

Microcontroladores

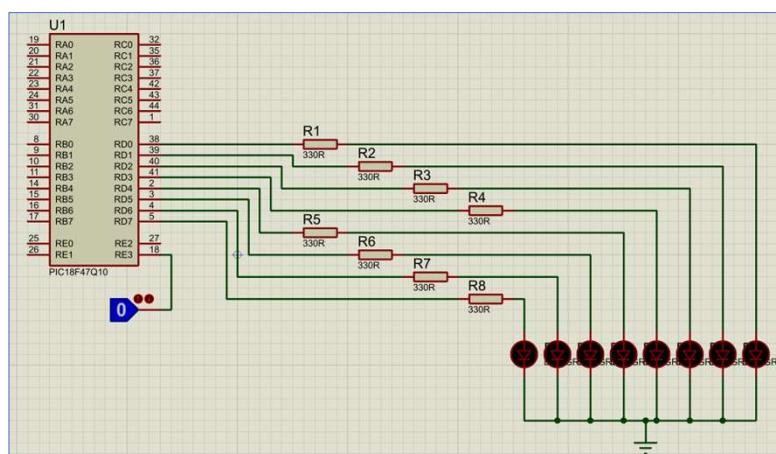
Semana 2

Semestre 2026-0

Profesor: Kalun José Lau Gan

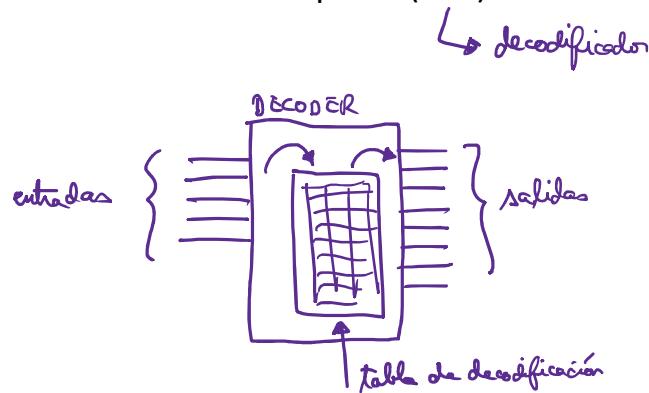
¿Cómo se resolvía el ejercicio del auto fantástico?

- Circuito en Proteus:



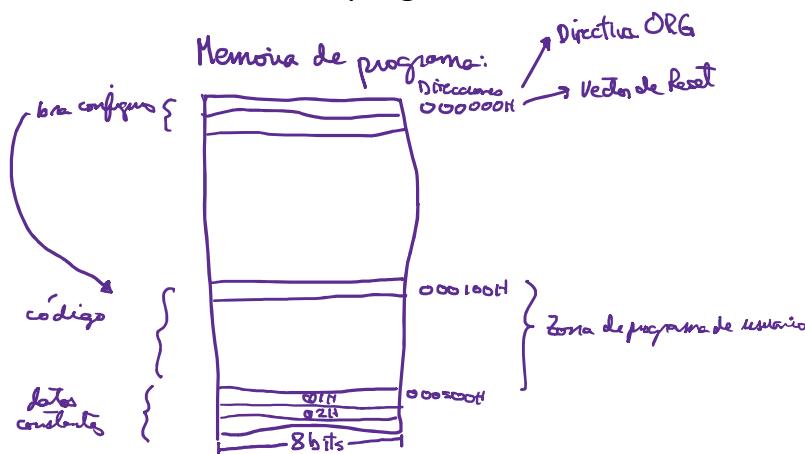
¿Cómo se resolvía el ejercicio del auto fantástico?

- Trabajando con tabla de búsqueda (LUT)



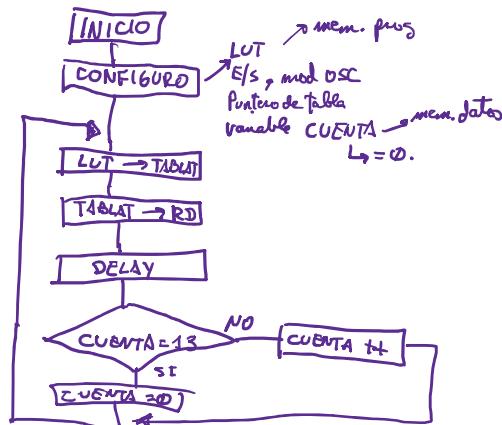
¿Cómo se resolvía el ejercicio del auto fantástico?

- Organización de la memoria de programa



¿Cómo se resolvía el ejercicio del auto fantástico?

- Diagrama de flujo del ejercicio del auto fantástico



Preguntas previas:

- Tengo errores al momento de realizar el primer ejemplo con el MPLAB X v6.15, me arroja “lexical error”

movlw 0x10
movlw 10H
movlw 010H ✓

Muy probablemente error de formato de un número declarado en el código XC8 PIC Assembler

- ¿Cuál es la diferencia entre BRA y GOTO?
 - BRA saltos cortos (ocupa 2 bytes al usarlo)
 - GOTO salto largos (ocupa 4 bytes al usarlo)

Preguntas previas:

- He visto videos de youtube y algunos libros (el del pic16f84 supongo de José Angulo Usategui) donde hay algunas discrepancias en la representación de números en el Assembler
 - No hay discrepancias, un número en cualquiera de sus representaciones numéricas (hex, bin o dec) siempre es el mismo valor, muy posible que sea por la mala interpretación de las instrucciones empleadas.

movlb -> rango de 0 a 63
 movlw -> rango de 0 a 255
 lfsr -> rango de 0 a 16383
- ¿Es mejor emplear el ASM para cuando use periféricos como I2C, SPI, UART frente al XC8? Porque estuve haciendo unas pruebas de esos módulos en ASM y no me salía
 - No te debe de salir por no configurar correctamente los módulos, pero se recomienda utilizar lenguaje de alto nivel ya que luego de emplear esos módulos muy posiblemente necesites hacer operaciones matemáticas (escalamiento, filtrado, promediación, control de errores, etc) el cual de hacerlo en ASM te va a demorar demasiado tiempo en implementarlo, cosa que en XC8 alto nivel sería mas rápido de hacer ya que puedes hacer operaciones aritméticas complejas y de precisión con float.

Preguntas previas:

- No me compila el proyecto en el MPLABX y no sale donde esta el error en la ventana de output
 - Muy posiblemente se ha colocado caracteres restringidos en el nombre yó en la ruta del proyecto.
 - Ej. Semana_2,45 <- No debe de colocarse el punto en el nombre
 - Ej. Semana 2_3 <- De preferencia no dejar espacios, rellenarlo con guion bajo
- No esta habilitado el botón de grabación del microcontrolador en el MPLABX
 - No has conectado el Curiosity Nano, o el cable no tiene capacidad de transferencia de datos
 - Te has olvidado de escoger la herramienta (propiedades del proyecto: Tool)
- No se colorean las palabras en el código fuente.
 - Primero deben de grabar todos los archivos fuente presionando el botón de diskette.

Preguntas previas:

- ¿Puede mostrarnos el diagrama esquemático y el diagrama de flujo de la asignación de la semana 2?

Diagrama esquemático:

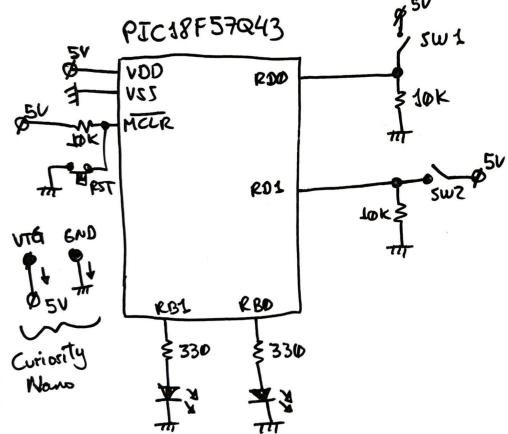
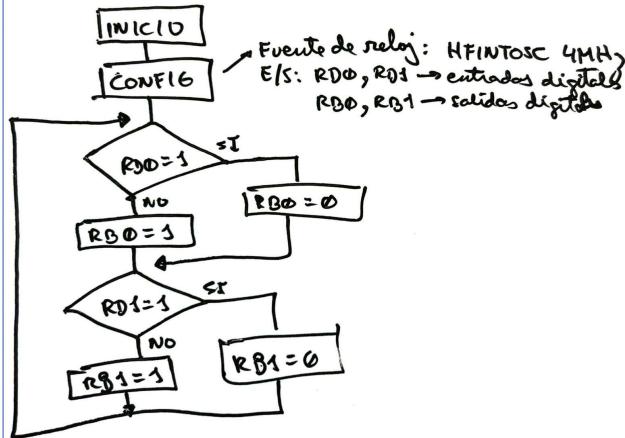


Diagrama de Flujo:



Preguntas previas

- Cantidad de vistas de las grabaciones de clase de los laboratorios de la semana 2:

Analytics

Total views: 14

Last viewed: April 14th, 2025 5:50:44 AM

[Close](#)

Analytics

Total views: 8

Last viewed: April 14th, 2025 8:51:50 AM

[Close](#)

