
MOVF ADRESH,W ;RESPALDO EL DATO
MOVWF V_ALTO
CALL CONVERSION
;DESPLEGANDO VALORES EN LCD
BCF PORTC,0 ;MODO INSTRUCCION
MOVLW 0X01 ;LIMPIA DISPLAY
MOVWF PORTD
CALL COMANDO
MOVLW 0X0C ;SELECCIONA LA PRIMER LINEA
MOVWF PORTD
CALL COMANDO
MOVLW 0X3C ;DA DE ALTA EL COMANDO
MOVWF PORTD ;SE CONFIGURA EL CURSOR
CALL COMANDO
BSF PORTC,0 ;MODO DATO
CALL FUENTE_2
CALL LINEA2
MOVF MILL,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENT,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '

```

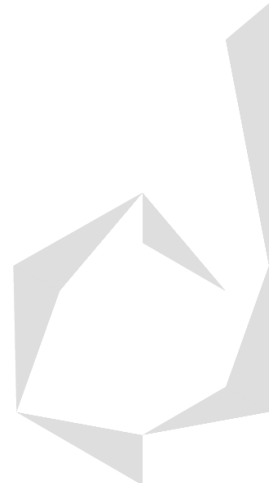


```

MOVWF PORTD
CALL ENVIAR
MOVF DECE,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF UNID,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF DECI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENTI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'v'
MOVWF PORTD
CALL ENVIAR
CALL T2TAD
BTFS PORTB,5
GOTO $-1
CLRF PORTD
GOTO REGRESA

; LM35_2
REN1COL2:CLRF CENTI
CLRF DECI
CLRF UNID
CLRF DECE
CLRF CENT
CLRF MILL
MOVLW B'11000101'
MOVWF ADCON0 ;OSC=RC,CHS=AN1,DONE,ADON.
CALL T_ESTA
OTRA12:BSF ADCON0,1
BTFS ADCON0,1 ;YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
CLRF V_ALTO
CLRF V_BAJO
MOVF ADRESH,W ;RESPALDO EL DATO
MOVWF V_ALTO
CALL CONVERSION
; DESPLEGANDO VALORES EN LCD
BCF PORTC,0 ;MODO INSTRUCCION
MOVLW 0X01 ;LIMPIA DISPLAY
MOVWF PORTD
CALL COMANDO
MOVLW 0X0C ;SELECCIONA LA PRIMER LINEA
MOVWF PORTD
CALL COMANDO ;DA DE ALTA EL COMANDO
MOVLW 0X3C ;SE CONFIGURA EL CURSOR
MOVWF PORTD
CALL COMANDO
BSF PORTC,0 ;MODO DATO
CALL LM35_2
CALL LINEA2
MOVLW 'T'
MOVWF PORTD
CALL ENVIAR
MOVLW '='
MOVWF PORTD
CALL ENVIAR
MOVF MILL,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENT,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF DECE,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF UNID,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW '.'
MOVWF PORTD
CALL ENVIAR
MOVF DECI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENTI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '

```



```

MOVWF PORTD
CALL ENVIAR
MOVLW 'C'
MOVWF PORTD
CALL ENVIAR
CALL T2TAD
BTFSS PORTB,5
GOTO $-1
CLRF PORTD
GOTO REGRESA
REN4COL1:BTFSS PORTB,4
GOTO $-1
GOTO REGRESA

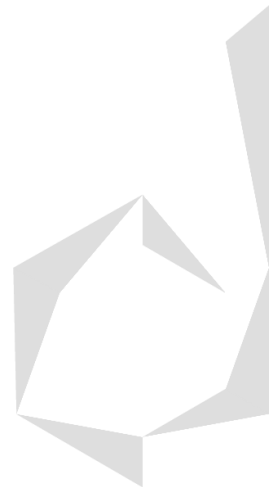
;PWM CON POT1
REN3COL1:MOVLW B'11010101'
MOVWF ADCON0 ;OSC=RC,CHS=AN5,DONE,ADON.
CALL T_ESTA
BSF ADCON0,1
BTFSC ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
BCF STATUS,RP1 ;B1.
MOVLW .49 ;Fpwm = 5 KHz.
MOVWF PR2
BCF STATUS,RP0 ;B0
BSF CCP1CON,3
BSF CCP1CON,2 ;TIMER 2 COMO PWM.
MOVF ADRESH,W
MOVWF CCP1L ;AJUSTANDO EL PERIODO DE CICLO UTIL

;MOVF ADRESL,W
BCF PIR1,TMR2IF ;BAJAMOS BANDERA
BCF T2CON,1
BCF T2CON,0 ;PRESCALER = 1
BSF T2CON,TMR2ON ;HABILITA TIMER2
BTFSS PIR1,TMR2IF ; ESPERAR A QUE SE BAJE LA BANDERA
GOTO $-1
BSF STATUS,RP0 ;B1
MOVLW B'11110010'
MOVWF TRISC ; PONER LOS BITS DE SALIDA RC1 Y RC2
BTFSS PORTB,4
GOTO $-1
GOTO REGRESA

; FUENTE 1
REN2COL1:CLRF CENTI
CLRF DECI
CLRF UNID
CLRF DECE
CLRF CENT
CLRF MILL
BCF PORTC,0 ; MODO INSTRUCCION
MOVLW 0X01 ; LIMPIA DISPLAY
BSF PORTC,0 ; MODO DATO
MOVLW B'11001001'
MOVWF ADCON0 ;OSC=RC,CHS=AN2,DONE,ADON.
CALL T_ESTA
BSF ADCON0,1 ; EMPIEZA LA CONVERSION
BTFSC ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
CLRF V_ALTO
CLRF V_BAJO
MOVF ADRESH,W ; RESPALDO EL DATO
MOVWF V_ALTO ;
CALL CONVERSION

; DESPLEGANDO VALORES EN LCD
BCF PORTC,0 ; MODO INSTRUCCION
MOVLW 0X01 ; LIMPIA DISPLAY
MOVWF PORTD
CALL COMANDO
MOVLW 0X0C ; SELECCIONA LA PRIMER LINEA
MOVWF PORTD
CALL COMANDO ; DA DE ALTA EL COMANDO
MOVLW 0X3C ; SE CONFIGURA EL CURSOR
MOVWF PORTD
CALL COMANDO
BSF PORTC,0 ; MODO DATO
CALL FUENTE_1
CALL LINEA2
MOVF MILL,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENT,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW '.'
MOVWF PORTD
CALL ENVIAR
MOVF DECE,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF UNID,W
CALL LCD
MOVWF PORTD
CALL ENVIAR

```

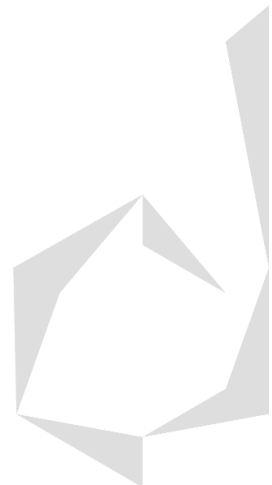


