INGENIERÍA MECATRÓNICA

DI_CERO

DIEGO CERVANTES RODRÍGUEZ

ENSAMBLADOR EN MICROCONTROLADORES

MPLAB X IDE v5.50

Ejercicios Básicos Ensamblador MPLABX

Contenido

Ejercicios Básicos de Ensamblador MPLABX

1.-Recibir Datos por un Puerto y Sacarlos en Otro:

1.- Lea el puerto B (PORTB) y saque el valor por el puerto C (PORTC).

Código Ensamblador:

```
;1.- Lea el puerto B (PORTB) y saque el valor por el puerto C (PORTC).

;La directiva PROCESSOR debe ser la primera que venga en el programa y sirve
;para indicar qué dispositivo estoy programando.
        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, 0X2BE4
   ;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).
   ;11:  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>

;La directiva ORG (u origen) le indica al programa desde qué dirección de la
;memoria FLASH debe empezar a guardar todas las instrucciones del código y para
;ello debe recibir una dirección de 13 bits.
       ORG       0X0000

;A PARTIR DE AHORA USAREMOS LAS 35 INSTRUCCIONES DEL PIC
;La directiva EQU asocia nombres a los registros de uso específico y además con
```

```
;ella puedo darle a algún registro de propósito general un nombre en específico,
;se usa mucho cuando se utilizan las 35 instrucciones del PIC.

;Siempre al inicializar el PIC nos encontramos en el banco 0 de la RAM porque
;los bits RP0 y RP1 del registro STATUS se inicializan con valor 0, se cambia de
;banco poniendo en este órden RP1 y RP0 porque el bit RP1 es el bit 8 del
;registro STATUS (osea el bit de su posición 7) y RP0 es el bit 7 del registro
;STATUS (posición 6), eso hace que se vea en el órden correcto como se ve abajo,
;el banco en el que nos encontramos cuando en la ventana de SFRs que permite
;ver el estado de los registros de propósito específico, vea el estado del
;registro STATUS.
        ;Banco 0: RP1 = 0, RP0 = 0
        ;Banco 1: RP1 = 0, RP0 = 1
        ;Banco 2: RP1 = 1, RP0 = 0
        ;Banco 3: RP1 = 1, RP0 = 1

;Ahora vamos a primero limpiar los pines de todos los puertos, ya que todos se
;encuentran en el banco 0 y después de un reset en el datasheet estos se
;muestran como x, esto implica que no sabemos con qué valor (1 o 0) se
;iniciarán, por lo que manualmente debemos poner sus bits como 0 (limpiarlos).
        ;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 pone la bandera Z = 1.
        CLRF  PORTA ;Hace 0 (limpia) todos los pines del puerto A
        CLRF  PORTB ;Hace 0 (limpia) todos los pines del puerto B
        CLRF  PORTC ;Hace 0 (limpia) todos los pines del puerto C
        CLRF  PORTD ;Hace 0 (limpia) todos los pines del puerto D
        CLRF  PORTE ;Hace 0 (limpia) todos los pines del puerto E

;Ahora debo indicar si mis pines serán entradas o salidas y si serán digitales
;o analógicos, para ello primero debo indicar si son digitales o analógicos y
;después indicar si son entradas o salidas, sino el PIC no hace caso.

;Los únicos puertos del PIC16F887 que pueden ser entradas/salidas analógicas o
;digitales son los puertos A, B y E (los demás siempre son digitales), para
;indicar si los pines de los puertos A y E son digitales o analógicos debo
;cambiar los bits del registro ANSEL individualmente y para indicar si los pines
;del puerto B son digitales o analógicos debo cambiar los bits del registro
;ANSELH (los registros ANSEL y ANSELH se inicializan con todos sus bits en 1,
;osea como pines analógicos), donde:
        ;Bit del registro ANSEL o ANSELH: 1 = Analógico
        ;Bit del registro ANSEL o ANSELH: 0 = Digital
;Para poder hacer esto me debo trasladar al banco 3, que es donde se encuentran
;los registros ANSEL y ANSELH, cambiando los bits RP0 y RP1 del registro STATUS.
        ;BSF   F, B: Pone un 1 en el bit B de la dirección F del registro
        ;RAM.
        ;BCF   F, B: Pone un 0 en el bit B de la dirección F del registro
        ;RAM.
        ;La posición del bit B en el registro F se puede indicar contando
        ;desde cero en decimal, osea poniendo 0, 1, 2, 3, 4, 5, 6 o 7.
        ;Pero como en este caso se usa un registro de propósito específico
        ;(el registro STATUS) en vez de poner un número hexadecimal para
        ;indicar el registro F, pondré su directiva EQU (osea el nombre del
        ;registro) y también en vez de poner un número decimal como B para
        ;indicar el bit que quiero afectar, pondré su directiva EQU (osea el
        ;nombre del bit).
        BSF       STATUS, RP1 ;RP1 = 1
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 3

        ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  ANSEL    ;Hace todos los pines de los puertos A y E digitales.
        CLRF  ANSELH   ;Hace todos los pines del puerto B digitales.

;Ahora debo indicar si los puertos son entradas o salidas, para hacer esto debo
;modificar los registros TRISA, TRISB, TRISC, TRISD y TRISE; la mayoría se
;encuentran en el banco 1, por lo que debo cambiar el estado de los bits RP0 y
;RP1 del registro STATUS
```

```
        BCF       STATUS, RP1 ;RP1 = 0
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 1
;------------Esta configuración se repetirá en todos los ejercicios-------------

;Ahora si ya puedo resolver el ejercicio:
;1.- Lea el puerto B (PORTB) y saque el valor por el puerto C (PORTC).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso el puerto B es de entrada y el C es de salida, por lo que solo
;debo cambiar al registro TRIS asociado al puerto C:
        ;CLRF   F  (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  TRISC ;Hace que todos los pines del puerto C sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        BCF       STATUS, RP1 ;RP1 = 0
        BCF       STATUS, RP0 ;RP0 = 0
        ;Con esto ya estoy en el banco 0

        ;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.
REPITE:     MOVF  PORTB, W ;PORTB = W
        ;Con esta instrucción el programa toma lo que haya en todos los
        ;pines del puerto B y lo guarda en el acumulador W.

        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTC ;PORTB = W = PORTC
        ;Toma lo que hay en el acumulador W que viene del puerto B y lo
        ;pasa al puerto C.

        ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
        ;código, indicado por un signo de pesos seguido de un signo menos y
        ;el número de instrucciones hacia atrás que quiero que brinque el
        ;programa o por una directiva EQU que tenga el nombre de la parte
        ;del código a donde quiero que brinque el programa que se debe poner
        ;en el lugar del valor literal k.
        GOTO  REPITE
        ;Todos los códigos en ensamblador deben tener una instrucción GOTO
        ;para que el PIC repita su función indefinidamente.

        ;Y además deben acabar con la directiva END.
        END
```

2.-Suma Binaria con un Puerto y un Registro:

2.- Lea el puerto B (PORTB), sume 0X54 y saque el resultado por el puerto C (PORTC).