Agenda:

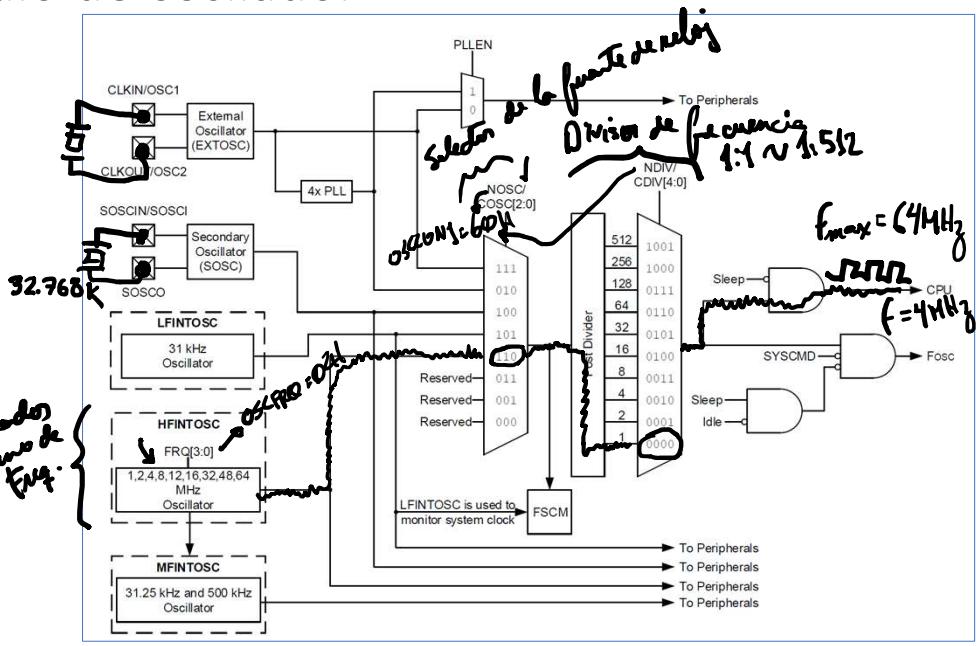
- Estructura interna de un microcontrolador
- Memoria de programa del microcontrolador PIC18F57Q43
- Memoria de datos del microcontrolador PIC18F57Q43
- Instrucciones básicas en XC8 PIC Assembler para el PIC18F57Q43
- ~~El contador de programa (PC) del CPU del microcontrolador PIC18F57Q43~~
- ~~Acceso a datos almacenados en la memoria de programa empleando el puntero del tabla (TBLPTR)~~
- ~~Acceso mediante punteros FSRx/INDFx a la memoria de datos~~
- ~~Interface a display de siete segmentos~~
- ~~Instrucciones de comparación numérica en XC8 PIC Assembler~~

Cambio de horario a partir de semana 3

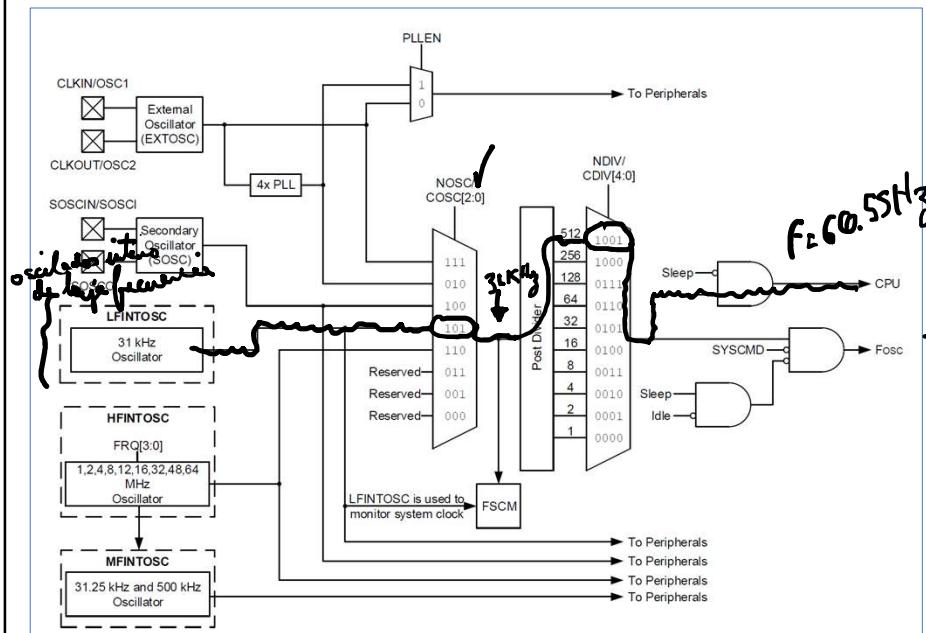
- Jueves 5PM-7PM y 7-10PM
- Viernes 5PM-7PM y 7-10PM
- La semana 3 el jueves se tendrá la clase grabada y el viernes haremos la discusión en vivo.

El módulo de oscilador

- Nosotros usaremos el HFINTOSC para nuestros ejemplos en el laboratorio



Camilo: ¿Cuál es la mínima frecuencia posible?



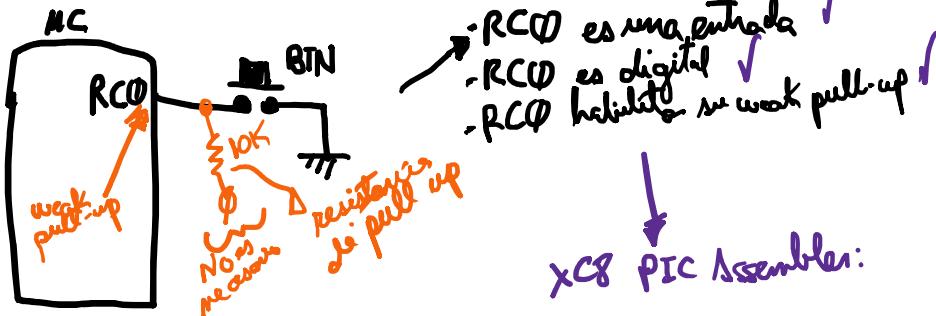
- Si la frecuencia que recibe el CPU es de 60.55Hz, cada instrucción simple en ensamblador se ejecutará en:

$$t_{\text{dec}} = \left(\frac{\text{FOSC}}{q} \right)^{-1}$$

$$= 6.6 \text{ ms}$$

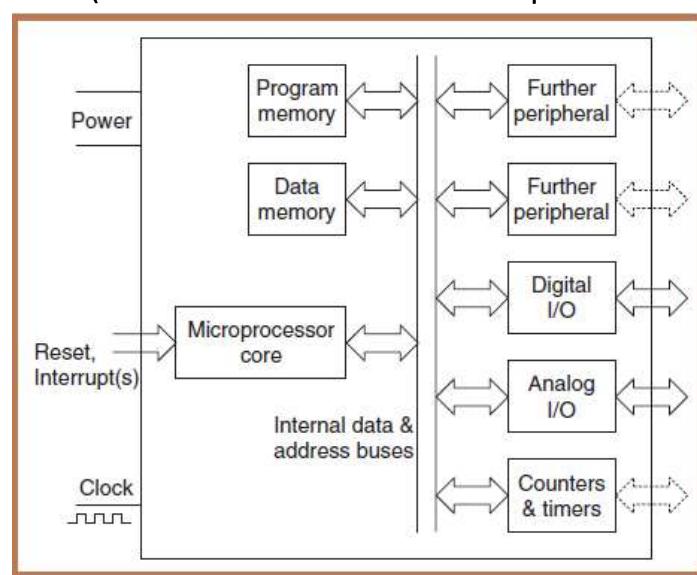
Recordando la configuración de los puertos de E/S:

- Definir las configuraciones al siguiente caso:



Según CLD: Los entradas no deben de estar al aire!

Estructura interna de un microcontrolador (introducción a la arquitectura de computadores)

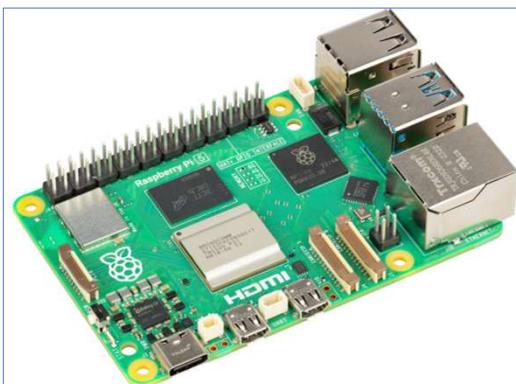


- CPU: Encargado de ejecutar las instrucciones
- Soporte del CPU: módulo de reloj, dispositivos de seguridad, etc.
- Memoria de programa:** No volátil, almacena programa y datos constantes
- Memoria de datos:** Volátil, almacena datos temporales y contiene los registros SFR
- Entradas y salidas
- Periféricos

¿Cuál es la diferencia entre microprocesador y microcontrolador?