```

MOVWF DECI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVWF CENTI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'v'
MOVWF PORTD
CALL ENVIAR
CALL T2TAD
BTFSS PORTB,4
GOTO $-1
CLRF PORTD
GOTO REGRESA

; LM35_1
RENICOL1:CLRF CENTI
CLRF DECI
CLRF UNID
CLRF DECE
CLRF CENT
CLRF MILL
MOVLW B'11000001'
MOVWF ADCON0 ; OSC=RC, CHS=AN0, DONE, ADON.
CALL T_ESTA
BSF ADCON0,1
BTFSC ADCON0,1 ; YA TERMINO DE HACER LA CONVERSIÓN?
GOTO $-1
CLRF V_ALTO
CLRF V_BAJO
MOVF ADRESH,W ; RESPALDO EL DATO
MOVWF V_ALTO
CALL CONVERSION
; DESPLEGANDO VALORES EN LCD
BCF PORTC,0 ; MODO INSTRUCCION
MOVLW 0X01 ; LIMPIA DISPLAY
MOVWF PORTD
CALL COMANDO
MOVLW 0X0C ; SELECCIONA LA PRIMER LINEA
MOVWF PORTD
CALL COMANDO ; DA DE ALTA EL COMANDO
MOVLW 0X3C ; SE CONFIGURA EL CURSOR
MOVWF PORTD
CALL COMANDO
BSF PORTC,0 ; MODO DATO
CALL LM35_1
CALL LINEA2
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'T'
MOVWF PORTD
CALL ENVIAR
MOVLW '='
MOVWF PORTD
CALL ENVIAR
MOVF MILL,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENT,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF DECE,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF UNID,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW '.'
MOVWF PORTD
CALL ENVIAR
MOVF DECI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVF CENTI,W
CALL LCD
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'C'
MOVWF PORTD
CALL ENVIAR
CALL T2TAD
BTFSS PORTB,4

```



```

        GOTO    $-1
        CLRF    PORTD
REGRESA:CALL    T2SMS
        BSF     STATUS,RP0
        ;BANCO 1
        MOVLW   0X0F
        MOVWF   TRISB
        BCF     OPTION_REG,7
        MOVLW   0XFF
        MOVWF   WPUB
        BCF     STATUS,RP0 ;B0
        CLRF    PORTB
        MOVF    PORTB,W
        BCF     INTCON,RBIF
CAMBIO_PB:BSF   STATUS,RP0 ;B1
        MOVLW   0XF0
        MOVWF   TRISB
        BCF     OPTION_REG,7
        MOVLW   0XFF
        MOVWF   WPUB
        BCF     STATUS,RP0 ;B0
        CLRF    PORTB
        RETURN

```

```

; ----- SUBROUTINAS -----

```

```

CONVERSION:
CONT_MILL:MOVLW .1000
        SUBWF   V_ALTO,F
        BTFSS   STATUS, C
        GOTO    CONTINUA_CENT
        INCF    MILL,F
        GOTO    CONT_MILL
CONTINUA_CENT:
        MOVLW   .1000
        ADDWF   V_ALTO
CONT_CENT:
        MOVLW   .100
        SUBWF   V_ALTO,F
        BTFSS   STATUS, C
        GOTO    CONTINUA_DECE
        INCF    CENT,F
        GOTO    CONT_CENT
CONTINUA_DECE:
        MOVLW   .100
        ADDWF   V_ALTO,F
CONT_DEC:
        MOVLW   .10
        SUBWF   V_ALTO, F
        BTFSS   STATUS, C
        GOTO    CONTINUA_UNID
        INCF    DECE,F
        GOTO    CONT_DEC
CONTINUA_UNID:
        MOVLW   .10
        ADDWF   V_ALTO,F
        MOVF    V_ALTO,W
        MOVWF   UNID
        CLRF    CENTI
        CLRF    DECI
        BSF     STATUS,RP0 ;B1
        MOVF    ADRESL,W
        BCF     STATUS,RP0 ;B0
        MOVWF   V_BAJO
        BTFSC   V_BAJO,7
        BSF     DECI,F
        BTFSC   V_BAJO,6
        BSF     CENTI,F
        RETURN

```

```

; LM35 -> 1

```

```

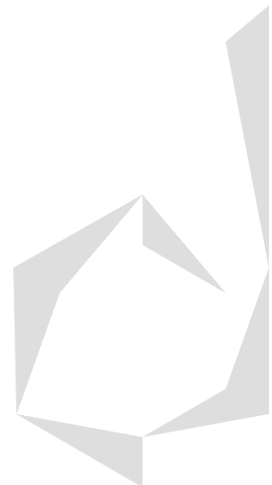
LM35_1:
        MOVLW   ' '
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   ' '
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   'L'
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   'M'
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   '3'
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   '5'
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   ' '
        MOVWF   PORTD
        CALL    ENVIAR
        MOVLW   '1'
        MOVWF   PORTD
        CALL    ENVIAR
        RETURN

```

```

; LM35 -> 2

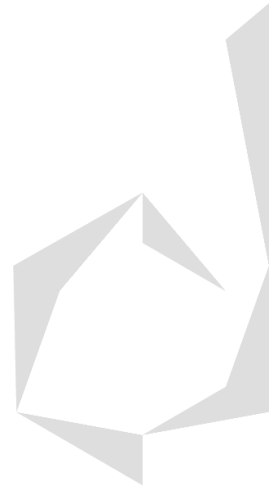
```



```

LM35_2:
    MOVLW    ' '
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    ' '
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'L'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'M'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    '3'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    '5'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    ' '
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    '2'
    MOVWF    PORTD
    CALL     ENVIAR
    RETURN
; POT -> 1
POT_1:
    MOVLW    'P'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'O'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'T'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'E'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'N'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'C'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'I'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'O'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'M'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'E'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'T'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'R'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'O'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    ' '
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    '1'
    MOVWF    PORTD
    CALL     ENVIAR
    RETURN
; POT -> 2
POT_2:
    MOVLW    'P'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'O'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'T'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'E'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'N'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'C'
    MOVWF    PORTD
    CALL     ENVIAR
    MOVLW    'I'