Código Ensamblador:

```
;2.- Lea el puerto B (PORTB), sume 0X54 y saque el resultado por el
;puerto C (PORTC).

;La directiva PROCESSOR debe ser la primera que venga en el programa y sirve
;para indicar qué dispositivo estoy programando.
```

```
        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, 0X2BE4
;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 y habilita el
;Puerto B como entradas y salidas digitales o analógicas (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).
;11:   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), esta configuración está relacionada con la de la palabra de
;configuración anterior.
;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>

;La directiva ORG (u origen) le indica al programa desde qué dirección de la
;memoria FLASH debe empezar a guardar todas las instrucciones del código y para
;ello debe recibir una dirección de 13 bits.
        ORG       0X0000

;A PARTIR DE AHORA USAREMOS LAS 35 INSTRUCCIONES DEL PIC
;La directiva EQU asocia nombres a los registros de uso específico y además con
;ella puedo darle a algún registro de propósito general un nombre en específico,
;se usa mucho cuando se utilizan las 35 instrucciones del PIC.

;Siempre al inicializar el PIC nos encontramos en el banco 0 de la RAM porque
;los bits RP0 y RP1 del registro STATUS se inicializan con valor 0, se cambia de
;banco poniendo en este órden RP1 y RP0 porque el bit RP1 es el bit 8 del
;registro STATUS (osea el bit de su posición 7) y RP0 es el bit 7 del registro
;STATUS (posición 6), eso hace que se vea en el órden correcto como se ve abajo,
;el banco en el que nos encontramos cuando en la ventana de SFRs que permite
;ver el estado de los registros de propósito específico, vea el estado del
;registro STATUS.
        ;Banco 0: RP1 = 0, RP0 = 0
        ;Banco 1: RP1 = 0, RP0 = 1
        ;Banco 2: RP1 = 1, RP0 = 0
        ;Banco 3: RP1 = 1, RP0 = 1
```

```
;Ahora vamos a primero limpiar los pines de todos los puertos, ya que todos se
;encuentran en el banco 0 y después de un reset en el datasheet estos se
;muestran como x, esto implica que no sabemos con qué valor (1 o 0) se
;iniciarán, por lo que manualmente debemos poner sus bits como 0 (limpiarlos).
        ;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 pone la bandera Z = 1.
        CLRF  PORTA ;Hace 0 (limpia) todos los pines del puerto A
        CLRF  PORTB ;Hace 0 (limpia) todos los pines del puerto B
        CLRF  PORTC ;Hace 0 (limpia) todos los pines del puerto C
        CLRF  PORTD ;Hace 0 (limpia) todos los pines del puerto D
        CLRF  PORTE ;Hace 0 (limpia) todos los pines del puerto E

;Ahora debo indicar si mis pines serán entradas o salidas y si serán digitales
;o analógicos, para ello primero debo indicar si son digitales o analógicos y
;después indicar si son entradas o salidas, sino el PIC no hace caso.

;Los únicos puertos del PIC16F887 que pueden ser entradas/salidas analógicas o
;digitales son los puertos A, B y E (los demás siempre son digitales), para
;indicar si los pines de los puertos A y E son digitales o analógicos debo
;cambiar los bits del registro ANSEL individualmente y para indicar si los pines
;del puerto B son digitales o analógicos debo cambiar los bits del registro
;ANSELH (los registros ANSEL y ANSELH se inicializan con todos sus bits en 1,
;osea como pines analógicos), donde:
        ;Bit del registro ANSEL o ANSELH: 1 = Analógico
        ;Bit del registro ANSEL o ANSELH: 0 = Digital
;Para poder hacer esto me debo trasladar al banco 3, que es donde se encuentran
;los registros ANSEL y ANSELH, cambiando los bits RP0 y RP1 del registro STATUS.
        ;BSF   F, B: Pone un 1 en el bit B de la dirección F del registro
        ;RAM.
        ;BCF   F, B: Pone un 0 en el bit B de la dirección F del registro
        ;RAM.
        ;La posición del bit B en el registro F se puede indicar contando
        ;desde cero en decimal, osea poniendo 0, 1, 2, 3, 4, 5, 6 o 7.
        ;Pero como en este caso se usa un registro de propósito específico
        ;(el registro STATUS) en vez de poner un número hexadecimal para
        ;indicar el registro F, pondré su directiva EQU (osea el nombre del
        ;registro) y también en vez de poner un número decimal como B para
        ;indicar el bit que quiero afectar, pondré su directiva EQU (osea el
        ;nombre del bit).
        BSF       STATUS, RP1 ;RP1 = 1
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 3

        ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  ANSEL    ;Hace todos los pines de los puertos A y E digitales.
        CLRF  ANSELH   ;Hace todos los pines del puerto B digitales.

;Ahora debo indicar si los puertos son entradas o salidas, para hacer esto debo
;modificar los registros TRISA, TRISB, TRISC, TRISD y TRISE; la mayoría se
;encuentran en el banco 1, por lo que debo cambiar el estado de los bits RP0 y
;RP1 del registro STATUS
        BCF       STATUS, RP1 ;RP1 = 0
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 1
;------------Esta configuración se repetirá en todos los ejercicios-------------

;Ahora si ya puedo resolver el ejercicio:
;2.- Lea el puerto B (PORTB), sume 0X54 y saque el resultado por el
;puerto C (PORTC).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
```

```
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso el puerto B es de entrada y el C es de salida, por lo que solo
;debo cambiar al registro TRIS asociado al puerto C:
        ;CLRF   F  (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  TRISC ;Hace todos los pines del puerto C sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        BCF       STATUS, RP1 ;RP1 = 0
        BCF       STATUS, RP0 ;RP0 = 0
        ;Con esto ya estoy en el banco 0

        ;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.
        MOVF  PORTB, W ;PORTB = W
        ;Con esta instrucción el programa toma lo que haya en todos los
        ;pines del puerto B y lo guarda en el acumulador W.

        ;ADDLW  K (C, DC, Z): Suma el contenido del acumulador W con un
        ;valor cualquiera de 8 bits llamado valor literal K indicado en
        ;hexadecimal, poniendo 0Xnúmero_hexadecimal y el resultado de la
        ;suma lo guarda de nuevo en el acumulador W.
        ADDLW 0X54 ;W = W + 0X54 = PORTB +0X54
        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
        ;parte superior del programa, a la derecha del contador de programa
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
        ;Se levanta la bandera poniéndose en letra mayúscula en la
        ;simulación del código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula en la
        ;simulación del código.

        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTC ;PORTC = W = PORTB + 0X54
        ;Toma lo que hay en el acumulador W que es la suma de lo que viene
        ;del puerto C + 0X54 y lo pasa al puerto B.

        ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
        ;código, indicado por un signo de pesos seguido de un signo menos y
        ;el número de instrucciones hacia atrás que quiero que brinque el
        ;programa o por una directiva EQU que tenga el nombre de la parte
        ;del código a donde quiero que brinque el programa que se debe poner
        ;en el lugar del valor literal k.
        GOTO  $-3 ;Brinca a la instrucción:  MOVF PORTB, W
        ;Todos los códigos en ensamblador deben tener una instrucción GOTO
        ;para que el PIC repita su función indefinidamente.

        ;Y además deben acabar con la directiva END.
        END
```

3.-Resta Binaria con un Puerto y un Registro:

3.- Lea el puerto C (PORTC), reste 0X3C y saque la magnitud del resultado por el puerto B (PORTB).