- **El microprocesador** es la unidad central de proceso de un computador o un microcontrolador.
- **El microcontrolador** contiene todos los elementos para un funcionamiento autónomo, es decir, contiene un microprocesador, memorias y dispositivos de E/S.

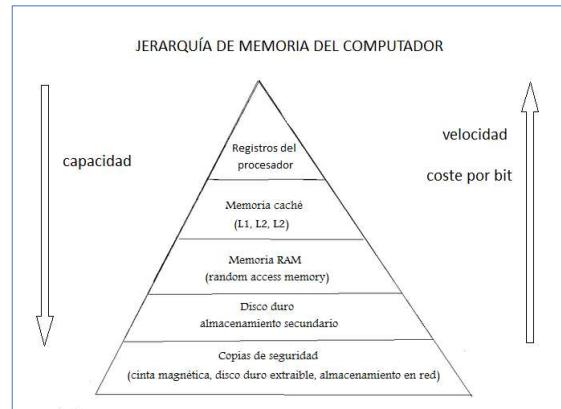
¿Y el Raspberry Pi qué cosa es, un microprocesador o un microcontrolador?



- No es un microcontrolador, tampoco es un microprocesador!
- Computador de Placa Reducida (SBC – Single Board Computer)

¿Por qué la memoria de datos siempre es del tipo volátil?

- La memoria de datos tiene que equiparar la velocidad de trabajo del CPU. La RAM es la que puede llegar a cumplir dicho requerimiento.



El Microcontrolador PIC18 de Microchip

- Estructura de la memoria de **programa** del PIC18F57Q43 y PIC18F47Q10:
 - 128Kbyte de capacidad (000000H-01FFFFH)
 - Data EEPROM (1Kbyte) se encuentra mapeado en 380000H
 - Bits de configuración están mapeadas en 300000H – 300009H
 - Rango de direcciones: 000000H-3FFFFFH (22 bits – 4Mbyte de direcciones)
- Recordar que la dirección 000000H es el vector de RESET

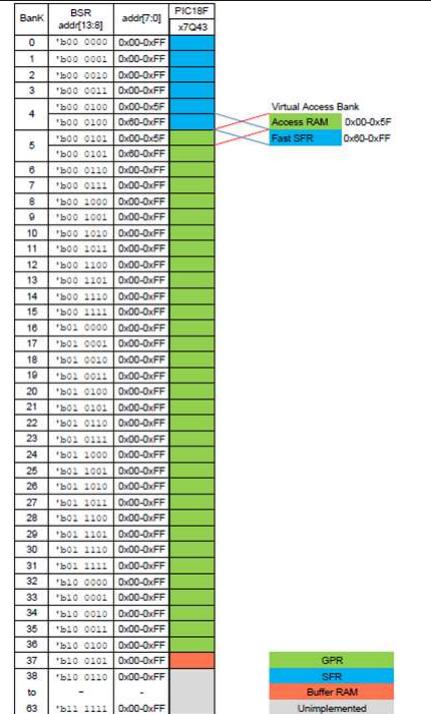
Address	Device
00 0000h to 00 3FFFh	PIC18F57Q43
00 4000h to 00 7FFFh	Program Flash Memory (64 KW) ⁽¹⁾
00 8000h to 00 FFFFh	
01 0000h to 01 FFFFh	
02 0000h to 1F FFFFh	Not Present ⁽²⁾
20 0000h to 20 003Fh	User ID ⁽³⁾ (32 Words) ⁽³⁾
20 0040h to 2B FFFFh	Reserved
2C 0000h to 2C 009Fh	Device Information Area (DIA)
2C 0100h to 2F FFFFh	Reserved
30 0000h to 30 0009h	Configuration Bytes ⁽²⁾
30 000Ah to 37 FFFFh	Reserved
38 0000h to 38 03FFh	Data EEPROM (1024 Bytes)
38 0400h to 3B FFFFh	Reserved
3C 0000h to 3C 0009h	Device Configuration Information
3C 000Ah to 3F FFFFh	Reserved
3F FFFCh to 3F FFFDh	Revision ID (1 Word) ^(2,4,5)
3F FFFEh to 3F FFFFh	Device ID (1 Word) ^(2,4,5)

Notes: 1. Storage Area Flash is implemented as the last 128 Words of User Flash, if enabled.
 2. The addresses do not roll over. The region is read as "0".
 3. Not code-protected.
 4. Hard-coded in silicon.
 5. This region cannot be written by the user and it is not affected by a Bulk Erase.

El Curiosity Nano PIC18F de Microchip

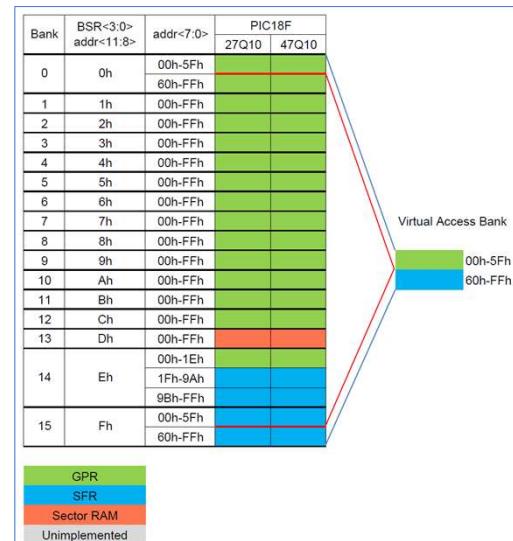
- Estructura de la memoria de **datos** del PIC18F57Q43:
 - Memoria del tipo volátil (se borra el contenido en un PoR – Power-on Reset)
 - A diferencia del PIC18F45K50, la memoria RAM (GPR) esta mapeada a partir del Bank 5 (500H) y los registros de funciones especiales (SFR) se encuentran entre Bank 0 y Bank 4
 - Los SFR son los registros que permiten configurar algún recurso del microcontrolador (ej fuente de reloj, manipulación de las E/S)
 - Tener en cuenta que la RAM de datos es de 8Kbyte
 - El rango total de direcciones: 0000H-3FFFH (14 bits - 16384 direcciones)

b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0			
1	0	0	1	0	1	0	0	0	0	0	0	0	0	2500H		
1	0	0	1	0	1	1	1	1	1	1	1	1	1	125FFH		
														2400H		
														24FFH		



El Curiosity Nano PIC18F de Microchip

- Estructura de la memoria de **datos** del PIC18F57Q10:
 - Memoria del tipo volátil (se borra el contenido en un PoR – Power-on Reset)
 - A diferencia del PIC18F57Q43, la memoria RAM (GPR) esta mapeada a partir del Bank 0 (000H) y los registros de funciones especiales (SFR) se encuentran entre Bank14 y Bank15 empezando desde la dirección E1FH
 - Los SFR son los registros que permiten configurar algún recurso del microcontrolador (ej fuente de reloj, manipulación de las E/S)
 - Tener en cuenta que la RAM de datos es de 4Kbyte
 - El rango total de direcciones: 000H-FFFH (12 bits - 4096 direcciones)



La importancia de escoger el banco correcto en la memoria de datos

- Los registros ó direcciones en la memoria de datos se encuentran clasificados en bancos (Bank) que son los bits mas significativos (bit 8 al bit 13) del valor de dirección.
- Es importante seleccionar el banco correcto antes de manipular un registro en la memoria de datos, revisar la hoja técnica para ver en qué banco se encuentra el registro a manipular.
- Por ejemplo: Deseo manipular el puerto RE0 como salida digital

Ejemplo para PIC18F57Q43:
 movlb 4H ;me voy al Bank4
 bcf TRISE, 0, 1 ;RE0 sea salida
 bcf ANSELE, 1, 1 ;RE0 sea digital

Ejemplo para PIC18F47Q10:
 movlb 0FH ;me voy al Bank15
 bcf TRISE, 0, 1 ;RE0 sea salida
 bcf ANSELE, 1, 1 ;RE0 sea digital

Recordando: Estructuras anidadas

En el lenguaje C:

```

unsigned char x-var = 0;           // 8 bits
for(x-var = 0; x-var < 10; x-var++) {
    }                                // se va a repetir 10 veces
}
  
```

límite → 255

Cómo hago para hacer más repeticiones teniendo como límite el tipo de variable?

```

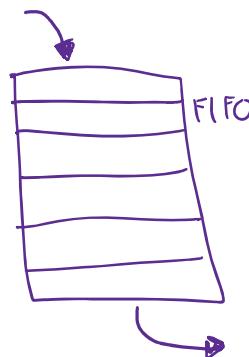
unsigned char x-var=0, y-var=0;
for(x-var=0; x-var<200; x-var++) {
    for(y-var=0; y-var<200; y-var++) {
        }                                // ¿Cuántas veces se repite? → 40000
    }
}
  
```

Recordando saltos a sub-rutinas

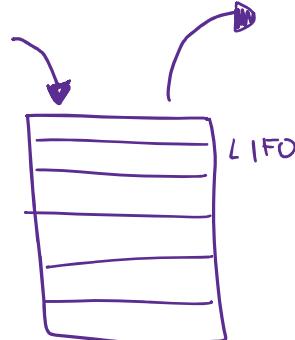
- ¿Cómo sabe el CPU que debe de retornar de donde saltó?
 - Existe una memoria el cual almacena la dirección de retorno y que le va a permitir al CPU regresar una vez atendido la eventualidad.
 - Dicha memoria se le conoce como “stack” o de pila (acción de apilar).