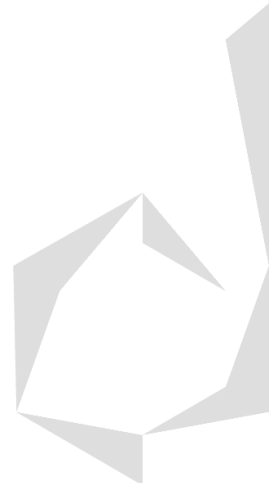
```



```

MOVWF PORTD
CALL ENVIAR
MOVLW 'O'
MOVWF PORTD
CALL ENVIAR
MOVLW 'M'
MOVWF PORTD
CALL ENVIAR
MOVLW 'E'
MOVWF PORTD
CALL ENVIAR
MOVLW 'T'
MOVWF PORTD
CALL ENVIAR
MOVLW 'R'
MOVWF PORTD
CALL ENVIAR
MOVLW 'O'
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW '2'
MOVWF PORTD
CALL ENVIAR
RETURN
; FUENTE -> 1
FUENTE_1:
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'F'
MOVWF PORTD
CALL ENVIAR
MOVLW 'U'
MOVWF PORTD
CALL ENVIAR
MOVLW 'E'
MOVWF PORTD
CALL ENVIAR
MOVLW 'N'
MOVWF PORTD
CALL ENVIAR
MOVLW 'T'
MOVWF PORTD
CALL ENVIAR
MOVLW 'E'
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW '1'
MOVWF PORTD
CALL ENVIAR
RETURN
; FUENTE -> 2
FUENTE_2:
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR
MOVLW 'F'
MOVWF PORTD
CALL ENVIAR
MOVLW 'U'
MOVWF PORTD
CALL ENVIAR
MOVLW 'E'
MOVWF PORTD
CALL ENVIAR
MOVLW 'N'
MOVWF PORTD
CALL ENVIAR
MOVLW 'T'
MOVWF PORTD
CALL ENVIAR
MOVLW 'E'
MOVWF PORTD
CALL ENVIAR
MOVLW ' '
MOVWF PORTD
CALL ENVIAR

```




```

        MOVLW    '2'
        MOVWF    PORTD
        CALL     ENVIAR
        RETURN
; FUENTE -> 3
FUENTE_3:
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'F'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'U'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'E'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'N'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'T'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'E'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    '3'
        MOVWF    PORTD
        CALL     ENVIAR
        RETURN

```

```

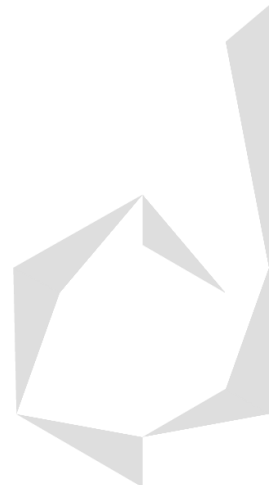
;-----
MENSAJE1:
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'D'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'I'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'S'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'P'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'O'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'S'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'I'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'T'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'I'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'V'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'O'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'S'
        MOVWF    PORTD
        CALL     ENVIAR
        RETURN

```

```

MENSAJE2:
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    ' '
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'P'
        MOVWF    PORTD
        CALL     ENVIAR

```



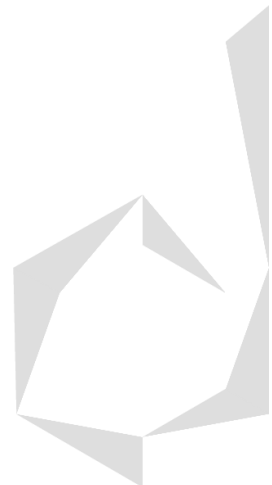
```

        MOVLW    'R'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'O'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'G'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'R'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'A'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'M'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'A'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'B'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'L'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'E'
        MOVWF    PORTD
        CALL     ENVIAR
        MOVLW    'S'
        MOVWF    PORTD
        CALL     ENVIAR
        RETURN

; ----- SUBROUTINAS DE LCD -----
LCD:    BCF      PORTC,0 ; MODO INSTRUCCION
        MOVLW    0X01 ; LIMPIA DISPLAY
        MOVWF    PORTD
        CALL     COMANDO
        MOVLW    0X0C ; SELECCIONA LA PRIMER LINEA
        MOVWF    PORTD
        CALL     COMANDO ; DA DE ALTA EL COMANDO
        MOVLW    0X3C ; SE CONFIGURA EL CURSOR
        MOVWF    PORTD
        CALL     COMANDO
        BSF      PORTC,0 ; MODO DATO
        RETURN
COMANDO: BSF      PORTC,3 ;PONER EL ENABLE EN 1
        CALL     T25MS
        BCF      PORTC,3
        CALL     T25MS
        RETURN
ENVIAR:  BSF      PORTC,0 ; MODO DATO
        CALL     COMANDO
        RETURN
LINEA2:  BCF      PORTC,0 ; MODO INSTRUCCION
        MOVLW    0X0C ; SELECCIONO LINEA 2 EN EL LCD
        MOVWF    PORTD
        CALL     COMANDO
        RETURN

; ----- SUBROUTINAS DE TIEMPO -----
T500MS:  MOVLW    .6
        MOVWF    0X61
        MOVLW    .115
        MOVWF    0X62
        CALL     ST2V
        MOVLW    .103
        MOVWF    0X60
        CALL     ST1V
        NOP
        NOP
        NOP
        NOP
        NOP
        RETURN
T25MS:   MOVLW    .245
        MOVWF    0X61
        MOVLW    .14
        MOVWF    0X62
        CALL     ST2V
        NOP
        NOP
        NOP
        RETURN
T_ESTA:  MOVLW    .2
        MOVWF    0X60
        CALL     ST1V
        NOP
        NOP
        NOP
        NOP
        NOP
        RETURN

```



```
T2TAD:    MOVLW    .1
          MOVWF    0x60
          CALL     ST1V
          RETURN
          END
```

27.- Melodía en Buzzer Pasivo:

27.- Melodía en Buzzer Pasivo.