Código Ensamblador:

```
;3.- Lea el puerto C (PORTC), reste 0X3C y saque la magnitud del resultado por
;el puerto B (PORTB).

;La directiva PROCESSOR debe ser la primera que venga en el programa y sirve
;para indicar qué dispositivo estoy programando.
        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, 0X2BE4
;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 y habilita el
;Puerto B como entradas y salidas digitales o analógicas (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).
;11:   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), esta configuración está relacionada con la de la palabra de
;configuración anterior.
;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>

;La directiva ORG (u origen) le indica al programa desde qué dirección de la
;memoria FLASH debe empezar a guardar todas las instrucciones del código y para
;ello debe recibir una dirección de 13 bits.
        ORG       0X0000

;A PARTIR DE AHORA USAREMOS LAS 35 INSTRUCCIONES DEL PIC
;La directiva EQU asocia nombres a los registros de uso específico y además con
;ella puedo darle a algún registro de propósito general un nombre en específico,
;se usa mucho cuando se utilizan las 35 instrucciones del PIC.
```

```
;Siempre al inicializar el PIC nos encontramos en el banco 0 de la RAM porque
;los bits RP0 y RP1 del registro STATUS se inicializan con valor 0, se cambia de
;banco poniendo en este órden RP1 y RP0 porque el bit RP1 es el bit 8 del
;registro STATUS (osea el bit de su posición 7) y RP0 es el bit 7 del registro
;STATUS (posición 6), eso hace que se vea en el órden correcto como se ve abajo,
;el banco en el que nos encontramos cuando en la ventana de SFRs que permite
;ver el estado de los registros de propósito específico, vea el estado del
;registro STATUS.
        ;Banco 0: RP1 = 0, RP0 = 0
        ;Banco 1: RP1 = 0, RP0 = 1
        ;Banco 2: RP1 = 1, RP0 = 0
        ;Banco 3: RP1 = 1, RP0 = 1

;Ahora vamos a primero limpiar los pines de todos los puertos, ya que todos se
;encuentran en el banco 0 y después de un reset en el datasheet estos se
;muestran como x, esto implica que no sabemos con qué valor (1 o 0) se
;iniciarán, por lo que manualmente debemos poner sus bits como 0 (limpiarlos).
        ;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 pone la bandera Z = 1.
        CLRF  PORTA ;Hace 0 (limpia) todos los pines del puerto A
        CLRF  PORTB ;Hace 0 (limpia) todos los pines del puerto B
        CLRF  PORTC ;Hace 0 (limpia) todos los pines del puerto C
        CLRF  PORTD ;Hace 0 (limpia) todos los pines del puerto D
        CLRF  PORTE ;Hace 0 (limpia) todos los pines del puerto E

;Ahora debo indicar si mis pines serán entradas o salidas y si serán digitales
;o analógicos, para ello primero debo indicar si son digitales o analógicos y
;después indicar si son entradas o salidas, sino el PIC no hace caso.

;PUERTOS DIGITALES O ANALÓGICOS
;Los únicos puertos del PIC16F887 que pueden ser entradas/salidas analógicas o
;digitales son los puertos A, B y E (los demás siempre son digitales), para
;indicar si los pines de los puertos A y E son digitales o analógicos debo
;cambiar los bits del registro ANSEL individualmente y para indicar si los pines
;del puerto B son digitales o analógicos debo cambiar los bits del registro
;ANSELH (los registros ANSEL y ANSELH se inicializan con todos sus bits en 1,
;osea como pines analógicos), donde:
        ;Bit del registro ANSEL o ANSELH: 1 = Analógico
        ;Bit del registro ANSEL o ANSELH: 0 = Digital
;Para poder hacer esto me debo trasladar al banco 3, que es donde se encuentran
;los registros ANSEL y ANSELH, cambiando los bits RP0 y RP1 del registro STATUS.
        ;BSF   F, B: Pone un 1 en el bit B de la dirección F del registro
        ;RAM.
        ;BCF   F, B: Pone un 0 en el bit B de la dirección F del registro
        ;RAM.
        ;La posición del bit B en el registro F se puede indicar contando
        ;desde cero en decimal, osea poniendo 0, 1, 2, 3, 4, 5, 6 o 7.
        ;Pero como en este caso se usa un registro de propósito específico
        ;(el registro STATUS) en vez de poner un número hexadecimal para
        ;indicar el registro F, pondré su directiva EQU (osea el nombre del
        ;registro) y también en vez de poner un número decimal como B para
        ;indicar el bit que quiero afectar, pondré su directiva EQU (osea el
        ;nombre del bit).
        BSF       STATUS, RP1 ;RP1 = 1
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 3

        ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  ANSEL    ;Hace todos los pines de los puertos A y E digitales.
        CLRF  ANSELH   ;Hace todos los pines del puerto B digitales.

;PUERTOS COMO ENTRADAS O SALIDAS
;Ahora debo indicar si los puertos son entradas o salidas, para hacer esto debo
;modificar los registros TRISA, TRISB, TRISC, TRISD y TRISE; la mayoría se
;encuentran en el banco 1, por lo que debo cambiar el estado de los bits RP0 y
;RP1 del registro STATUS
```

```
        BCF       STATUS, RP1 ;RP1 = 0
        BSF       STATUS, RP0 ;RP0 = 1
        ;Con esto ya estoy en el banco 1
;------------Esta configuración se repetirá en todos los ejercicios-------------

;Ahora si ya puedo resolver el ejercicio:
;3.- Lea el puerto C (PORTC), reste 0X3C y saque la magnitud del resultado por
;el puerto B (PORTB).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso el puerto C es de entrada y el B es de salida, por lo que solo
;debo cambiar al registro TRIS asociado al puerto B:
        ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  TRISB ;Hace todos los pines del puerto B sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        BCF       STATUS, RP1 ;RP1 = 0
        BCF       STATUS, RP0 ;RP0 = 0
        ;Con esto ya estoy en el banco 0

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
        ;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.
RESTA:      MOVLW 0X3C ; W = 0X3C

        ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
        ;registro de RAM menos lo que haya en el acumulador W, la resta se
        ;lleva a cabo haciendo una suma entre el registro F y el
        ;complemento A2 de lo que haya en el acumulador W, el resultado de
        ;la resta se guardará en el registro F o en el acumulador W
        ;dependiendo de cuál se ponga en la posición de D.
        SUBWF PORTC, W ; W = PORTC - W = PORTC - 0X3C = PORTC + CA2(0X3C)
        ;Podemos saber el signo del resultado dependiendo del estado de la
        ;bandera C (acarreo):
    ;Si C = 0 el resultado de la resta fue negativo
        ;Sigue la ejecución normal del código
        ;Saca el complemento A2 del resultado para obtener su magnitud,
        ;esto se hace restando 0 hexadecimal menos el resultado negativo.

    ;Si C = 1 el resultado de la resta fue positivo
        ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
        ;Saca el resultado de la resta directamente.

        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
        ;parte superior del programa, a la derecha del contador de programa
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
```

```
        ;Se levanta la bandera poniéndose en letra mayúscula en la
        ;simulación del código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula en la
        ;simulación del código.

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
        ;Sigue la ejecución normal del código.
        ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
        ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
        ;en 1 al contador de programa o PC).
        ;Tarda 2 ciclos de máquina en ejecutarse.
   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, C ;Checar la bandera C del registro STATUS
    ;Si C = 0 el resultado de la resta fue negativo
       ;Sigue la ejecución normal del código.
       ;Saca el complemento A2 del resultado para obtener su magnitud,
       ;esto se hace restando 0 hexadecimal menos el resultado negativo.

    ;Si C = 1 el resultado de la resta fue positivo
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;Saca el resultado de la resta directamente.

       ;SUBLW  K (C, DC, Z): Resta el valor literal K de 8 bits menos lo
       ;que haya en el acumulador W y el resultado de la resta lo guarda de
       ;nuevo en el acumulador W, los números hexadecimales se declaran
       ;poniendo 0Xnúmero_hexadecimal.
       ;PARA SACAR EL COMPLEMENTO A2 DE UN NÚMERO NEGATIVO Y OBTENER SU
       ;MAGNITUD DEBEMOS RESTAR: MAGNITUD = 0 - NÚMERO_NEGATIVO
       SUBLW 0X00
       ;W = 0X00 - W = 0X00 - (PORTC - 0X3C) = CA2(PORTC - 0X3C)

       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTB ;PORTB = W = CA2(PORTC - 0X30) o PORTC - 0X30
       ;Toma lo que hay en el acumulador W que es la resta de lo que viene
       ;del puerto C - 0X30 y lo pasa al puerto B, si el resultado de la
       ;resta fue negativo, osea que la bandera C = 0, se le sacó el
       ;complemento A2 restando cero menos el resultado y sino simplemente
       ;se mostró el resultado de la resta binaria.

       ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
       ;código, indicado por un signo de pesos seguido de un signo menos y
       ;el número de instrucciones hacia atrás que quiero que brinque el
       ;programa o por una directiva EQU que tenga el nombre de la parte
       ;del código a donde quiero que brinque el programa que se debe poner
       ;en el lugar del valor literal k.
       GOTO  RESTA
       ;Todos los códigos en ensamblador deben tener una instrucción GOTO
       ;para que el PIC repita su función indefinidamente.
       ;Y además deben acabar con la directiva END.
       END
```

4.-Suma Binaria entre 2 Puertos:

4.- Lea el puerto B (PORTB) y el puerto C (PORTC), sume los contenidos y saque el resultado por el puerto D (PORTD).

Código Ensamblador:

```
;4.- Lea el puerto B (PORTB) y el puerto C (PORTC), sume los contenidos y saque
;el resultado por el puerto D (PORTD).

;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:
;4.- Lea el puerto B (PORTB) y el puerto C (PORTC), sume los contenidos y saque
;el resultado por el puerto D (PORTD).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada y el D es de salida, por lo que
;solo debo cambiar al registro TRIS asociado al puerto D:
        ;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.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción ADDWF.
        ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
        ;bandera Z, indicando si lo que ingresó al registro es cero o no.
SUM_PUERTOS:MOVF  PORTB, W ;W = PORTB

        ;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.
        ADDWF PORTC, W ;W = W + PORTC = PORTB + PORTC
```

```
;La instrucción afecta las banderas C de acarreo, DC de acarreo
;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
;C = Acerreo: Se pone como 1 lógico cuando al final de una
;operación matemática sobró un 1, también indica el signo del
;resultado después de efectuar una operación (se levanta la bandera
;cuando tiene un 1 lógico y en el programa lo podemos notar porque
;se pone en letra mayúscula, las banderas las podemos ver en la
;parte superior del programa, a la derecha del contador de programa
;o PC).

;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
;realizarse una operación matemática pasó un 1 lógico de los 4
;primeros bits (los de la derecha) del número binario de 8 bits
;a los segundos 4 bits del número binario (los de la izquierda).
;Se levanta la bandera poniéndose en letra mayúscula al simular el
;código.

;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
;operación matemática el resultado es completamente cero. Se
;levanta la bandera poniéndose en letra mayúscula al simular el
;código.

;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F.
MOVWF PORTD ;PORTD = W = PORTB + PORTC

;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por un signo de pesos seguido de un signo menos y
;el número de instrucciones hacia atrás que quiero que brinque el
;programa o por una directiva EQU que tenga el nombre de la parte
;del código a donde quiero que brinque el programa que se debe poner
;en el lugar del valor literal k.
GOTO  SUM_PUERTOS
;Todos los códigos en ensamblador deben tener una instrucción GOTO
;para que el PIC repita su función indefinidamente.

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

5.-Resta Binaria entre 2 Puertos:

5.- Lea el puerto B (PORTB) y el puerto C (PORTC), haga la resta PORTB - PORTC y saque la magnitud del resultado por el puerto D (PORTD).

Código Ensamblador:

```
;5.- Lea el puerto B (PORTB) y el puerto C (PORTC), haga la resta PORTB - PORTC
;y saque la magnitud del resultado por el puerto D (PORTD).

;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:
;5.- Lea el puerto B (PORTB) y el puerto C (PORTC), haga la resta PORTB - PORTC
;y saque la magnitud del resultado por el puerto D (PORTD).
```

```
;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
       ;Bit del registro TRIS: 1 = Entrada (Input)
       ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada y el D es de salida, por lo que
;solo debo cambiar al registro TRIS asociado al puerto D:
       ;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.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
       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

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
       ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
       ;bandera Z, indicando si lo que ingresó al registro es cero o no.
RES_PUERTOS:MOVF  PORTC, W ;W = PORTC

       ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
       ;registro de RAM menos lo que haya en el acumulador W, la resta se
       ;lleva a cabo haciendo una suma entre el registro F y el
       ;complemento A2 de lo que haya en el acumulador W, el resultado de
       ;la resta se guardará en el registro F o en el acumulador W
       ;dependiendo de cuál se ponga en la posición de D.
       SUBWF PORTB, W ;W = PORTB - W = PORTB - PORTC = PORTB + CA2(PORTC)
       ;Podemos saber el signo del resultado dependiendo del estado de la
       ;bandera C (acarreo):
       ;Si C = 0 el resultado de la resta fue negativo.
       ;Si C = 1 el resultado de la resta fue positivo.