Memorias “Stack” o de pila (apilamiento)

- Estructura FIFO
(first in – first out)



- Estructura LIFO
(last in – first out)



Aspectos relacionados con el obsoleto MPASM y el actual XC8 PIC Assembler

- El año 2014 Microchip dejó de lado el MPASM para dar lugar al XC8 PIC Assembler (PIC-AS)
- Nuevos diseños deberán emplean XC8 PIC Assembler en lugar de MPASM.
- XC8 PIC Assembler es un lenguaje de bajo nivel (orientado a la máquina), **nosotros debemos de conocer primero cómo funciona la máquina** para luego hacer que funcione mediante la codificación de un programa en Assembler y dar solución al problema planteado.

MPASM vs XC8 PIC Assembler: Partes de un programa

MPASM:

```

list p=18f4550
#include<pl18f4550.inc>

;too; aqui declaramos los bi
CONFIG FOSC = XT_XT
CONFIG PWRT = ON
CONFIG BOR = OFF
CONFIG WDT = OFF
CONFIG PBADEN = OFF
CONFIG IVP = OFF

.org 0x0000
goto configuracion

.org 0x0020
configuracion:
    bsf TRISB, 0
    bcf TRISD, 0

principal:
    btfss PORTB, 0
    goto principal
    btg LATD, 0
otro:
    btfsc PORTB, 0
    goto otro
    goto principal

end

```

Labels

Directivas

Configuraciones

Programa

XC8 PIC ASM:

```

PROCESSOR 18F4550
#include "cabecera.inc"

PSECT rstVect,class=CODE, reloc=2, abs
ORG 0000H

rstVect:
    goto configuracion
    ORG 0020H

configuracion:
    bsf TRISB, 0, 0
    bcf TRISD, 0, 0

principal:
    btfss PORTB, 0
    goto principal
    btg LATD, 0, 0
otro:
    btfsc PORTB, 0
    goto otro
    goto principal

end rstVect

```

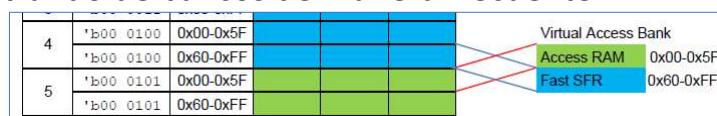
Nota: Los bits de configuración se alojaron en un archivo header llamado "cabecera.inc"

¿Por qué aprender Assembler (más difícil) si en el C también se puede hacer?

- En el assembler se prioriza en la eficiencia (espacio empleado en la memoria de programa y de datos, control detallado del consumo energético y del funcionamiento en general)
 - En el C se prioriza en menor tiempo de Desarrollo
 - Hay que recordar que el uso de lenguaje de alto nivel siempre será menos eficiente frente al assembler (va a ocupar mas espacio, lo vas a ver como caja negra al microcontrolador)

Access-Bank vs BSR

- Son las formas para acceder a la memoria de datos
 - **Access-Bank** es una forma directa (pero limitada) de acceder a la memoria datos, se usa para tener acceso a una porción del Bank4 (460H-4FFH) y una porción de memoria RAM (500H-55FH) y evitar estar cambiando de bancos de manera frecuente



- **BSR (Bank Select Register)** es la forma estándar para acceder a la memoria de datos, previamente se tiene que seleccionar el banco de entre 0 y 63 usando el registro selector de bancos BSR ó usando directamente la instrucción “movlb”

Repertorio de instrucciones en Assembler del PIC18

- Revisar capítulo 44 de la hoja técnica del PIC18F57Q43

Table 44-2. Standard Instruction Set

Mnemonic Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb			LSb		
BYTE-ORIENTED FILE REGISTER INSTRUCTIONS								
ADDWF f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1
ADIWFC f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1
ANDWF f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1
CLRF f, a	Clear f	1	0010	101a	ffff	ffff	Z	
COMF f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1
DECF f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1
INCF f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1
IORWF f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1
MOVF f, d, a	Move f to WREG or f	1	0001	00da	ffff	ffff	Z, N	1
MOVWF f ₀ , f ₀	Move f ₀ (12-bit source) to f ₀ (12-bit destination)	2	1100	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	None	1, 3, 4
			1111	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀		
			0000	0000	0110	0110		
			1111	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	None	1, 3
			1111	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀	f ₀ f ₀ f ₀ f ₀		
MOVWF f, a	Move WREG to f	1	0010	111a	ffff	ffff	None	
MULWF f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1
NEGF f, a	Negate f	1	0010	110a	ffff	ffff	C, DC, Z, OV, N	1
RLCF f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1
RLCF f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1
RCRF f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	1
RRCF f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	1
SETF f, a	Set f	1	0010	100a	ffff	ffff	None	
SUBFWB f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1
SUBFWF f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1
SUBWFB f, d, a	Subtract WREG from f with Borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1
XORWF f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	1
			0001	00da	ffff	ffff	Z, N	1

Repertorio de instrucciones en Assembler del PIC18

- Revisen las instrucciones que se han empleado en el ejemplo de la semana pasada con esta tabla de instrucciones:

- movlb → seleccionar el banco de trabajo, ej movlb 0H ;vas al Bank0
- movlw → mover un literal hacia el registro W, ej movlw 60H
- movwf → mover el contenido de W hacia un registro (memoria de datos)
- bsf → setear un bit de un registro (poner a uno lógico)
- bcf → resetear un bit de un registro (poner a cero lógico)
- btfss → preguntar si un bit de un registro es igual a uno
- bra → salto incondicional, ej bra inicio

Detalle de una instrucción con opción {a}

- Ejemplo:

Trabajando con Access bank:
movwf TRISB, 0

Trabajando con BSR:

(previamente especificar en el BSR cuál es el banco de acceso)

movwf TRISB, 1

access bank

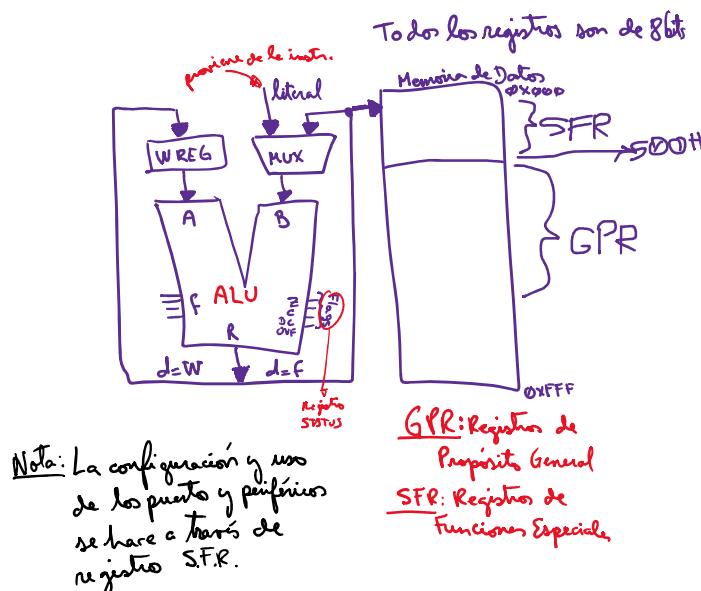
movwf TRISB, a
movwf TRISB, b

BSR

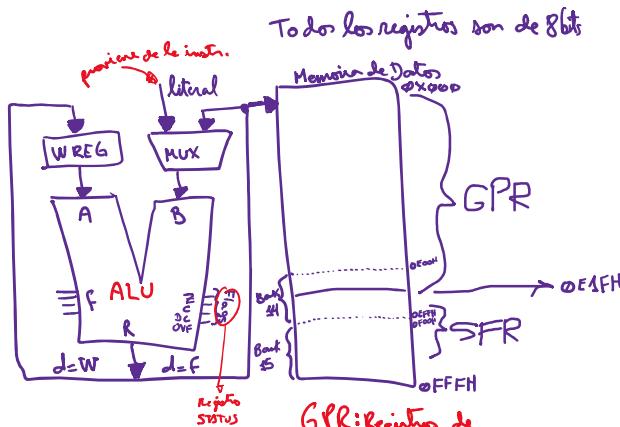
- No todas las instrucciones tienen el parámetro {a}, revisar capítulo 44 del datasheet

MOVWF	Move W to f		
Syntax	MOVWF f { ,a }		
Operands	0 ≤ f ≤ 255 a ∈ [0, 1]		
Operation	(W) → f		
Status Affected	None		
Encoding	0110 111a ffff ffff		
Description	Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Mode for details.		
Words	1		
Cycles	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read W	Process Data	Write register 'f'
Example: MOVWF REG, 0			
Before Instruction			
W = 4Fh REG = FFh			
After Instruction			
W = 4Fh REG = 4Fh			