```
;27.- Melodía en Buzzer Pasivo.
          INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\Configuracion_PIC.asm>
          BCF        TRISB,RB4
          CLRF        TRISC
          BCF        STATUS,RP0      ;B0.
          CLRF        PORTC
          CLRF        PORTB
          CONT        EQU 0x20
          BEAT3       EQU 0x21
          CLRF        CONT
          CLRF        BEAT3

INICIO    BTFSS        PORTB,0
          GOTO        $-1
          CALL        REBOTE
          BTFSC        PORTB,0
          GOTO        $-1
          CALL        REBOTE
          CLRF        CONT
          BCF        STATUS,Z

;-----INICIO MELODIA-----

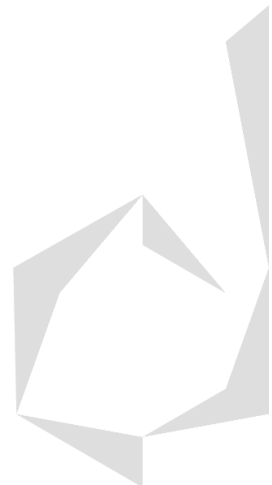
;-----PRIMERAS 2 PARTITURAS-----
UNO_DOS   BCF        STATUS,Z
          BCF        STATUS,C
          CLRF        BEAT3

AA1       CALL        SOL_SOST_2
          INCF        BEAT3,F
          MOVF        BEAT3,W
          XORLW        .77
          BTFSS        STATUS,Z
          GOTO        AA1
          CLRF        BEAT3

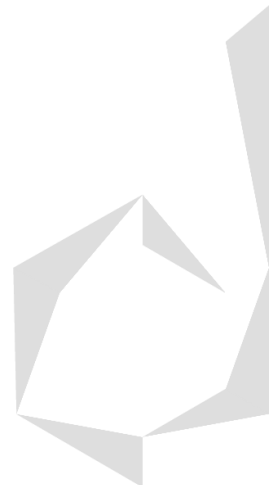
AA2       CALL        DO_SOST_3
          INCF        BEAT3,F
          MOVF        BEAT3,W
          XORLW        .103
          BTFSS        STATUS,Z
          GOTO        AA2
          CLRF        BEAT3

AA3       CALL        MI_3
          INCF        BEAT3,F
          MOVF        BEAT3,W
          XORLW        .122
          BTFSS        STATUS,Z
          GOTO        AA3
          INCF        CONT,F
          MOVF        CONT,W
          XORLW        0x08
          BTFSS        STATUS,Z
          GOTO        UNO_DOS
          CLRF        CONT
          BCF        STATUS,Z

;-----TERCERA PARTITURA-----
TRES     CLRF        BEAT3
```



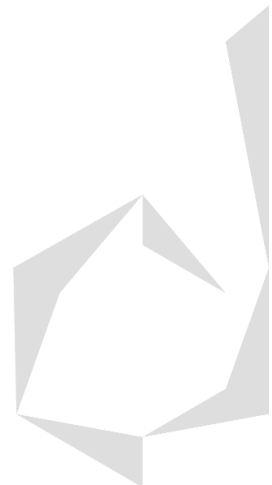
AA4	CALL INCF MOVF XORLW BTFSS GOTO CLRF	LA_2 BEAT3,F BEAT3,W .82 STATUS,Z AA4 BEAT3
AA5	CALL INCF MOVF XORLW BTFSS GOTO CLRF	DO_SOST_3 BEAT3,F BEAT3,W .103 STATUS,Z AA5 BEAT3
AA6	CALL INCF MOVF XORLW BTFSS GOTO INCF MOVF XORLW BTFSS GOTO CLRF	MI_3 BEAT3,F BEAT3,W .122 STATUS,Z AA6 CONT,F CONT,W 0X02 STATUS,Z TRES BEAT3
AA7	CALL INCF MOVF XORLW BTFSS GOTO CLRF	LA_2 BEAT3,F BEAT3,W .82 STATUS,Z AA7 BEAT3
AA8	CALL INCF MOVF XORLW BTFSS GOTO CLRF	RE_E_3 BEAT3,F BEAT3,W .109 STATUS,Z AA8 BEAT3
AA9	CALL INCF MOVF XORLW BTFSS GOTO CLRF	FA_SOST_3 BEAT3,F BEAT3,W .137 STATUS,Z AA9 BEAT3
AB1	CALL INCF MOVF XORLW BTFSS GOTO CLRF	LA_2 BEAT3,F BEAT3,W .82 STATUS,Z AB1 BEAT3
AB2	CALL INCF MOVF XORLW BTFSS GOTO CLRF	RE_SOST_3 BEAT3,F BEAT3,W .115 STATUS,Z AB2 BEAT3
AB3	CALL	FA_SOST_3



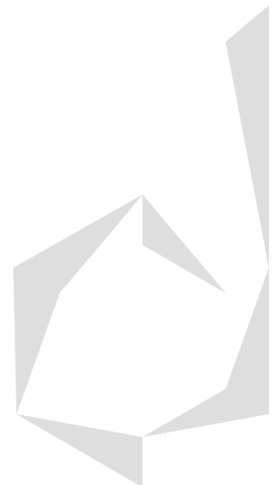
```

        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .137
        BTFSS     STATUS,Z
        GOTO      AB3
;-----CUARTA PARTITURA-----
CUATRO      CLRF      BEAT3
AB4
        CALL     SOL_SOST_2
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .77
        BTFSS     STATUS,Z
        GOTO     AB4
        CLRF     BEAT3
AB5
        CALL     SI_BEM_2
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .86
        BTFSS     STATUS,Z
        GOTO     AB5
        CLRF     BEAT3
AB6
        CALL     FA_SOST_3
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .137
        BTFSS     STATUS,Z
        GOTO     AB6
        CLRF     BEAT3
AB7
        CALL     SOL_SOST_2
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .77
        BTFSS     STATUS,Z
        GOTO     AB7
        CLRF     BEAT3
AB8
        CALL     DO_SOST_3
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .103
        BTFSS     STATUS,Z
        GOTO     AB8
        CLRF     BEAT3
AB9
        CALL     MI_3
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .122
        BTFSS     STATUS,Z
        GOTO     AB9
        CLRF     BEAT3
AC1
        CALL     SOL_SOST_2
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .77
        BTFSS     STATUS,Z
        GOTO     AC1
        CLRF     BEAT3
AC2
        CALL     DO_SOST_3
        INCF     BEAT3,F
        MOVF     BEAT3,W
        XORLW     .103
        BTFSS     STATUS,Z
        GOTO     AC2

```



	CLRF	BEAT3
AC3	CALL	RE_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.115
	BTFS	STATUS,Z
	GOTO	AC3
	CLRF	BEAT3
AC4	CALL	FA_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.69
	BTFS	STATUS,Z
	GOTO	AC4
	CLRF	BEAT3
AC5	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFS	STATUS,Z
	GOTO	AC5
	CLRF	BEAT3
AC6	CALL	RE_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.115
	BTFS	STATUS,Z
	GOTO	AC6
	CLRF	CONT
	BCF	STATUS,Z
;-----QUINTA PARTITURA-----		
QUINTO		
	CLRF	BEAT3
AC7	CALL	MI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.61
	BTFS	STATUS,Z
	GOTO	AC7
	CLRF	BEAT3
AC8	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFS	STATUS,Z
	GOTO	AC8
	CLRF	BEAT3
AC9	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFS	STATUS,Z
	GOTO	AC9
CINCO SOLDOMI		
	CLRF	BEAT3
AD1	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFS	STATUS,Z
	GOTO	AD1
	CLRF	BEAT3
AD2		