       ;La instrucción afecta las banderas C de acarreo, DC de acarreo
       ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
       ;C = Acerreo: Se pone como 1 lógico cuando al final de una
       ;operación matemática sobró un 1, también indica el signo del
       ;resultado después de efectuar una operación (se levanta la bandera
       ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
       ;se pone en letra mayúscula, las banderas las podemos ver en la
       ;parte superior del programa, a la derecha del contador de programa
       ;o PC).

       ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
       ;realizarse una operación matemática pasó un 1 lógico de los 4
       ;primeros bits (los de la derecha) del número binario de 8 bits
       ;a los segundos 4 bits del número binario (los de la izquierda).
       ;Se levanta la bandera poniéndose en letra mayúscula al simular el
       ;código.

       ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
       ;operación matemática el resultado es completamente cero. Se
       ;levanta la bandera poniéndose en letra mayúscula al simular el
       ;código.

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
```

```
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.
   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, C ;Checar la bandera C del registro STATUS
    ;Si C = 0 el resultado de la resta fue negativo
       ;Sigue la ejecución normal del código
       ;Saca el complemento A2 del resultado para obtener su magnitud,
       ;esto se hace restando 0 hexadecimal menos el resultado negativo.

    ;Si C = 1 el resultado de la resta fue positivo
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;Saca el resultado de la resta directamente.

       ;SUBLW  K (C, DC, Z): Resta el valor literal K de 8 bits menos lo
       ;que haya en el acumulador W y el resultado de la resta lo guarda de
       ;nuevo en el acumulador W, los números hexadecimales se declaran
       ;poniendo 0Xnúmero_hexadecimal.
       ;PARA SACAR EL COMPLEMENTO A2 DE UN NÚMERO NEGATIVO Y OBTENER SU
       ;MAGNITUD DEBEMOS RESTAR: MAGNITUD = 0 - NÚMERO_NEGATIVO
       SUBLW 0X00
       ;W = 0X00 - W = 0X00 - (PORTB - PORTC) = CA2(PORTB - PORTC)

       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTD ;PORTD = W = PORTB - PORTC

       ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
       ;código, indicado por un signo de pesos seguido de un signo menos y
       ;el número de instrucciones hacia atrás que quiero que brinque el
       ;programa o por una directiva EQU que tenga el nombre de la parte
       ;del código a donde quiero que brinque el programa que se debe poner
       ;en el lugar del valor literal k.
       GOTO  RES_PUERTOS
       ;Todos los códigos en ensamblador deben tener una instrucción GOTO
       ;para que el PIC repita su función indefinidamente.

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

6.-Igualdad o Desigualdad entre 2 Puertos:

6.- Lea el puerto B (PORTB) y el puerto C (PORTC):

Si PORTB = PORTC se debe sacar 0XFD por el puerto D (PORTD).

Si PORTB ≠ PORTC se debe sacar 0X24 por el puerto D (PORTD).

Código Ensamblador:

```
;6.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB ? PORTC se debe sacar 0X24 por el puerto D (PORTD).
;Si PORTB = PORTC se debe sacar 0XFD por el puerto D (PORTD).

;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:
;6.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB ? PORTC se debe sacar 0X24 por el puerto D (PORTD).
;Si PORTB = PORTC se debe sacar 0XFD por el puerto D (PORTD).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada y el D es de salida, por lo que
;solo debo cambiar al registro TRIS asociado al puerto D:
        ;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.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;Puedo ver si dos valores son iguales restándolos entre sí (no importando el
;órden de la resta) y checando si su resultado es cero o no, puedo checar si es
;cero o no con la bandera Z = ceros:
    ;Si Z = 0 el resultado de la resta no es cero:
        ;No son iguales los números binarios que ingresaron por los puertos.

    ;Si Z = 1 el resultado de la resta es cero:
        ;SON IGUALES LOS NÚMEROS BINARIOS QUE INGRESARON POR LOS PUERTOS.
;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
        ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
        ;bandera Z, indicando si lo que ingresó al registro es cero o no.
IGUALDAD:   MOVF   PORTC, W ;W = PORTC

        ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
```

```
        ;registro de RAM menos lo que haya en el acumulador W, la resta se
        ;lleva a cabo haciendo una suma entre el registro F y el
        ;complemento A2 de lo que haya en el acumulador W, el resultado de
        ;la resta se guardará en el registro F o en el acumulador W
        ;dependiendo de cuál se ponga en la posición de D.
        SUBWF PORTB, W ;W = PORTB - W = PORTB - PORTC = PORTB + CA2(PORTC)
        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
        ;parte superior del programa, a la derecha del contador de programa
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
        ;Se levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
        ;Sigue la ejecución normal del código.
        ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
        ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
        ;en 1 al contador de programa o PC).
        ;Tarda 2 ciclos de máquina en ejecutarse.
   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, Z ;Checar la bandera Z del registro STATUS
    ;Si Z = 0 el resultado de la resta no es cero:
        ;No son iguales los números binarios que ingresaron por los puertos.
        ;Sigue la ejecución normal del código.

    ;Si Z = 1 el resultado de la resta es cero:
        ;SON IGUALES LOS NÚMEROS BINARIOS QUE INGRESARON POR LOS PUERTOS.
        ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.

        ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
        ;código, indicado por un signo de pesos seguido de un signo menos y
        ;el número de instrucciones hacia atrás que quiero que brinque el
        ;programa o por una directiva EQU que tenga el nombre de la parte
        ;del código a donde quiero que brinque el programa que se debe poner
        ;en el lugar del valor literal k.
        GOTO  NO_IGUAL
        ;Si Z = 0 se ejecuta este GOTO, sino se lo brinca.
        ;Hace que el programa brinque a la instrucción NO_IGUAL que carga el
        ;valor 0X24 en el acumulador W cuando PORTB es diferente a PORTC
```

```
        ;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 0XFD ;Si Z = 1, osea PORTB = PORTC, W = 0XFD

        ;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  RESULTADO
        ;Si Z = 0, se ejecutó GOTO  NO_IGUAL y el programa se habrá brincado
        ;esta instrucción.
        ;Si Z = 1, antes se cargó W = 0XFD y se ejecuta este GOTO para
        ;brincar a la instrucción RESULTADO que carga el valor del
        ;acumulador W en el puerto D, PORTD = W = 0XFD.

        ;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.
NO_IGUAL:   MOVLW 0X24 ;Si Z = 0, osea cuando PORTB ? PORTC, W = 0X24
        ;Si Z = 0, antes se brincó por medio de un GOTO a esta instrucción y
        ;la siguiente instrucción a ejecutarse lo que hace es cargar el
        ;acumulador W al puerto D, PORTD = W = 0X24.
        ;Si Z = 1, se ejecutó GOTO  RESULTADO y el programa se habrá
        ;brincado esta instrucción y llegado a la siguiente.

        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
RESULTADO:  MOVWF PORTD
        ;Si Z = 0, PORTB ? PORTC, PORTD = W = 0X24
        ;Si Z = 1, PORTB = PORTC, PORTD = W = 0XFD

        ;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  IGUALDAD
        ;Todos los códigos en ensamblador deben tener una instrucción GOTO
        ;hasta el final (antes de la instrucción END) para que el PIC repita
        ;su función indefinidamente.

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

7.-Valor Mayor/Igual o Menor entre 2 Puertos:

7.- Lea el puerto B (PORTB) y el puerto C (PORTC):

Si PORTB ≥ PORTC se debe sacar 0XAB por el puerto D (PORTD).

Si PORTB < PORTC se debe sacar 0XC5 por el puerto D (PORTD).

Código Ensamblador:

```
;7.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB >= PORTC se debe sacar 0XAB por el puerto D (PORTD).
;Si PORTB < PORTC se debe sacar 0XC5 por el puerto D (PORTD).

;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:
;7.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB >= PORTC se debe sacar 0XAB por el puerto D (PORTD).
;Si PORTB < PORTC se debe sacar 0XC5 por el puerto D (PORTD).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada y el D es de salida, por lo que
;solo debo cambiar al registro TRIS asociado al puerto D:
        ;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.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;Puedo ver si dos valores son mayores o menores entre sí restándolos, en este
;tipo de comparación sí importa el órden de la resta ya que será mayor o igual
;el primer número binario ingresado si el resultado es positivo y menor si el
;resultado es negativo, puedo checar si es positivo o negativo el resultado con
;la bandera C = acarreo:
    ;Si C = 0 el resultado de la resta fue negativo.
        ;El primer dato ingresado ES MENOR al segundo dato ingresado.

    ;Si C = 1 el resultado de la resta fue positivo.
        ;El primer dato ingresado ES MAYOR O IGUAL al segundo dato ingresado.