El flujo de datos en el microcontrolador PIC18F57Q43



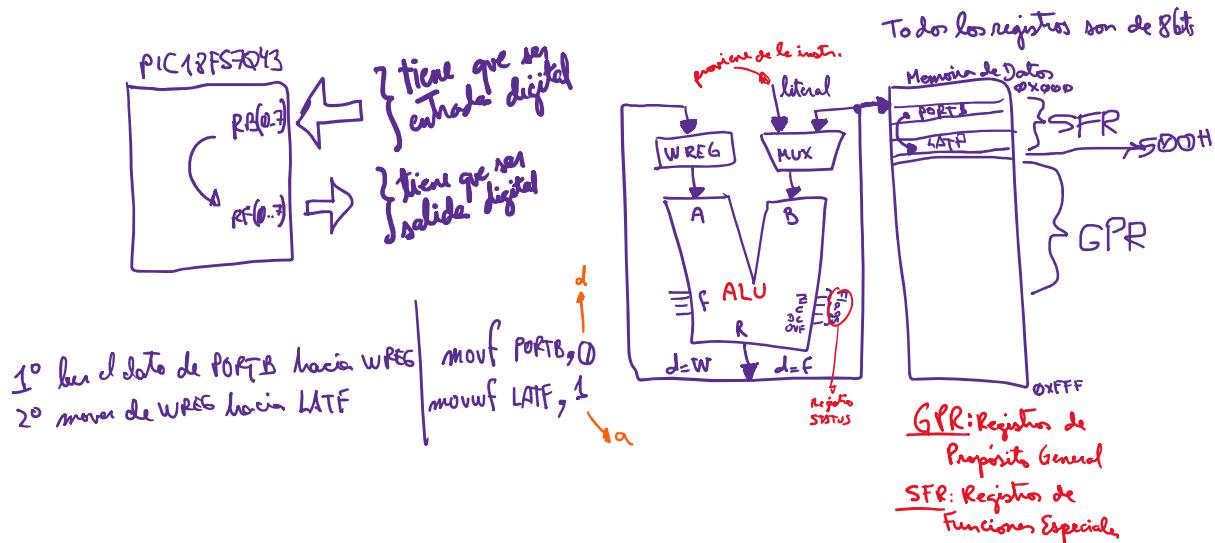
El flujo de datos en el microcontrolador PIC18F47Q10



Comentarios y Notas:

- En los programas desarrollados en XC8 PIC Assembler, la mayor cantidad de instrucciones serán las de movimiento de datos.
- Es preferible trabajar con BSR ($a=1$), debido a que en el modo Access Bank solo se tiene una parte del registros del Bank4 y una parte de GPR del Bank5, si tu quieres emplear algún otro registro fuera de ese grupo, igual tendrás que el BSR.

Ejemplo: Pasar un dato del puerto B al puerto F



Registro STATUS

- Encuentras las banderas de la ALU, se actualizan cuando ocurre alguna operación aritmética ó lógica en este dispositivo.

7.7.7 STATUS									
Name: STATUS									
Address: 0x4D8									
STATUS Register									
Bit	7	6	5	4	3	2	1	0	
Access		TO	PD	N	OV	Z	DC	R/W	
Reset	1	1	0	0	0	0	0	0	

de la ALU

Bandera "N": Cuando en una operación que realiza la ALU el resultado fué un número negativo

Bandera "OV": Cuando ocurre un desbordamiento del dato que se ha incrementado

Bandera "Z": Cuando en una operación que realiza la ALU el resultado salió cero (00000000B ó 00H ó 0D)

Bandera "DC": El digit carry

Bandera "C/~B": Bit carry/~borrow, empleado en las operaciones aritméticas de suma y resta

Instrucciones básicas en XC8 PIC Assembler

- movlb -> para establecer el banco donde se va a trabajar
- movlw -> para mover un literal hacia el registro Wreg
- movwf -> para mover contenido del Wreg hacia un registro
- movff -> para mover el contenido de un registro a otro registro
- bsf, bcf -> para colocar 1 ó 0 a un bit (7-0) de un registro
- btfss, btfsc -> para probar si un bit es uno ó cero
- nop -> no operación (pierde el tiempo según t_ejec)
- decf, incf -> para decrementar/incrementar el valor de un registro
- incfsz, decfsz -> incremento/decreto con pregunta si llegó a cero
- setf, clrf -> coloca todos a uno/cero los bits de un registro
- comf -> complementa todos los bits de un registro
- bra, goto -> saltos, bra = cortos, goto = largos
- call, return -> saltos a subrutina (con opción a retornar)

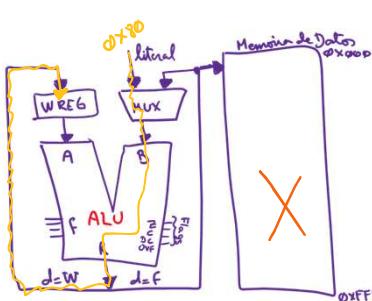
Instrucciones básicas en XC8 PIC Assembler

- Instrucciones de movimiento de datos

movlw [literal]

-mover un literal hacia WREG

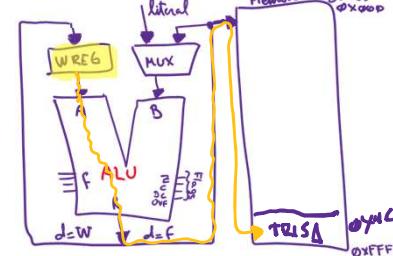
Ej: movlw 0x80



movwf [registro], f

-mover el contenido de WREG hacia [registro]

Ej: movwf TRISA, f

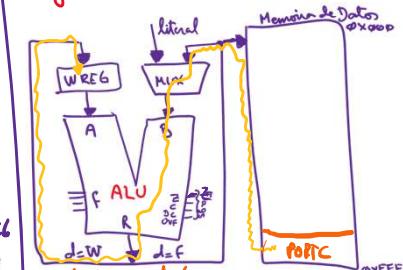


movf [registro], d , f

mover el contenido de [registro] y lo muere según "d".

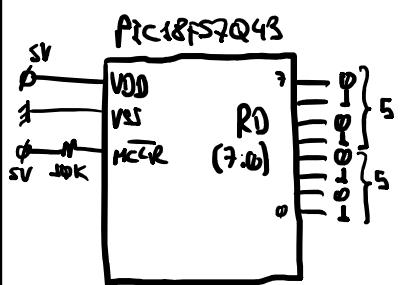
movlw 4H

Ej: movf PORTC, 0, f



Ejemplo: Escribir el número 55H en el puerto D del microcontrolador PIC18F57Q43

- Primero: hacer que el puerto D sea salida digital
- Segundo: cargar el numero 55H en el registro W
- Tercero: mover el contenido de registro W hacia LATD



```

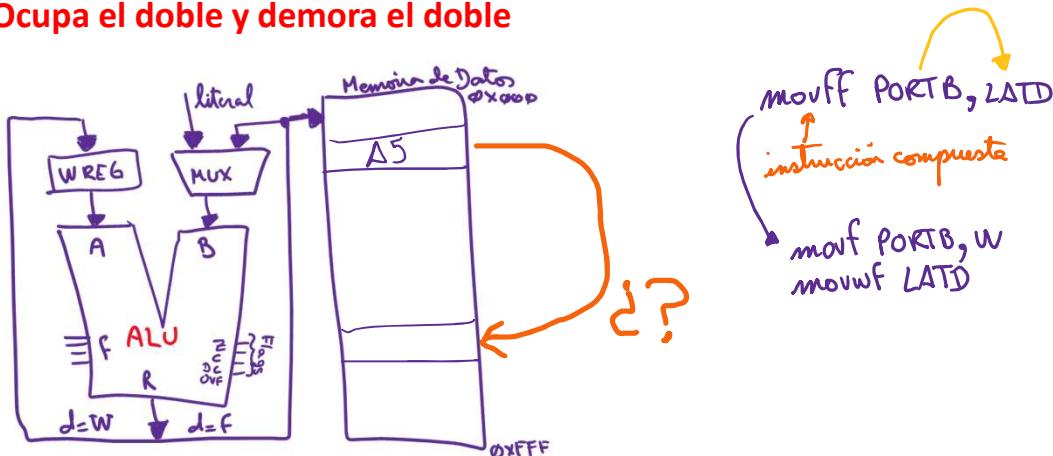
inicio:    movlb 4H      ;yendo al Bank4
           movlw 00H      ;cargo literal 00H a Wreg
           movwf TRISD, 1 ;muevo contenido de Wreg hacia TRISD
           movwf ANSELD, 1 ;muevo contenido de Wreg hacia ANSEL
           movlw 55H      ;muevo literal 55H hacia Wreg
           movwf LATD, 1   ;muevo contenido de Wreg hacia LATD

end

```

Instrucción movff [registro1], [registro2]

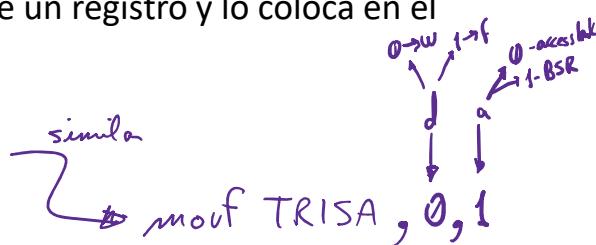
- Mueve el contenido de registro1 hacia registro2
- **Ocupa el doble y demora el doble**



¿Instrucción movfw?