```

CALL      DO_SOST_3
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .103
BTFSS     STATUS,Z
GOTO      AD2
CLRF      BEAT3

AD3

CALL      MI_3
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .122
BTFSS     STATUS,Z
GOTO      AD3
INCF      CONT,F
MOVF      CONT,W
XORLW     0X03
BTFSS     STATUS,Z
GOTO      CINCO SOLDOMI
CLRF      CONT
BCF       STATUS,Z

;-----SEXTA PARTITURA-----
SEXTA
CLRF      BEAT3

AD4

CALL      SOL_SOST_2
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .77
BTFSS     STATUS,Z
GOTO      AD4
CLRF      BEAT3

AD5

CALL      RE_SOST_3
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .115
BTFSS     STATUS,Z
GOTO      AD5
CLRF      BEAT3

AD6

CALL      FA_SOST_3
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .137
BTFSS     STATUS,Z
GOTO      AD6
INCF      CONT,F
MOVF      CONT,W
XORLW     0X04
BTFSS     STATUS,Z
GOTO      SEXTA
CLRF      CONT
BCF       STATUS,Z

;-----SEPTIMA PARTITURA-----
SEPTIMA
CLRF      BEAT3

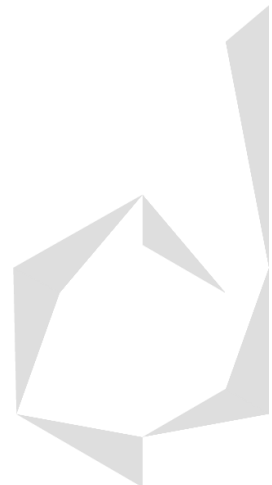
AD7

CALL      SOL_SOST_2
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .77
BTFSS     STATUS,Z
GOTO      AD7
CLRF      BEAT3

AD8

CALL      DO_SOST_3
INCF      BEAT3,F
MOVF      BEAT3,W
XORLW     .103
BTFSS     STATUS,Z

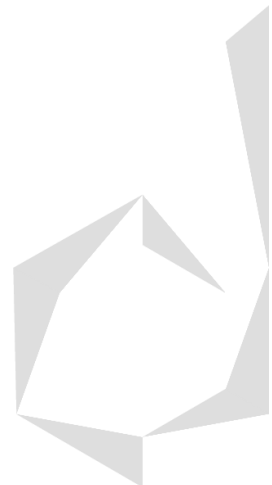
```



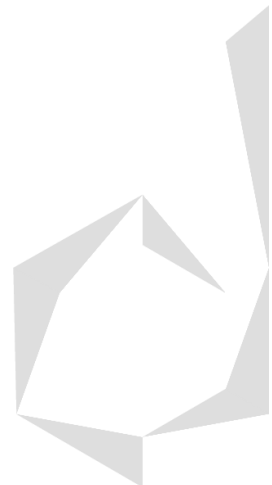
	GOTO	AD8
	CLRF	BEAT3
AD9		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AD9
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x02
	BTFSS	STATUS,Z
	GOTO	SEPTIMA
	CLRF	CONT
	BCF	STATUS,Z
SEPTIMALADOFA		
	CLRF	BEAT3
AE1		
	CALL	LA_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.82
	BTFSS	STATUS,Z
	GOTO	AE1
	CLRF	BEAT3
AE2		
	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFSS	STATUS,Z
	GOTO	AE2
	CLRF	BEAT3
AE3		
	CALL	FA_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.137
	BTFSS	STATUS,Z
	GOTO	AE3
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x02
	BTFSS	STATUS,Z
	GOTO	SEPTIMALADOFA
	CLRF	CONT
	BCF	STATUS,Z
;-----OCTAVA PARTITURA-----		
OCTAVA		
	CLRF	BEAT3
AE4		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AE4
	CLRF	BEAT3
AE5		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AE5
	CLRF	BEAT3
AE6		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W



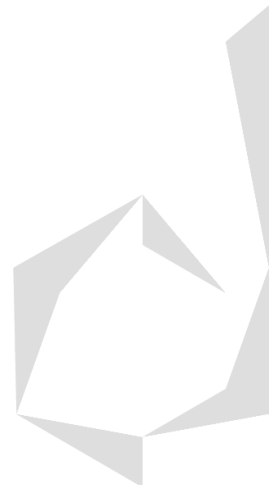
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AE6
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x02
	BTFSS	STATUS,Z
	GOTO	OCTAVA
	CLRF	CONT
	BCF	STATUS,Z
OCTAVALASIRE		
	CLRF	BEAT3
AE7		
	CALL	LA_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.82
	BTFSS	STATUS,Z
	GOTO	AE7
	CLRF	BEAT3
AE8		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AE8
	CLRF	BEAT3
AE9		
	CALL	RE_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.115
	BTFSS	STATUS,Z
	GOTO	AE9
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x02
	BTFSS	STATUS,Z
	GOTO	OCTAVALASIRE
	CLRF	CONT
	BCF	STATUS,Z
;-----NOVENA PARTITURA-----		
NOVENA		
	CLRF	BEAT3
AF1		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AF1
	CLRF	BEAT3
AF2		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AF2
	CLRF	BEAT3
AF3		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AF3
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x04



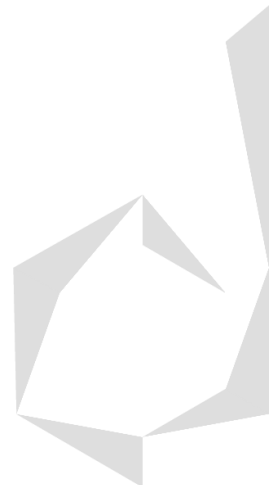
	BTFSS	STATUS,Z
	GOTO	NOVENA
	CLRF	CONT
	BCF	STATUS,Z
DECIMA		
	CLRF	BEAT3
AF4		
	CALL	SOL_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.73
	BTFSS	STATUS,Z
	GOTO	AF4
	CLRF	BEAT3
AF5		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AF5
	CLRF	BEAT3
AF6		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AF6
;-----DECIMA PARTITURA-----		
DECIMASOLSIMI		
	CLRF	BEAT3
AF7		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AF7
	CLRF	BEAT3
AF8		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AF8
	CLRF	BEAT3
AF9		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AF9
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x03
	BTFSS	STATUS,Z
	GOTO	DECIMASOLSIMI
	CLRF	CONT
	BCF	STATUS,Z
;-----ONCEAVA PARTITURA-----		
DECIMA1		
	CLRF	BEAT3
AG1		
	CALL	SOL_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.73
	BTFSS	STATUS,Z



	GOTO	AG1
	CLRF	BEAT3
AG2		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AG2
	CLRF	BEAT3
AG3		
	CALL	FA_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.129
	BTFSS	STATUS,Z
	GOTO	AG3
DECIMA1SOLSIFA		
	CLRF	BEAT3
AG4		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AG4
	CLRF	BEAT3
AG5		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AG5
	CLRF	BEAT3
AG6		
	CALL	FA_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.137
	BTFSS	STATUS,Z
	GOTO	AG6
	INCF	CONT,F
	MOVF	CONT,W
	XORLW	0x03
	BTFSS	STATUS,Z
	GOTO	DECIMA1SOLSIFA
	CLRF	CONT
	BCF	STATUS,Z
	;-----DOCEAVA PARTITURA-----	
DECIMA2		
	CLRF	BEAT3
AG7		
	CALL	SOL_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.73
	BTFSS	STATUS,Z
	GOTO	AG7
	CLRF	BEAT3
AG8		
	CALL	DO_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.97
	BTFSS	STATUS,Z
	GOTO	AG8
	CLRF	BEAT3
AG9		
	CALL	MI_3
	INCF	BEAT3,F



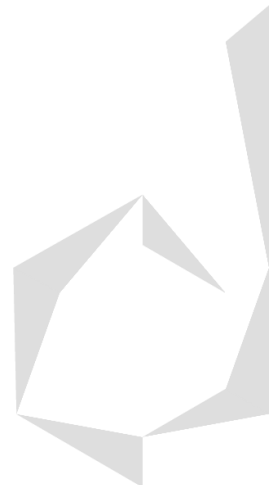
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AG9
	CLRF	BEAT3
AH1		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AH1
	CLRF	BEAT3
AH2		
	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AH2
	CLRF	BEAT3
AH3		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AH3
	CLRF	BEAT3
AH4		
	CALL	SOL_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.77
	BTFSS	STATUS,Z
	GOTO	AH4
	CLRF	BEAT3
AH5		
	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFSS	STATUS,Z
	GOTO	AH5
	CLRF	BEAT3
AH6		
	CALL	MI_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.122
	BTFSS	STATUS,Z
	GOTO	AH6
	CLRF	BEAT3
AH7		
	CALL	FA_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.69
	BTFSS	STATUS,Z
	GOTO	AH7
	CLRF	BEAT3
AH8		
	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFSS	STATUS,Z
	GOTO	AH8
	CLRF	BEAT3
AH9		
	CALL	MI_3