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
        ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
        ;bandera Z, indicando si lo que ingresó al registro es cero o no.
COMPARA:    MOVF   PORTC, W ;W = PORTC

        ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
        ;registro de RAM menos lo que haya en el acumulador W, la resta se
        ;lleva a cabo haciendo una suma entre el registro F y el
        ;complemento A2 de lo que haya en el acumulador W, el resultado de
        ;la resta se guardará en el registro F o en el acumulador W
        ;dependiendo de cuál se ponga en la posición de D.
        SUBWF PORTB, W ;W = PORTB - W = PORTB - PORTC = PORTB + CA2(PORTC)
        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
        ;parte superior del programa, a la derecha del contador de programa
```

```
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
        ;Se levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

;ANTES DE USAR UN CONDICIONAL VAMOS A ASIGNAR UN VALOR AL ACUMULADOR W QUE
;CORRESPONDE A CUANDO PORTB >= PORTC, YA QUE SI EL ACARREO ES 1, EL CONDICIONAL
;BTFSS HARÁ QUE EL PROGRAMA SE SALTE LA SIGUIENTE INSTRUCCIÓN QUE CORRESPONDERÁ
;A CUANDO PORTB < PORTC Y EL VALOR ACTUAL SE MANTENDRÁ, SI EL ACARREO ES 0, LA
;SIGUIENTE INSTRUCCIÓN DE CÓDIGO QUE VA DESPUÉS DE BTFSS SI SE EJECUTARÁ Y
;REESCRIBIRÁ EL VALOR GUARDADO AHORITA EN EL ACUMULADOR W POR EL QUE CORRESPONDE
;CUANDO PORTB < PORTC.
        ;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.
MAYOR:      MOVLW 0XAB ;W = 0XAB
        ;La directiva EQU de la línea de código no está conectada a nignuna
        ;instrucción GOTO, solamente es informativa.
        ;Si C = 1, osea que el resultado de PORTB - PORTC fue positivo,
        ;PORTB >= PORTC, por lo tanto W = 0XAB

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
        ;Sigue la ejecución normal del código.
        ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
        ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
        ;en 1 al contador de programa o PC).
        ;Tarda 2 ciclos de máquina en ejecutarse.

   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, C ;Checar la bandera C del registro STATUS
    ;Si C = 0 el resultado de la resta fue negativo
        ;Sigue la ejecución normal del código
        ;El primer dato ingresado a la comparación es menor, PORTB < PORTC.

    ;Si C = 1 el resultado de la resta fue positivo
        ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
        ;EL PRIMER DATO INGRESADO A LA COMPARACIÓN ES MAYOR O IGUAL, PORTB >= PORTC.

;Si C = 1, osea que PORTB ? PORTC esta instrucción de código se la salta el
;programa por el condicional BTFSS y en el acumulador se queda el valor W = 0XAB
;Sino la siguiente insrrucción del código sobreescribe ese valor.
        ;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.
MENOR:      MOVLW 0XC5 ;W = 0XC5
```

```
;La directiva EQU de la línea de código no está conectada a nignuna
;instrucción GOTO, solamente es informativa.
;Si C = 0, osea que el resultado de PORTB - PORTC fue negativo,
;PORTB < PORTC, por lo tanto W = 0XC5

;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F.
MOVWF PORTD
;Si C = 0, PORTB < PORTC, PORTD = W = 0XC5
;Si C = 1, PORTB >= PORTC, PORTD = W = 0XAB

;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
;código, indicado por un signo de pesos seguido de un signo menos y
;el número de instrucciones hacia atrás que quiero que brinque el
;programa o por una directiva EQU que tenga el nombre de la parte
;del código a donde quiero que brinque el programa que se debe poner
;en el lugar del valor literal k.
GOTO  COMPARA
;Todos los códigos en ensamblador deben tener una instrucción GOTO
;hasta el final (antes de la instrucción END) para que el PIC repita
;su función indefinidamente.

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

8.-Valor Menor/Igual o Mayor entre 2 Puertos:

8.- Lea el puerto B (PORTB) y el puerto C (PORTC):

Si PORTB ≤ PORTC se debe sacar 0XDF por el puerto D (PORTD) y 0X8F por el puerto A (PORTA).

Si PORTB > PORTC se debe sacar 0X34 por el puerto D (PORTD) y 0X6A por el puerto A (PORTA).

Código Ensamblador:

```
;8.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB ? PORTC se debe sacar 0XDF por el puerto D (PORTD) y 0X8F por el
;puerto A (PORTA).
;Si PORTB > PORTC se debe sacar 0X34 por el puerto D (PORTD) y 0X6A por el
;puerto A (PORTA).

;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:
;8.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB ? PORTC se debe sacar 0XDF por el puerto D (PORTD) y 0X8F por el
;puerto A (PORTA).
;Si PORTB > PORTC se debe sacar 0X34 por el puerto D (PORTD) y 0X6A por el
;puerto A (PORTA).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
```

```
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada, el A y D son de salida, por lo
;que solo debo cambiar al registro TRIS asociado al puerto A y D:
        ;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  TRISA ;Hace todos los pines del puerto A sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;Puedo ver si dos valores son mayores o menores entre sí restándolos, en este
;tipo de comparación sí importa el órden de la resta ya que será mayor o igual
;el primer número binario ingresado si el resultado es positivo y menor si el
;resultado es negativo, puedo checar si es positivo o negativo el resultado con
;la bandera C = acarreo:
    ;Si C = 0 el resultado de la resta fue negativo.
        ;El primer dato ingresado ES MENOR al segundo dato ingresado.

    ;Si C = 1 el resultado de la resta fue positivo.
        ;El primer dato ingresado ES MAYOR O IGUAL al segundo dato ingresado.

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
        ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
        ;bandera Z, indicando si lo que ingresó al registro es cero o no.
COMPARA:    MOVF   PORTC, W ;W = PORTC

        ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
        ;registro de RAM menos lo que haya en el acumulador W, la resta se
        ;lleva a cabo haciendo una suma entre el registro F y el
        ;complemento A2 de lo que haya en el acumulador W, el resultado de
        ;la resta se guardará en el registro F o en el acumulador W
        ;dependiendo de cuál se ponga en la posición de D.
        SUBWF PORTB, W ;W = PORTB - W = PORTB - PORTC = PORTB + CA2(PORTC)
        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
        ;parte superior del programa, a la derecha del contador de programa
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
        ;Se levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.
```

```
;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.

   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, C ;Checar la bandera C del registro STATUS
    ;Si C = 0 el resultado de la resta fue negativo
       ;Sigue la ejecución normal del código
       ;El primer dato ingresado a la comparación es menor, PORTB < PORTC.

    ;Si C = 1 el resultado de la resta fue positivo
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;EL PRIMER DATO INGRESADO A LA COMPARACIÓN ES MAYOR O IGUAL, PORTB >= PORTC.

;Si el resultado fue negativo, PORTB < PORTC usamos una instrucción GOTO para
;que el programe brinque a la parte del código donde se ejecutan acciones que
;corresponden esa condición, sino esta instrucción se la brincará el programa.
       ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
       ;código, indicado por un signo de pesos seguido de un signo menos y
       ;el número de instrucciones hacia atrás que quiero que brinque el
       ;programa o por una directiva EQU que tenga el nombre de la parte
       ;del código a donde quiero que brinque el programa que se debe poner
       ;en el lugar del valor literal k.
       GOTO  MENOR
       ;Si C = 0 se ejecuta este GOTO, si C = 1 se lo brinca.

;Si C = 1, osea que PORTB ? PORTC, PORTD = 0XDF y PORTA = 0X8F:
       ;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.
MAYOR:      MOVLW 0XDF ;W = 0XDF
       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTD ;PORTD = W = 0XDF
       ;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 0XC5 ;W = 0XC5
       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTA ;PORTA = W = 0XC5
       ;Si C = 1, PORTB >= PORTC, PORTD = W = 0XDF
       ;Si C = 1, PORTB >= PORTC, PORTA = W = 0X8F
       ;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  COMPARA
       ;Con esto se termina la funcionalidad del código para cuando se
       ;cumplió la condición PORTB >= PORTC y el GOTO hace que el PIC
       ;vuelva a buscar en los pines de los puertos B y C para ver si esta
```

```
        ;condición se sigue cumpliendo, sino ejecutará lo de abajo.

;Si C = 0, osea que PORTB < PORTC, la instrucción GOTO manda al programa a esta
;parte donde se indica que PORTD = 0X34 y PORTA = 0X6A:
        ;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.
MENOR:      MOVLW 0X34 ;W = 0X34
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0X34
        ;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 0X6A ;W = 0X6A
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTA ;PORTA = W = 0X6A
        ;Si C = 1, PORTB >= PORTC, PORTD = W = 0X34
        ;Si C = 1, PORTB >= PORTC, PORTA = W = 0X6A
        ;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  COMPARA
        ;Todos los códigos en ensamblador deben tener al menos una
        ;instrucción GOTO cuya función sea ocasionar que el PIC repita su
        ;función indefinidamente, aunque en este caso pudimos ver que
        ;existieron 2 que realizan esta función.

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

9.-Valor Menor, Mayor o Igual entre 2 Puertos:

9.- Lea el puerto B (PORTB) y el puerto C (PORTC):

Si PORTB > PORTC se debe sacar 0X12 por el puerto D (PORTD) y 0XCC por el puerto A (PORTA).

Si PORTB < PORTC se debe sacar 0X69 por el puerto D (PORTD) y 0X7B por el puerto A (PORTA).

Si PORTB = PORTC se debe sacar 0XE1 por el puerto D (PORTD) y 0XDD por el puerto A (PORTA).

Código Ensamblador:

```
;9.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB > PORTC se debe sacar 0X12 por el puerto D (PORTD) y 0XCC por el
;puerto A (PORTA).
;Si PORTB < PORTC se debe sacar 0X69 por el puerto D (PORTD) y 0X7B por el
;puerto A (PORTA).
;Si PORTB = PORTC se debe sacar 0XE1 por el puerto D (PORTD) y 0XDD por el
;puerto A (PORTA).