- ¡Instrucción no documentada en la hoja técnica!
- Instrucción "legacy", instrucción de compatibilidad
- Esta no es una instrucción en sí, sino una "pseudo-instrucción" del lenguaje assembler.
- Lo que hace es mover el contenido de un registro y lo coloca en el Wreg.

Ej: movfw TRISA *simila*



movf TRISA, 0, 1

Instrucciones básicas en XC8 PIC Assembler

- Instrucciones de manipulación de un bit determinado, en un registro

bsf [registro], #bit, ^a posición
coloca a "1" el #bit de [registro]
Ej:
 TRISB

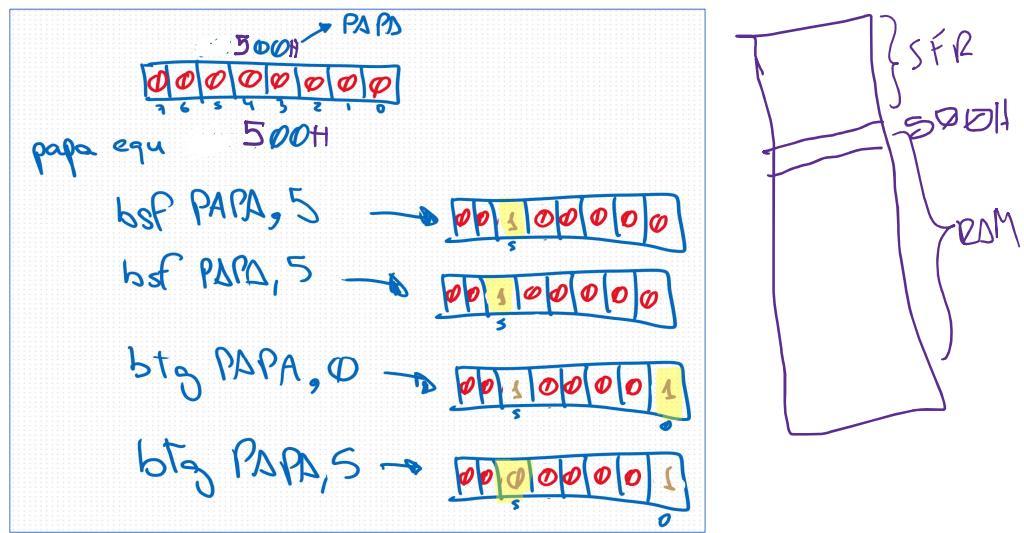
 bsf TRISB, 3, 1
 TRISB

bcf [registro], #bit, ^a
coloca a "0" el #bit de [registro]
Ej:
 TRISB

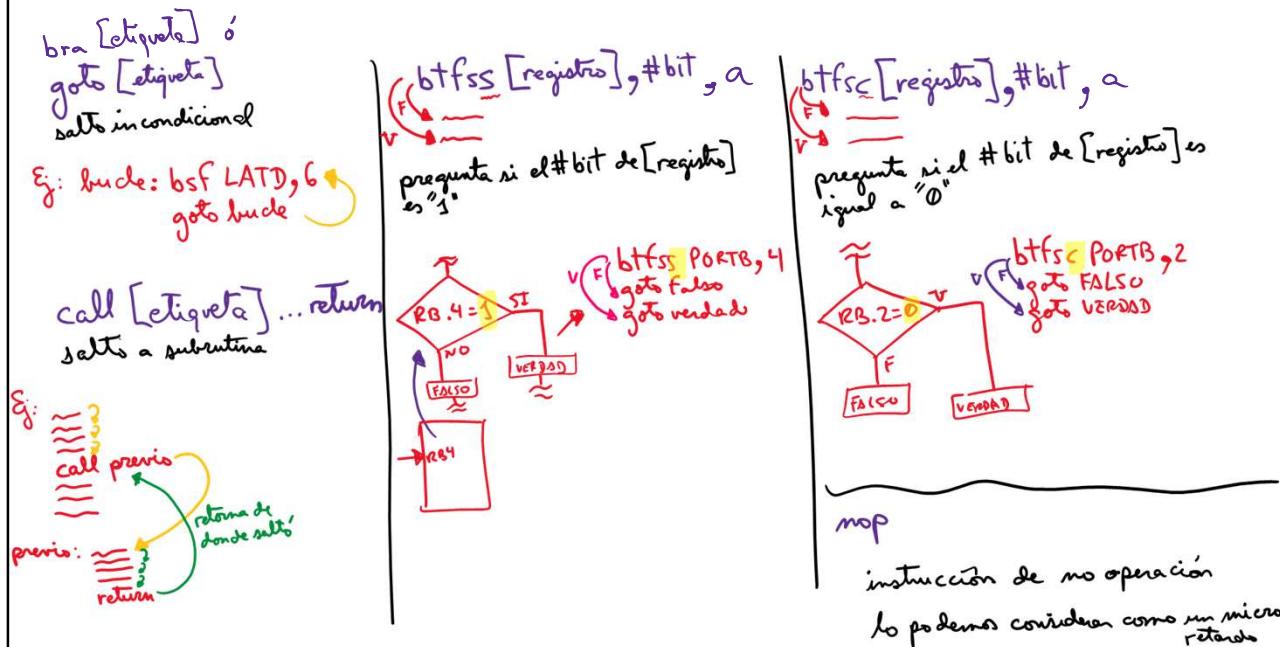
 bcf TRISB, 7, 1
 TRISB

btg [registro], #bit, ^a
aplica complemento al #bit de [registro]
Ej: btg TRISB, 6, 1
 TRISB

Ejemplo de uso de instrucciones de manipulación de bits en un registro



Instrucciones básicas en XC8 PIC Assembler

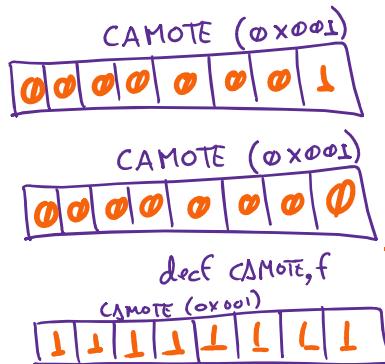


Instrucciones básicas en XC8 PIC Assembler

- Instrucción decf / incf
 - Decreto (decf) o incremento (incf) de registro, ambos de uno en uno

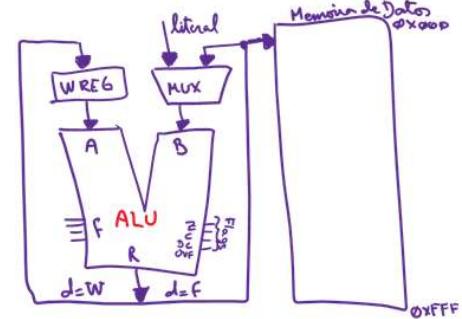
Ej: decf [registro], d, a incf [registro], d, a

"d" puede ser: f ó w



decf CAMOTE, f

Registro STATUS: Z=1



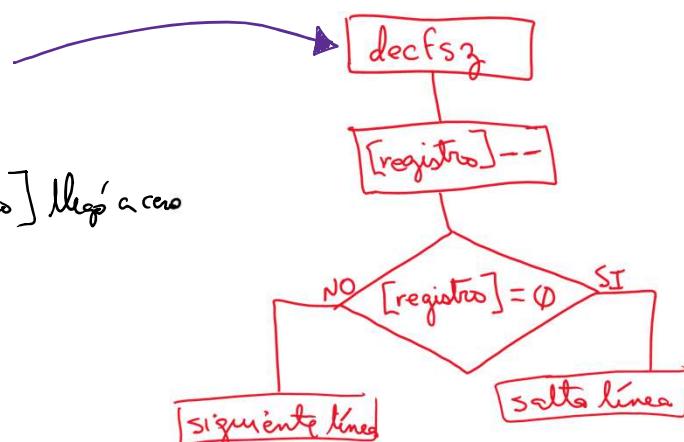
Instrucciones básicas en XC8 PIC Assembler

decfsz [registro], d

decrementa y pregunta si [registro] llegó a cero

incfsz [registro], d

incrementa y pregunta si [registro] llegó a cero



Instrucciones setf [registro], clrf [registro], comf [registro]

- **setf [registro]** : Coloca todos los bits del registro indicado a uno lógico

ejemplo: 

`setf TRISB`

\Rightarrow 

- **clrf [registro]** : Coloca todos los bits del registro indicado a cero lógico

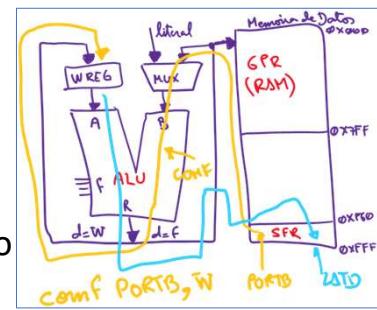
ejemplo: 