```

        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .122
        BTFSS     STATUS,Z
        GOTO      AH9
; ; -----TRECEAVA PARTITURA-----
DECIMA3
        CLRF      BEAT3
AI1
        CALL      FA_SOST_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .69
        BTFSS     STATUS,Z
        GOTO      AI1
        CLRF      BEAT3
AI2
        CALL      SI_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .91
        BTFSS     STATUS,Z
        GOTO      AI2
        CLRF      BEAT3
AI3
        CALL      RE_E_3
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .109
        BTFSS     STATUS,Z
        GOTO      AI3
        CLRF      BEAT3
AI4
        CALL      FA_SOST_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .69
        BTFSS     STATUS,Z
        GOTO      AI4
        CLRF      BEAT3
AI5
        CALL      SI_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .91
        BTFSS     STATUS,Z
        GOTO      AI5
        CLRF      BEAT3
AI6
        CALL      RE_SOST_3
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .115
        BTFSS     STATUS,Z
        GOTO      AI6
        CLRF      BEAT3
AI7
        CALL      SOL_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .73
        BTFSS     STATUS,Z
        GOTO      AI7
        CLRF      BEAT3
AI8
        CALL      SI_2
        INCF      BEAT3,F
        MOVF      BEAT3,W
        XORLW     .91
        BTFSS     STATUS,Z
        GOTO      AI8

```



	CLRF	BEAT3
AI9	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFSS	STATUS,Z
	GOTO	AI9
	CLRF	BEAT3
AJ1	CALL	MI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.61
	BTFSS	STATUS,Z
	GOTO	AJ1
	CLRF	BEAT3
AJ2	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AJ2
	CLRF	BEAT3
AJ3	CALL	DO_SOST_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.103
	BTFSS	STATUS,Z
	GOTO	AJ3
; CATORCEAVA PARTITURA		
DECIMA4	CLRF	BEAT3
AJ4	CALL	FA_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.69
	BTFSS	STATUS,Z
	GOTO	AJ4
	CLRF	BEAT3
AJ5	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.91
	BTFSS	STATUS,Z
	GOTO	AJ5
	CLRF	BEAT3
AJ6	CALL	RE_E_3
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.109
	BTFSS	STATUS,Z
	GOTO	AJ6
	CLRF	BEAT3
AJ7	CALL	FA_SOST_2
	INCF	BEAT3,F
	MOVF	BEAT3,W
	XORLW	.69
	BTFSS	STATUS,Z
	GOTO	AJ7
	CLRF	BEAT3
AJ8	CALL	SI_2
	INCF	BEAT3,F
	MOVF	BEAT3,W



```

XORLW      .91
BTFSS      STATUS,Z
GOTO       AJ8
CLRFB      BEAT3

AJ9
CALL       RE_SOST_3
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .115
BTFSS      STATUS,Z
GOTO       AJ9
CLRFB      BEAT3

AK1
CALL       FA_SOST_2
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .69
BTFSS      STATUS,Z
GOTO       AK1
CLRFB      BEAT3

AK2
CALL       LA_SOST_2
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .86
BTFSS      STATUS,Z
GOTO       AK2
CLRFB      BEAT3

AK3
CALL       DO_SOST_3
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .103
BTFSS      STATUS,Z
GOTO       AK3
CLRFB      BEAT3

AK4
CALL       FA_SOST_2
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .69
BTFSS      STATUS,Z
GOTO       AK4
CLRFB      BEAT3

AK5
CALL       LA_2
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .82
BTFSS      STATUS,Z
GOTO       AK5
CLRFB      BEAT3