;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:
;9.- Lea el puerto B (PORTB) y el puerto C (PORTC):
;Si PORTB > PORTC se debe sacar 0X12 por el puerto D (PORTD) y 0XCC por el
;puerto A (PORTA).
;Si PORTB < PORTC se debe sacar 0X69 por el puerto D (PORTD) y 0X7B por el
;puerto A (PORTA).
;Si PORTB = PORTC se debe sacar 0XE1 por el puerto D (PORTD) y 0XDD por el
;puerto A (PORTA).

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada, el A y D son de salida, por lo
;que solo debo cambiar al registro TRIS asociado al puerto A y D:
        ;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  TRISA ;Hace todos los pines del puerto A sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;Puedo ver si dos valores son mayores o menores entre sí restándolos, en este
;tipo de comparación sí importa el órden de la resta ya que será mayor o igual
;el primer número binario ingresado si el resultado es positivo y menor si el
;resultado es negativo, puedo checar si es positivo o negativo el resultado con
;la bandera C = acarreo:
    ;Si C = 0 el resultado de la resta fue negativo.
       ;El primer dato ingresado ES MENOR al segundo dato ingresado.

    ;Si C = 1 el resultado de la resta fue positivo.
       ;El primer dato ingresado ES MAYOR O IGUAL al segundo dato ingresado.

;Para efectuar una resta, primero debo haber cargado el número que quiero restar
;al acumulador W y luego usar la instrucción SUBWF.
        ;MOVF  F,D(Z): 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 lo que pongamos como D. Afecta la
        ;bandera Z, indicando si lo que ingresó al registro es cero o no.
COMPARA:   MOVF   PORTC, W ;W = PORTC

        ;SUBWF   F, D (C, DC, Z): Resta lo que haya en la dirección F del
        ;registro de RAM menos lo que haya en el acumulador W, la resta se
        ;lleva a cabo haciendo una suma entre el registro F y el
        ;complemento A2 de lo que haya en el acumulador W, el resultado de
        ;la resta se guardará en el registro F o en el acumulador W
        ;dependiendo de cuál se ponga en la posición de D.
        SUBWF PORTB, W ;W = PORTB - W = PORTB - PORTC = PORTB + CA2(PORTC)
        ;La instrucción afecta las banderas C de acarreo, DC de acarreo
        ;decimal y Z de ceros que son los bit 0, 1 y 2 del registro STATUS:
        ;C = Acerreo: Se pone como 1 lógico cuando al final de una
        ;operación matemática sobró un 1, también indica el signo del
        ;resultado después de efectuar una operación (se levanta la bandera
        ;cuando tiene un 1 lógico y en el programa lo podemos notar porque
        ;se pone en letra mayúscula, las banderas las podemos ver en la
```

```
        ;parte superior del programa, a la derecha del contador de programa
        ;o PC).

        ;DC = Acerreo Decimal: Se pone como 1 lógico cuando al
        ;realizarse una operación matemática pasó un 1 lógico de los 4
        ;primeros bits (los de la derecha) del número binario de 8 bits
        ;a los segundos 4 bits del número binario (los de la izquierda).
        ;Se levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

        ;Z = Ceros: Se pone como 1 lógico cuando al realizarse una
        ;operación matemática el resultado es completamente cero. Se
        ;levanta la bandera poniéndose en letra mayúscula al simular el
        ;código.

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.

   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;la bandera de acarreo C para saber si el resultado de la resta fue negativo
   ;o positivo, siempre que querramos ver el estado de una bandera debemos
   ;acceder al registro de propósito específico STATUS, por lo que en vez de
   ;poner un número hexadecimal para indicar el registro F, pondré su directiva
   ;EQU (osea el nombre del registro) y en vez de poner un número decimal
   ;como B para indicar el bit que quiero alcanzar, pondré su directiva EQU
   ;(osea el nombre de la bandera).
       BTFSS STATUS, C ;Checa la bandera C del registro STATUS
    ;Si C = 0 el resultado de la resta fue negativo
       ;Sigue la ejecución normal del código
       ;El primer dato ingresado a la comparación es menor, PORTB < PORTC.

    ;Si C = 1 el resultado de la resta fue positivo
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;EL PRIMER DATO INGRESADO A LA COMPARACIÓN ES MAYOR O IGUAL, PORTB >= PORTC.

;Si el resultado fue negativo, C = 0, es porque PORTB < PORTC y usaremos una
;instrucción GOTO para que el programe brinque a la parte del código donde se
;ejecutan acciones que corresponden esa condición, sino esta instrucción se la
;brincará el programa cuando C = 1.
       ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
       ;código, indicado por un signo de pesos seguido de un signo menos y
       ;el número de instrucciones hacia atrás que quiero que brinque el
       ;programa o por una directiva EQU que tenga el nombre de la parte
       ;del código a donde quiero que brinque el programa que se debe poner
       ;en el lugar del valor literal k.
       GOTO  MENOR
       ;Si C = 0 se ejecuta este GOTO, si C = 1 se lo brinca.

   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
```

```
        ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
        ;en 1 al contador de programa o PC).
        ;Tarda 2 ciclos de máquina en ejecutarse.

        BTFSS STATUS, Z ;Checa la bandera Z del registro STATUS
    ;Si Z = 0 el resultado de la resta no es cero:
        ;No son iguales los números binarios que ingresaron por los puertos,
        ;osea que uno es mayor que el otro, porque si hubiera sido menor, el
        ;GOTO anterior se hubiera saltado esta línea de código.
        ;Sigue la ejecución normal del código y se ejecuta el siguiente GOTO.

    ;Si Z = 1 el resultado de la resta es cero:
        ;SON IGUALES LOS NÚMEROS BINARIOS QUE INGRESARON POR LOS PUERTOS.
        ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.

;Si el resultado fue diferente a cero, Z = 0, es porque PORTB >= PORTC, sabemos
;esto porque en la condición abterior se evaluó la bandera C y vimos si
;PORTB < PORTC, si se hubiera cumplido esa condición esta línea de código se la
;hubiera saltado el programa, por lo que ahora usaremos una instrucción GOTO
;para que el programe brinque a la parte del código donde se ejecutan acciones
;que corresponden a cuando PORTB >= PORTC. Esta instrucción se la brincará el
;programa cuando Z = 1.
        ;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  MAYOR
        ;Si Z = 0 se ejecuta este GOTO, sino se lo brinca.

;Si Z = 1, osea que PORTB = PORTC, PORTD = 0XDF y PORTA = 0X8F:
        ;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.
IGUAL:      MOVLW 0XE1 ;W = 0XE1
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0XE1
        ;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 0XDD ;W = 0XDD
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTA ;PORTA = W = 0XDD
        ;Si C = 1, PORTB >= PORTC, PORTD = W = 0X12
        ;Si C = 1, PORTB >= PORTC, PORTA = W = 0XCC
        ;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  COMPARA
        ;Con esto se termina la funcionalidad del código para cuando se
        ;cumplió la condición PORTB = PORTC y el GOTO hace que el PIC
        ;vuelva a buscar en los pines de los puertos B y C para ver si esta
        ;condición se sigue cumpliendo.

;Si Z = 0 y C = 1, osea que PORTB > PORTC, PORTD = 0XDF y PORTA = 0X8F:
        ;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.
MAYOR:      MOVLW 0X12 ;W = 0X12
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0X12
        ;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 0XCC ;W = 0XCC
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTA ;PORTA = W = 0XCC
        ;Si C = 1, PORTB >= PORTC, PORTD = W = 0X12
        ;Si C = 1, PORTB >= PORTC, PORTA = W = 0XCC
```

```
        ;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  COMPARA
        ;Con esto se termina la funcionalidad del código para cuando se
        ;cumplió la condición PORTB >= PORTC y el GOTO hace que el PIC
        ;vuelva a buscar en los pines de los puertos B y C para ver si esta
        ;condición se sigue cumpliendo.

;Si C = 0, osea que PORTB < PORTC, la instrucción GOTO manda al programa a esta
;parte donde se indica que PORTD = 0X34 y PORTA = 0X6A:
        ;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.
MENOR:      MOVLW 0X69 ;W = 0X69
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0X69
        ;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 0X7B ;W = 0X7B
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTA ;PORTA = W = 0X7B
        ;Si C = 1, PORTB >= PORTC, PORTD = W = 0X69
        ;Si C = 1, PORTB >= PORTC, PORTA = W = 0X7B
        ;GOTO: Sirve para hacer que el programa brinque a otra parte del
        ;código, indicado por un signo de pesos seguido de un signo menos y
        ;el número de instrucciones hacia atrás que quiero que brinque el
        ;programa o por una directiva EQU que tenga el nombre de la parte
        ;del código a donde quiero que brinque el programa.
        GOTO  COMPARA
        ;Todos los códigos en ensamblador deben tener al menos una
        ;instrucción GOTO cuya función sea ocasionar que el PIC repita su
        ;función indefinidamente, aunque en este caso pudimos ver que
        ;existieron 3 que realizan esta función.

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

10.-Intercambio de Nibbles Bajo por Alto:

10.- Lea el nibble bajo del puerto C (PORTC) y saque ese valor como el nibble alto del puerto D (PORTD), además lea el nibble alto del puerto B (PORTB) y saque ese valor como el nibble bajo del puerto D (PORTD).