`clrf TRISB`



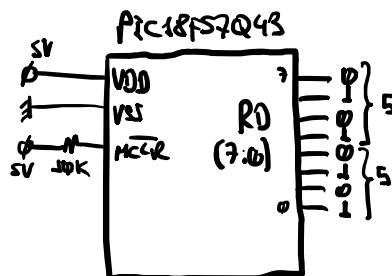


- **comf [registro]** : Complementa todos los bits del registro



Ejemplo: Escribir el número 55H en el puerto D del microcontrolador PIC18F57Q43

- Optimizando con la instrucción clrf



```

inicio: movlb 4H           ;yendo al Bank4
        clrf TRISD, 1      ;TRISD todos sus bits a cero
        clrf ANSELD, 1      ;ANSELD todos sus bits a cero
        movlw 55H            ;muevo literal 55H hacia Wreg
        movwf LATD, 1         ;muevo contenido de Wreg hacia LATD

        end
    
```

Ejemplo: Puerto D todos sus bits como salida digital:

Opción 1: usando bcf

```
bcf TRISD,0
bcf TRISD,1
:
bcf TRISD,7
    R ocho instrucciones
```

Opción 2: usando valor numérico

```
movlw 0DH
movwf TRISD,1
    parámetro!
```

do instrucción

Opción 3: usando clrf

```
clrf TRISD,1
    una sola
    instrucción!
```

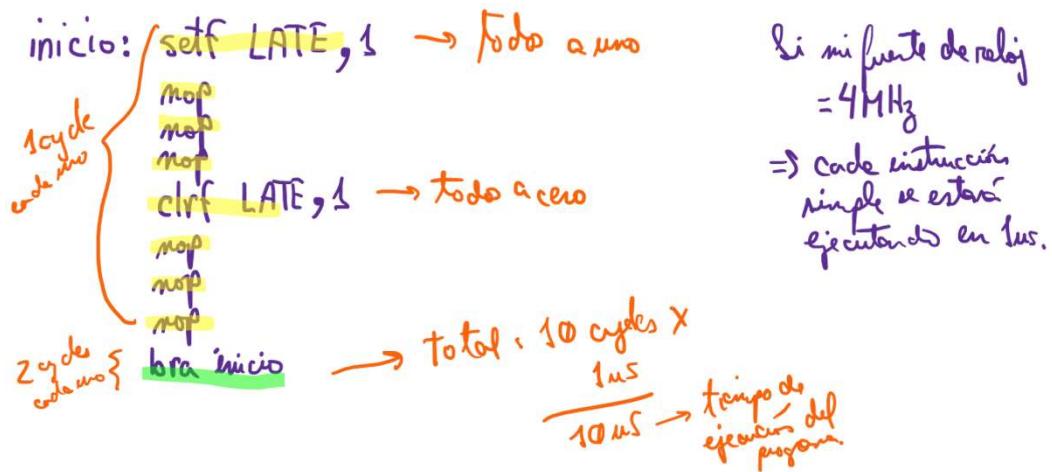
Tiempo de ejecución de las instrucciones en XC8 PIC Assembler

- Se utiliza la siguiente fórmula:

$$t_{opc} = \left(\frac{f_{osc}}{4} \right)^{-1} \text{ s}$$

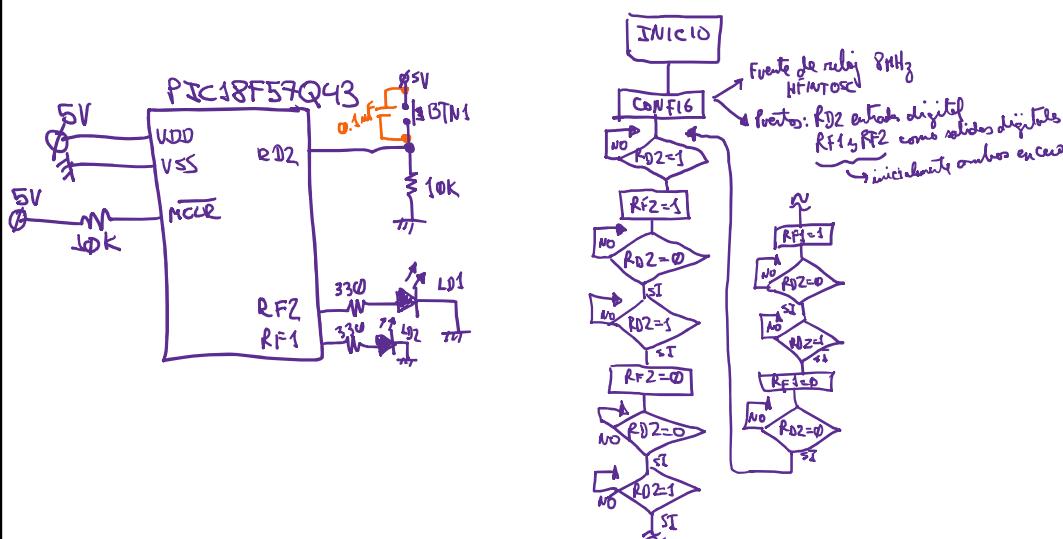
- Hay instrucciones simples, dobles y especiales (revisar 44.0 de la datasheet)
- Recordando la relación periodo vs frecuencia: $f = \frac{1}{T}$

Ejemplo de tiempo de ejecución



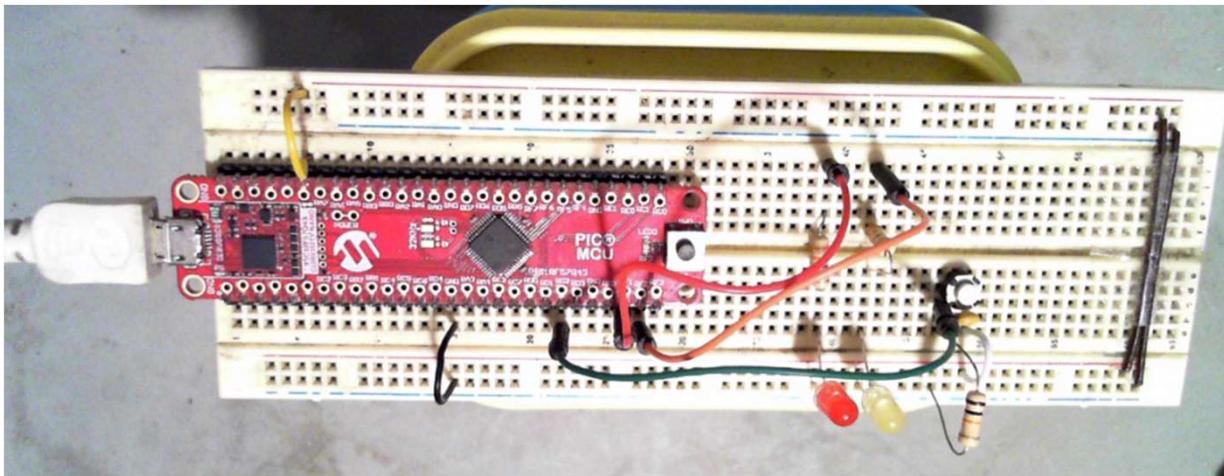
Ejemplo 02: Desarrollar un dispositivo LATCH con el Curiosity Nano PIC18F57Q43

- Latch: Dispositivo enclavador – Mantiene el valor de salida hasta que pulses el botón



Ejemplo 02: Desarrollar un dispositivo LATCH con el Curiosity Nano PIC18F57Q43

- Latch: Dispositivo enclavador – Mantiene el valor de salida hasta que pulses el botón



Ejemplo 02: Desarrollar un dispositivo LATCH con el Curiosity Nano PIC18F57Q43

- Latch: Dispositivo enclavador – Mantiene el valor de salida hasta que pulses el botón