AK6
CALL       DO_SOST_3
INCF       BEAT3,F
MOVF       BEAT3,W
XORLW      .103
BTFSS      STATUS,Z
GOTO       AK6

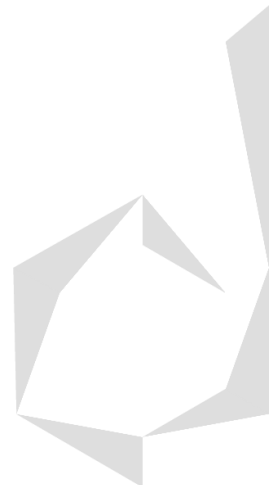
```

```
GOTO      INICIO
```

```

;-----REBOTE-----
REBOTE
MOVLW      .245
MOVWF      0X61
MOVLW      .14
MOVWF      0X62
CALL       ST2V
RETURN

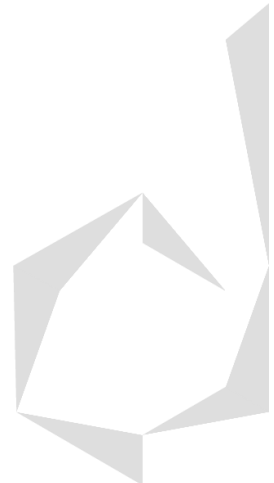
```



```

;A UNA FRECUENCIA DE 54BPM SE REPITE 51 VECES
DO_SOST_2      BSF          PORTC,2
                MOVLW       .5
                MOVWF       0X61
                MOVLW       .102
                MOVWF       0X62
                CALL        ST2V
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                BCF          PORTC,2
                MOVLW       .5
                MOVWF       0X61
                MOVLW       .102
                MOVWF       0X62
                CALL        ST2V
                NOP
                NOP
                NOP
                RETURN
;----A UNA FRECUENCIA DE 54BPM SE REPITE 58 VECES-----
RE_SOST_2      BSF          PORTC,2
                MOVLW       .26
                MOVWF       0X61
                MOVLW       .17
                MOVWF       0X62
                CALL        ST2V
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                BCF          PORTC,2
                MOVLW       .26
                MOVWF       0X61
                MOVLW       .17
                MOVWF       0X62
                CALL        ST2V
                NOP
                NOP
                RETURN
;----A UNA FRECUENCIA DE 54BPM SE REPITE 61 VECES-----
MI_2          BSF          PORTC,2
                MOVLW       .7
                MOVWF       0X61
                MOVLW       .61
                MOVWF       0X62
                CALL        ST2V
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                BCF          PORTC,2
                MOVLW       .7
                MOVWF       0X61

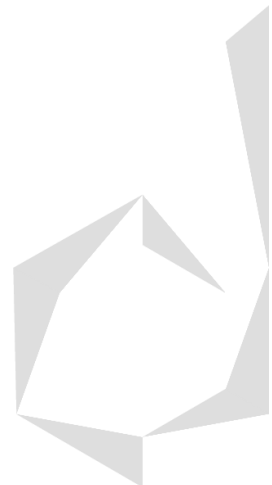
```




```

        MOVLW      .61
        MOVWF      0x62
        CALL       ST2V
        NOP
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 69 VECES-----
FA_SOST_2
        BSF        PORTC,2
        MOVLW      .17
        MOVWF      0x61
        MOVLW      .22
        MOVWF      0x62
        CALL       ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BCF        PORTC,2
        MOVLW      .17
        MOVWF      0x61
        MOVLW      .22
        MOVWF      0x62
        CALL       ST2V
        NOP
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 73 VECES-----
SOL_2
        BSF        PORTC,2
        MOVLW      .4
        MOVWF      0x61
        MOVLW      .90
        MOVWF      0x62
        CALL       ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BCF        PORTC,2
        MOVLW      .4
        MOVWF      0x61
        MOVLW      .90
        MOVWF      0x62
        CALL       ST2V
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 77 VECES-----
SOL_SOST_2
        BSF        PORTC,2
        MOVLW      .52
        MOVWF      0x61
        MOVLW      .6
        MOVWF      0x62
        CALL       ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BCF        PORTC,2
        MOVLW      .52
        MOVWF      0x61
        MOVLW      .6
        MOVWF      0x62

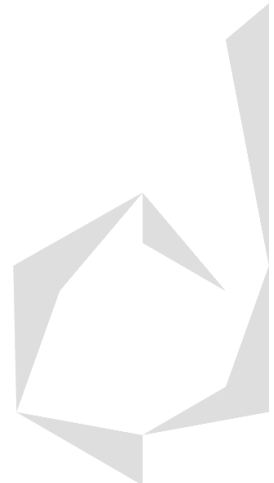
```



```

        CALL          ST2V
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 82 VECES-----
LA_2
        BSF           PORTC,2
        MOVLW         .4
        MOVWF         0X61
        MOVLW         .80
        MOVWF         0X62
        CALL          ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BCF           PORTC,2
        MOVLW         .4
        MOVWF         0X61
        MOVLW         .80
        MOVWF         0X62
        CALL          ST2V
        NOP
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 86 VECES-----
LA_SOST_2
        BSF           PORTC,2
        MOVLW         .85
        MOVWF         0X61
        MOVLW         .3
        MOVWF         0X62
        CALL          ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BCF           PORTC,2
        MOVLW         .85
        MOVWF         0X61
        MOVLW         .3
        MOVWF         0X62
        CALL          ST2V
        NOP
        NOP
        NOP
        NOP
        NOP
        RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 91 VECES-----
SI_2
        BSF           PORTC,2
        MOVLW         .3
        MOVWF         0X61
        MOVLW         .95
        MOVWF         0X62
        CALL          ST2V
        NOP
        NOP
        NOP
        NOP

```



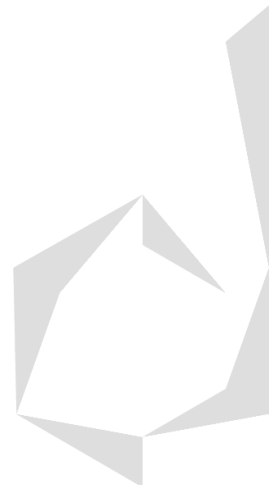
```

NOP
NOP
NOP
NOP
NOP
BCF      PORTC,2
MOVLW   .3
MOVWF   0x61
MOVLW   .95
MOVWF   0x62
CALL    ST2V
NOP
NOP
RETURN

;----A UNA FRECUENCIA DE 54BPM SE REPITE 86 VECES-----
SI_BEM_2
BSF      PORTC,2
MOVLW   .85
MOVWF   0x61
MOVLW   .3
MOVWF   0x62
CALL    ST2V
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
BCF      PORTC,2
MOVLW   .85
MOVWF   0x61
MOVLW   .3
MOVWF   0x62
CALL    ST2V
NOP
NOP
NOP
NOP
RETURN