Código Ensamblador:

```
        INCLUDE    <C:\Users\diego\OneDrive\Documents\Aprendiendo\MPLAB (Ensamblador)\1.-
        Ejercicios PIC16F887\Configuracion_PIC.asm>
        CLRF    TRISD
        BCF     STATUS,RP0        ;B0
CICLO:  MOVLW       0X0F
        ANDWF       PORTC,W
        MOVWF       0X20
        MOVLW       0XF0
        ANDWF       PORTB,W
        ADDWF       0X20,F
        SWAPF       0X20,W
        MOVWF       PORTD
        GOTO    CICLO
        END
```

11.-Lectura de Pin y Sacar su Valor por un Puerto:

11.- Lea el pin 0 del puerto B (PORTB.0):

Si el PORTB.0 = 0, el PORTD debe sacar 0XF5.

Si el PORTB.0 = 1, el PORTD debe sacar 0X24.

Código Ensamblador:

```
;11.- Lea el pin 0 del puerto B (PORTB.0):
;Si el PORTB.0 = 0, el PORTD debe sacar 0XF5.
;Si el PORTB.0 = 1, el PORTD debe sacar 0X24.

;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
;pines de los puertos A, B y E (que son los únicos que pueden ser analógicos o
;digitales) sean todos digitales.

;Ahora si ya puedo resolver el ejercicio:
;11.- Lea el pin 0 del puerto B (PORTB.0):
;Si el PORTB.0 = 0, el PORTD debe sacar 0XF5.
;Si el PORTB.0 = 1, el PORTD debe sacar 0X24.

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada, el A y D son de salida, por lo
;que solo debo cambiar al registro TRIS asociado al puerto A y D:
        ;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.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;ANTES DE USAR UN CONDICIONAL VAMOS A ASIGNAR UN VALOR AL ACUMULADOR W QUE
;CORRESPONDE A 0X24 CUANDO PORTB.0 = 1, YA QUE SI PORTB.0 = 0, OSEA QUE EL PIN 0
;DEL PUERTO B ES 0, LA INSTRUCCIÓN DE CÓDIGO QUE VA DESPUÉS DEL CONDICIONAL
;BTFSS SI SE EJECUTARÁ Y REESCRIBIRÁ EL VALOR GUARDADO AHORITA EN EL ACUMULADOR
;W POR EL QUE CORRESPONDE CUANDO PORTB.0 = 0 que es 0XF5.
        ;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.
BOTON:      MOVLW 0X24 ;W = 0X24

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.

   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;uno de los pines del puerto B, por lo que debemos acceder al registro de
   ;propósito específico por su nombre o directiva EQU, osea PORTB, en vez de
   ;poner un número hexadecimal para indicar el registro F y pondré un número
   ;decimal como B para indicar el PIN que quiero alcanzar del puerto.
       BTFSS PORTB, 0 ;Checa el valor del PIN 0 del Puerto B
    ;Si PORTB.0 = 0:
       ;Sigue la ejecución normal del código
       ;Reescribe el valor de W por W = 0XF5

    ;Si PORTB.0 = 1:
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;MANTIENE EL VALOR DEL ACUMULADOR W, W = 0X24

;SOBREESCRIBIR EL VALOR DEL ACUMULADOR CUANDO PORTB.0 = 0
        ;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 0XF5 ;W = 0XF5

;ASIGNA EL VALOR DEL ACUMULADOR A TODOS LOS PINES DEL PUERTO D
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD
        ;Si PORTB.0 = 0, PORTD = W = 0XF5.
        ;Si PORTB.0 = 1, PORTD = W = 0X24.

        ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
        ;código, indicado por un signo de pesos seguido de un signo menos y
        ;el número de instrucciones hacia atrás que quiero que brinque el
        ;programa o por una directiva EQU que tenga el nombre de la parte
        ;del código a donde quiero que brinque el programa que se debe poner
        ;en el lugar del valor literal k.
        GOTO  BOTON
        ;Todos los códigos en ensamblador deben tener una instrucción GOTO
        ;hasta el final (antes de la instrucción END) para que el PIC repita
        ;su función indefinidamente.

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

12.-Lectura de dos Pines y Sacar su Valor por un Puerto:

12.- Lea el pin 0 del puerto A (PORTA.0) y el pin 3 del puerto C (PORTC.3):

Si el PORTA.0 = 0 y el PORTC.3 = 0, el PORTD debe sacar 0XF3 y el PORTB debe sacar 0X89.

Si el PORTA.0 = 0 y el PORTC.3 = 1, el PORTD debe sacar 0X45 y el PORTB debe sacar 0X7C.

Si el PORTA.0 = 1 y el PORTC.3 = 0, el PORTD debe sacar 0X94 y el PORTB debe sacar 0X42.

Si el PORTA.0 = 1 y el PORTC.3 = 1, el PORTD debe sacar 0XFF y el PORTB debe sacar 0XDE.

Código Ensamblador:

```
;12.- Lea el pin 0 del puerto A (PORTA.0) y el pin 3 del puerto C (PORTC.3):
;Si el PORTA.0 = 0 y el PORTC.3 = 0, el PORTD debe sacar 0XF3 y el PORTB debe sacar 0X89.
;Si el PORTA.0 = 0 y el PORTC.3 = 1, el PORTD debe sacar 0X45 y el PORTB debe sacar 0X7C.
;Si el PORTA.0 = 1 y el PORTC.3 = 0, el PORTD debe sacar 0X94 y el PORTB debe sacar 0X42.
;Si el PORTA.0 = 1 y el PORTC.3 = 1, el PORTD debe sacar 0XFF y el PORTB debe sacar 0XDE.

;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:
;12.- Lea el pin 0 del puerto A (PORTA.0) y el pin 3 del puerto C (PORTC.3):
;Si el PORTA.0 = 0 y el PORTC.3 = 0, el PORTD debe sacar 0XF3 y el PORTB debe sacar 0X89.
;Si el PORTA.0 = 0 y el PORTC.3 = 1, el PORTD debe sacar 0X45 y el PORTB debe sacar 0X7C.
;Si el PORTA.0 = 1 y el PORTC.3 = 0, el PORTD debe sacar 0X94 y el PORTB debe sacar 0X42.
;Si el PORTA.0 = 1 y el PORTC.3 = 1, el PORTD debe sacar 0XFF y el PORTB debe sacar 0XDE.

;PUERTOS COMO ENTRADAS O SALIDAS
;Para indicar si los distintos pines de los puertos son entradas o salidas debo
;cambiar los bits de los registros TRISA, TRISB, TRISC, TRISD y TRISE, indicando
;de la siguiente manera si son entradas o salidas:
        ;Bit del registro TRIS: 1 = Entrada (Input)
        ;Bit del registro TRIS: 0 = Salida (Output)
;De esta manera se indica para cada registro TRIS, que está asociado a cada
;puerto A, B, C, D o E si su pin es de entrada o salida.
;Por default todos los bits de los registros TRIS después de un reset se ponen
;en 1 lógico, osea que son entradas, por lo que solo debo cambiar y poner en 0
;los bits asociados a los pines de los puertos que quiero que se vuelvan salidas.

;En este caso los puertos B y C son de entrada, el A y D son de salida, por lo
;que solo debo cambiar al registro TRIS asociado al puerto A y D:
        ;CLRF   F (Z): Llena de ceros la dirección F del registro RAM
        ;indicado. Siempre levanta la bandera ceros, Z = 1.
        CLRF  TRISB ;Hace todos los pines del puerto B sean salidas.
        CLRF  TRISD ;Hace todos los pines del puerto D sean salidas.

;LUEGO ME DEBO REGRESAR AL BANCO 0 PARA QUE PUEDA MANIPULAR LOS PUERTOS
        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

;AHORA VAMOS A USAR UNA DE LAS 35 OPERACIONES QUE ME PERMITEN REALIZAR
;CONDICIONALES PARECIDOS A UN IF.
;NO CONFUNDIR BTFSS (BRINCA CON B = 1) CON BTFSC (BRINCA CON B = 0).
   ;BTFSS   F, B: Esta operación es lo más parecido a un condicional if que
```

```
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 1 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.

   ;La posición del bit B en el registro F se puede indicar poniendo el nombre
   ;del bit (osea su directiva EQU) o contando desde cero en decimal, osea
   ;poniendo 0, 1, 2, 3, 4, 5, 6 o 7. En este caso queremos checar el estado de
   ;uno de los pines del puerto A, por lo que debemos acceder al registro de
   ;propósito específico por su nombre o directiva EQU (osea PORTA) en vez de
   ;poner un número hexadecimal para indicar el registro F y pondré un número
   ;decimal como B para indicar el PIN que quiero alcanzar del puerto, en este
   ;como en la instrucción del ejercicio dice PORTA.0, pondré el número cero.
CHECA:      BTFSS PORTA, 0 ;Checa el valor del PIN 0 del Puerto A
    ;Si PORTA.0 = 0:
       ;Sigue la ejecución normal del código.
       ;Salta con un GOTO a una parte del código donde se analiza el valor
       ;de PORTC.3 ya sabiendo que PORTA.0 = 0, osea PA0.
        ;PORTA.0 = 0 y PORTC.3 = 0
           ;PORTD = 0XF3.
           ;PORTB = 0X89.
        ;PORTA.0 = 0 y PORTC.3 = 1
           ;PORTD = 0X45.
           ;PORTB = 0X7C.