;----A UNA FRECUENCIA DE 54BPM SE REPITE 97 VECES-----
DO_3
BSF      PORTC,2
MOVLW   .12
MOVWF   0x61
MOVLW   .22
MOVWF   0x62
CALL    ST2V
NOP
NOP
NOP
NOP
NOP
NOP
NOP
BCF      PORTC,2
MOVLW   .12
MOVWF   0x61
MOVLW   .22
MOVWF   0x62
CALL    ST2V
RETURN

```



```
;----A UNA FRECUENCIA DE 54BPM SE REPITE 103 VECES-----  
DO_SOST_3
```

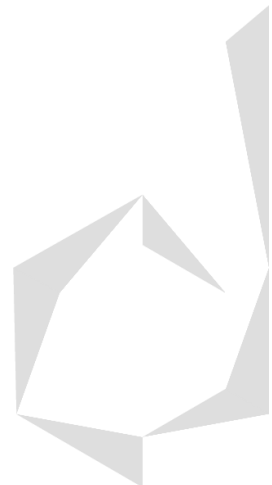
```
    BSF          PORTC,2  
    MOVLW        .6  
    MOVWF        0X61  
    MOVLW        .42  
    MOVWF        0X62  
    CALL         ST2V  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    BCF          PORTC,2  
    MOVLW        .6  
    MOVWF        0X61  
    MOVLW        .42  
    MOVWF        0X62  
    CALL         ST2V  
    NOP  
    RETURN
```

```
;----A UNA FRECUENCIA DE 54BPM SE REPITE 109 VECES-----  
RE_E_3
```

```
    BSF          PORTC,2  
    MOVLW        .241  
    MOVWF        0X60  
    CALL         ST1V  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    BCF          PORTC,2  
    MOVLW        .241  
    MOVWF        0X60  
    CALL         ST1V  
    NOP  
    NOP  
    RETURN
```

```
;----A UNA FRECUENCIA DE 54BPM SE REPITE 115 VECES-----  
RE_SOST_3
```

```
    BSF          PORTC,2  
    MOVLW        .227  
    MOVWF        0X60  
    CALL         ST1V  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    BCF          PORTC,2  
    MOVLW        .227  
    MOVWF        0X60  
    CALL         ST1V  
    NOP  
    NOP
```



```

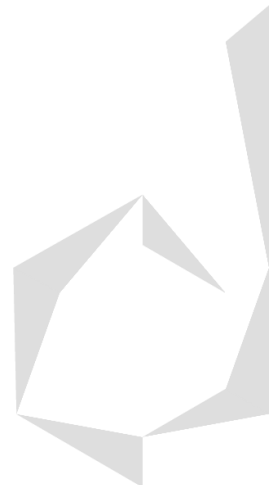
NOP
NOP
NOP
RETURN
;-----A UNA FRECUENCIA DE 54BPM SE REPITE 122 VECES-----
MI_3
    BSF        PORTC,2
    MOVLW      .215
    MOVWF      0X60
    CALL       ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF        PORTC,2
    MOVLW      .215
    MOVWF      0X60
    CALL       ST1V
    RETURN

;-----A UNA FRECUENCIA DE 65BPM SE REPITE 129 VECES-----
FA_3
    BSF        PORTC,2
    MOVLW      .202
    MOVWF      0X60
    CALL       ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF        PORTC,2
    MOVLW      .202
    MOVWF      0X60
    CALL       ST1V
    NOP
    NOP
    NOP
    NOP
    RETURN

;-----A UNA FRECUENCIA DE 54BPM SE REPITE 137 VECES-----
FA_SOST_3
    BSF        PORTC,2
    MOVLW      .191
    MOVWF      0X60
    CALL       ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF        PORTC,2
    MOVLW      .191
    MOVWF      0X60
    CALL       ST1V
    NOP
    RETURN

;-----A UNA FRECUENCIA DE 54BPM SE REPITE 145 VECES-----
SOL_3

```



```

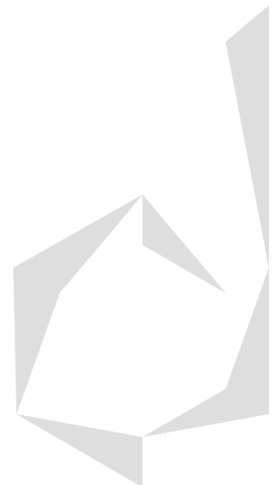
    BSF          PORTC,2
    MOVLW        .180
    MOVWF        0X60
    CALL         ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF          PORTC,2
    MOVLW        .180
    MOVWF        0X60
    CALL         ST1V
    NOP
    NOP
    RETURN

;----A UNA FRECUENCIA DE 54BPM SE REPITE 154 VECES-----
SOL_SOST_3
    BSF          PORTC,2
    MOVLW        .170
    MOVWF        0X60
    CALL         ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF          PORTC,2
    MOVLW        .170
    MOVWF        0X60
    CALL         ST1V
    RETURN

;----A UNA FRECUENCIA DE 54BPM SE REPITE 163 VECES-----
LA_3
    BSF          PORTC,2
    MOVLW        .160
    MOVWF        0X60
    CALL         ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF          PORTC,2
    MOVLW        .160
    MOVWF        0X60
    CALL         ST1V
    NOP
    NOP
    NOP
    RETURN

;----A UNA FRECUENCIA DE 54BPM SE REPITE 173 VECES-----
LA_SOST_3
    BSF          PORTC,2
    MOVLW        .151
    MOVWF        0X60
    CALL         ST1V

```



```

NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
BCF          PORTC,2
MOVLW       .151
MOVWF       0X60
CALL        ST1V
NOP
NOP
RETURN

;-----A UNA FRECUENCIA DE 54BPM SE REPITE 183 VECES-----
SI_3
    BSF          PORTC,2
    MOVLW       .142
    MOVWF       0X60
    CALL        ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF          PORTC,2
    MOVLW       .142
    MOVWF       0X60
    CALL        ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    RETURN

;-----A UNA FRECUENCIA DE 54BPM SE REPITE 205 VECES-----
DO_SOST_4
    BSF          PORTC,2
    MOVLW       .127
    MOVWF       0X60
    CALL        ST1V
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF          PORTC,2
    MOVLW       .127
    MOVWF       0X60
    CALL        ST1V
    RETURN

INCLUDE      <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
Ejercicios PIC16F887\SubRutinaTiempo.asm>
END

```