    ;Si PORTA.0 = 1:
       ;SE SALTA LA SIGUIENTE LÍNEA DEL CÓDIGO.
       ;Salta con un GOTO a una parte del código donde se analiza el valor
       ;de PORTC.3 ya sabiendo que PORTA.0 = 1, osea PA1.
        ;PORTA.0 = 1 y PORTC.3 = 0
           ;PORTD = 0X94.
           ;PORTB = 0X42.
        ;PORTA.0 = 1 y PORTC.3 = 1
           ;PORTD = 0XFF.
           ;PORTB = 0XDE.

;CON INSTRUCCIONES GOTO PUEDO SALTAR A UNAS PARTES DEL CÓDIGO DONDE SE EVALÚA EL
;ESTADO DEL PIN 3 DEL PUERTO C YA SABIENDO EL VALOR DE PORTA.0:
       ;GOTO   k: Sirve para hacer que el programa brinque a otra parte del
       ;código, indicado por un signo de pesos seguido de un signo menos y
       ;el número de instrucciones hacia atrás que quiero que brinque el
       ;programa o por una directiva EQU que tenga el nombre de la parte
       ;del código a donde quiero que brinque el programa que se debe poner
       ;en el lugar del valor literal k.
       GOTO  PA0
       ;Si PORTA.0 = 0 se ejecuta este GOTO.
       ;Si PORTA.0 = 1 se lo brinca.

       GOTO  PA1
       ;Si PORTA.0 = 1 se ejecuta este GOTO porque se habrá brincado el
       ;anterior.

;YA SABIENDO QUE PORTA.0 = 0, PRIMERO CARGA EN EL ACUMULADOR W EL VALOR QUE
;PUEDE ADOPTAR EL PUERTO D SI PORTC.3 = 0, OSEA W = 0XF3, LUEGO EVALÚA EL VALOR
;DE PORTC.3 Y SI ES 0, SE BRINCARÁ EL GOTO PARA ASIGNAR W = 0X89 = PORTB, SI
;PORTC.3 ES 1, CON UN GOTO SE BRINCARÁ A LA PARTE PA0_PC1.
   ;PORTA.0 = 0 y PORTC.3 = 0
    ;PORTD = 0XF3.
    ;PORTB = 0X89.
       ;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.
PA0:    MOVLW 0XF3 ;W = 0XF3
   ;BTFSC   F, B: Esta operación es lo más parecido a un condicional if que
```

```
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 0 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
       ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
       ;en 1 al contador de programa o PC).
       ;Tarda 2 ciclos de máquina en ejecutarse.

    ;Si el bit B es uno (1):
       ;Sigue la ejecución normal del código.
       ;Tarda 1 ciclo de máquina en ejecutarse.
       BTFSC PORTC, 3 ;Checa el valor del PIN 3 del Puerto C

       ;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  PA0_PC1
       ;Si PORTC.3 = 0 se lo brinca al GOTO.
       ;Si PORTC.3 = 1 se ejecuta este GOTO.

       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTD ;PORTD = W = 0XF3
       ;MOVLW  K: Coloca directamente en el acumulador W un número binario
       ;cualquiera de 8 bits dado por el valor literal K indicado en forma
       ;hexadecimal, poniendo 0Xnúmero_hexadecimal.
       MOVLW 0X89 ;W = 0X89
       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTB ;PORTB = W = 0X89
       ;PORTA.0 = 0 y PORTC.3 = 0
        ;PORTD = 0XF3.
        ;PORTB = 0X89.
       ;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  CHECA
       ;Con esto se termina la funcionalidad del código para cuando se
       ;cumplió la condición PORTA.0 = 0 y PORTC.3 = 0, el GOTO hace que el
       ;PIC vuelva a buscar en los pines de los puertos B y C para ver si
       ;esta condición se sigue cumpliendo.

;CUANDO SE CUMPLIÓ EN LA PARTE ANTERIOR QUE PORTC.3 = 1, SE REESCRIBE EL VALOR
;QUE TENÍA CARGADO EL ACUMULADOR W PARA CARGARSE AL PUERTO D, SE ASIGNA UN
;NUEVO VALOR AL ACUMULADOR W Y ESTE NUEVO VALOR SE CARGA AL PUERTO B.
   ;PORTA.0 = 0 y PORTC.3 = 1
    ;PORTD = 0X45.
    ;PORTB = 0X7C.
       ;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.
PA0_PC1:    MOVLW 0X45 ;W = 0X45
       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTD ;PORTD = W = 0X45
       ;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 0X7C ;W = 0X7C
       ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
       ;registro de la RAM indicado por la dirección F.
       MOVWF PORTB ;PORTB = W = 0X7C
       ;PORTA.0 = 0 y PORTC.3 = 1
        ;PORTD = 0X45.
        ;PORTB = 0X7C.
       ;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  CHECA
       ;Con esto se termina la funcionalidad del código para cuando se
       ;cumplió la condición PORTA.0 = 0 y PORTC.3 = 1, el GOTO hace que el
```

```
        ;PIC vuelva a buscar en los pines de los puertos B y C para ver si
        ;esta condición se sigue cumpliendo.


;YA SABIENDO QUE PORTA.0 = 1, PRIMERO CARGA EN EL ACUMULADOR W EL VALOR QUE
;PUEDE ADOPTAR EL PUERTO D SI PORTC.3 = 0, OSEA W = 0X94, LUEGO EVALÚA EL VALOR
;DE PORTC.3 Y SI ES 0, SE BRINCARÁ EL GOTO PARA ASIGNAR W = 0X42 = PORTB, SI
;PORTC.3 ES 1, CON UN GOTO SE BRINCARÁ A LA PARTE PA1_PC1.
   ;PORTA.0 = 1 y PORTC.3 = 0
    ;PORTD = 0X94.
    ;PORTB = 0X42.
        ;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.
PA1:    MOVLW 0X94 ;W = 0X94
   ;BTFSC   F, B: Esta operación es lo más parecido a un condicional if que
   ;existe en el idioma ensamblador, su condición evalúa si el bit B del
   ;registro F es uno o cero y si es 0 se brinca la siguiente instrucción,
   ;sino sigue la ejecución normal:
    ;Si el bit B es cero (0):
        ;SE BRINCA LA SIGUIENTE INSTRUCCIÓN QUE HAYA EN EL CÓDIGO (aumenta
        ;en 1 al contador de programa o PC).
        ;Tarda 2 ciclos de máquina en ejecutarse.

    ;Si el bit B es uno (1):
        ;Sigue la ejecución normal del código.
        ;Tarda 1 ciclo de máquina en ejecutarse.
        BTFSC PORTC, 3 ;Checa el valor del PIN 3 del Puerto C

        ;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  PA1_PC1
        ;Si PORTC.3 = 0 se lo brinca al GOTO.
        ;Si PORTC.3 = 1 se ejecuta este GOTO.

        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0X94
        ;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 0X42 ;W = 0X42
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTB ;PORTB = W = 0X42
        ;PORTA.0 = 1 y PORTC.3 = 0
         ;PORTD = 0X94.
         ;PORTB = 0X42.
        ;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  CHECA
        ;Con esto se termina la funcionalidad del código para cuando se
        ;cumplió la condición PORTA.0 = 1 y PORTC.3 = 0, el GOTO hace que el
        ;PIC vuelva a buscar en los pines de los puertos B y C para ver si
        ;esta condición se sigue cumpliendo.

;CUANDO SE CUMPLIÓ EN LA PARTE ANTERIOR QUE PORTC.3 = 1, SE REESCRIBE EL VALOR
;QUE TENÍA CARGADO EL ACUMULADOR W PARA CARGARSE AL PUERTO D, SE ASIGNA UN
;NUEVO VALOR AL ACUMULADOR W Y ESTE NUEVO VALOR SE CARGA AL PUERTO B.
   ;PORTA.0 = 1 y PORTC.3 = 1
    ;PORTD = 0XFF.
    ;PORTB = 0XDE.
        ;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.
PA1_PC1:    MOVLW 0XFF ;W = 0XFF
        ;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
        ;registro de la RAM indicado por la dirección F.
        MOVWF PORTD ;PORTD = W = 0XFF
```

```
;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 0XDE ;W = 0XDE
;MOVWF   F: Lee el contenido del acumulador W y lo coloca en un
;registro de la RAM indicado por la dirección F.
MOVWF PORTB ;PORTB = W = 0XDE
;PORTA.0 = 0 y PORTC.3 = 1
 ;PORTD = 0XFF.
 ;PORTB = 0XDE.
;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  CHECA
;Con esto se termina la funcionalidad del código para cuando se
;cumplió la condición PORTA.1 = 0 y PORTC.3 = 1, el GOTO hace que el
;PIC vuelva a buscar en los pines de los puertos B y C para ver si
;esta condición se sigue cumpliendo.

;Todos los códigos en ensamblador deben tener al menos una
;instrucción GOTO cuya función sea ocasionar que el PIC repita su
;función indefinidamente, aunque en este caso pudimos ver que
;existieron 4 que realizan esta función.

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