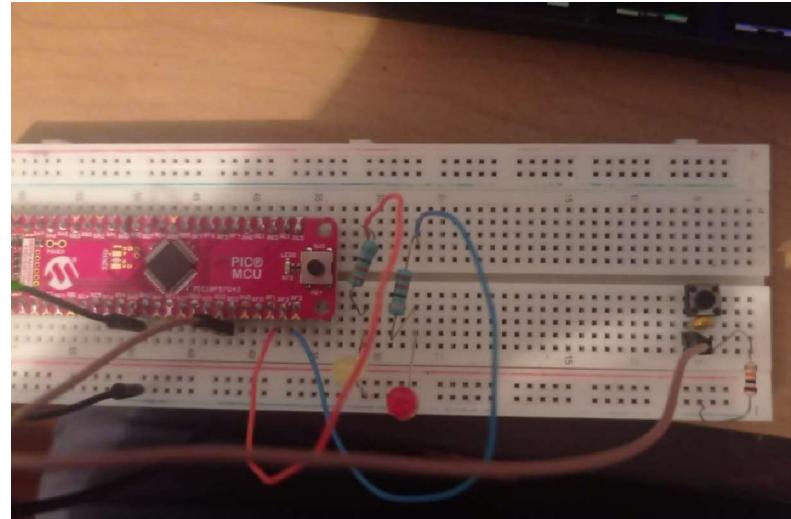
```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6      ORG 000000H          ;vector de reset
7      bra configuro        ;salto a label configuro
8
9  configuro:                ;label configuro
10     movlb 0H              ;al BANK0
11     movlw 60H
12     movwf OSCCON1, 1       ;HFINTOSC y 1:1
13     movlw 03H
14     movwf OSCFRQ, 1         ;HFINTOSC a 8MHz
15     movlw 40H
16     movwf OSCE1, 1          ;HFINTOSC habilitado
17     movlb 4H              ;al BANK4
18     bsf TRISD, 2, 1        ;RD2 como entrada
19     bcf ANSEL0, 2, 1        ;RD2 como digital
20     bcf TRISF, 1, 1        ;RF1 como salida
21     bcf ANSELF, 1, 1        ;RF1 como digital
22     bcf TRISF, 2, 1        ;RF2 como salida
23     bcf ANSELF, 2, 1        ;RF2 como digital
24     bcf LATF, 1, 1          ;RF1 en cero
25     bcf LATF, 2, 1          ;RF2 en cero
26  inicio:                  ;regreso a label inicio
27     btfss PORTD, 2, 1       ;pregunto si presione el boton
28
29     bra inicio             ;falso, no presione el boton y vuelvo a preguntar
30     bsf LATF, 1, 1          ;verdad, enciendo primer LED
31     otrol:
32         btfsc PORTD, 2, 1   ;pregunto si deje de presionar el boton
33         bra otrol            ;falso y vuelvo a preguntar
34     otroal:
35         btfss PORTD, 2, 1   ;pregunto si presione el boton
36         bra otroal            ;verdad, apago el LED
37     otroa2:
38         btfsc PORTD, 2, 1   ;pregunto si solte el boton
39         bra otroa2            ;falso y vuelvo a preguntar
40     otro2:
41         btfss PORTD, 2, 1   ;pregunto si presione el boton
42         bra otro2            ;falso, no presione el boton y vuelvo a preguntar
43         bsf LATF, 2, 1          ;verdad, enciendo primer LED
44     otro3:
45         btfsc PORTD, 2, 1   ;pregunto si deje de presionar el boton
46         bra otro3            ;falso y vuelvo a preguntar
47     otroa3:
48         btfss PORTD, 2, 1   ;pregunto si pulse el boton
49         bra otroa3            ;verdad, apago el LED
50     otro4:
51         btfsc PORTD, 2, 1   ;pregunto si deje de presionar el boton
52         bra otro4            ;falso y vuelvo a preguntar
53     bra inicio             ;regreso a label inicio
54
55     end upcino
56

```

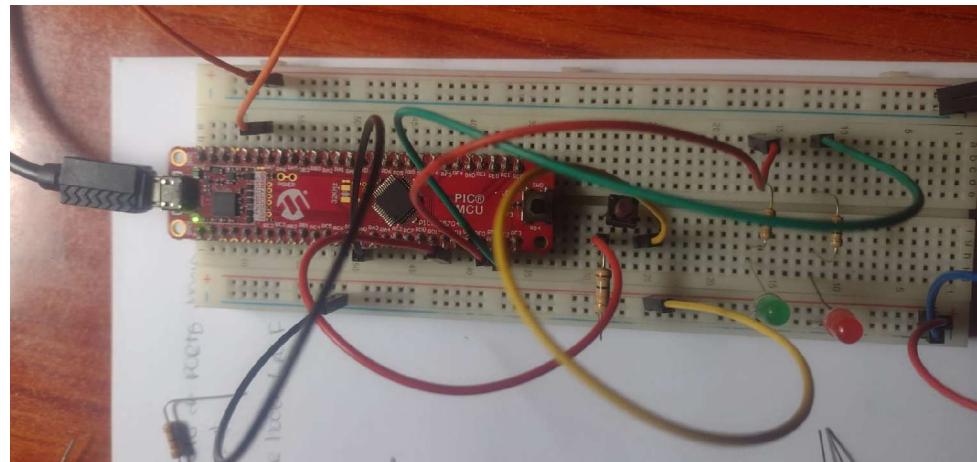
Error en la implementación

- “Profesor compila pero no funciona”
- Ubica el error:



Error en la implementación

- “Profesor compila pero no funciona”
- Ubica el error:

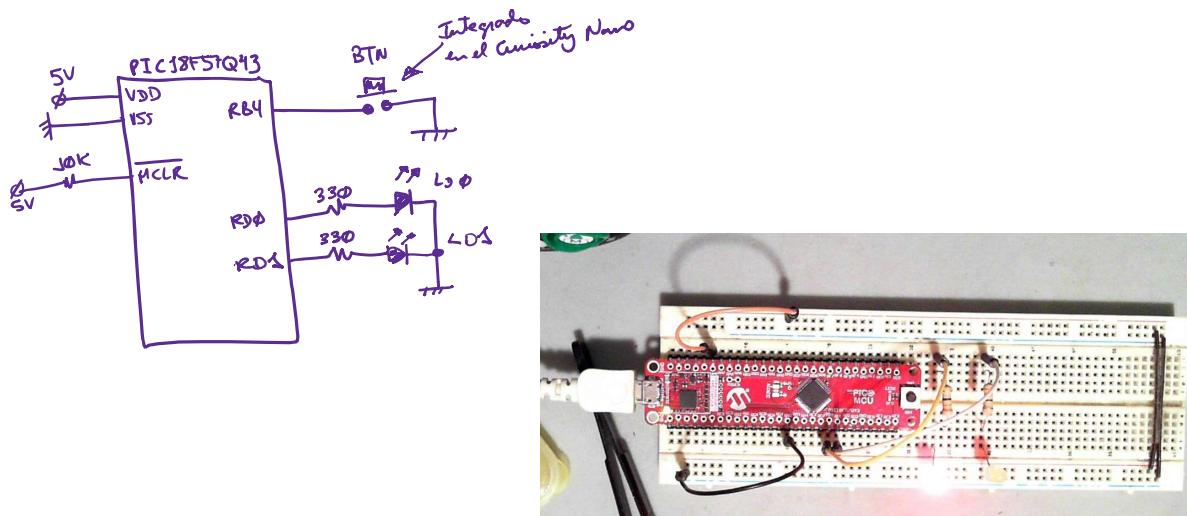


Problemas encontrados

- No funciona bien, parece que hay pulsaciones falsas en el botón
- Los botones tienen problema de rebote, por lo tanto se tienen que implementar una estrategia “anti-rebote”

Ejemplo 03: Señal de tren con dos LEDs controlados por pulsador con el Curiosity Nano PIC18F57Q43

- Diseño del hardware



Ejemplo 03: Señal de tren con dos LEDs controlados por pulsador con el Curiosity Nano PIC18F57Q43

- Código en XC8 PIC Assembler

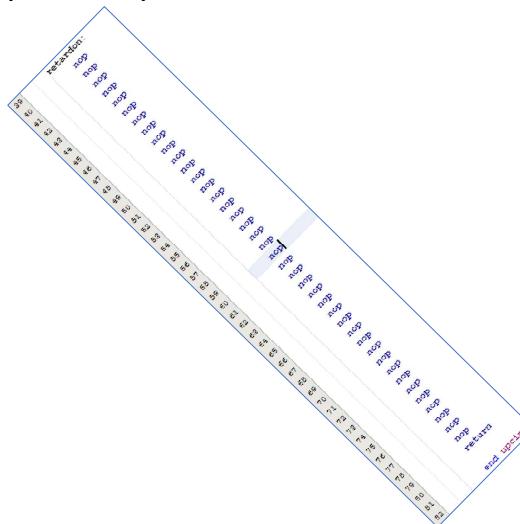
```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6      ORG 000000H
7      bra configuro
8
9      ORG 000020H
10 configuro:
11     movlb 0H
12     movlw 60H
13     movwf OSCCON1, 1
14     movlw 02H
15     movwf OSCFRQ, 1
16     movlw 40H
17     movwf OSCEN, 1
18
19     movlb 4H
20     bsf TRISB, 4, 1
21     bcf ANSELB, 4, 1
22     bsf WPUB, 4, 1
23     movlw 0FCH
24     movwf TRISD, 1          ;RD0 y RD1 como salidas
25     movwf ANSEL0, 1         ;RD0 y RD1 como digitales
26     movlw 01H
27     movwf LATD, 1           ;RD0 encendido y RD1 apagado
28
29     inicio:
30         btfsc PORTB, 4, 1  ;Pregunta si RB4 es cero (si presione el boton)
31         bra inicio        ;falso, regresa a preguntar
32         comf LATD, 1, 1    ;complemento a registro
33         call retardon
34     otro:
35         btfss PORTB, 4, 1  ;Pregunta si RB4 es uno (si solte el boton)
36         bra otro          ;falso, vuelvo a preguntar
37
38
39     retardon:
40         nop
41         nop
42         nop
43         nop
44         nop
45         nop
46         nop
47         nop
48         nop
49         nop
50         nop
51         nop
52         nop
53         nop
54         nop
55         nop
56         nop
57         nop
58         nop
59         nop
60         nop
61         nop
62         nop
63         nop
64         nop
65         nop
66         nop
67         nop
68         nop
69         nop
70         nop
71         nop
72         nop
73         nop
74         nop
75         nop
76         nop
77         nop
78         nop
79         nop
80         return
81
82     end upcino

```

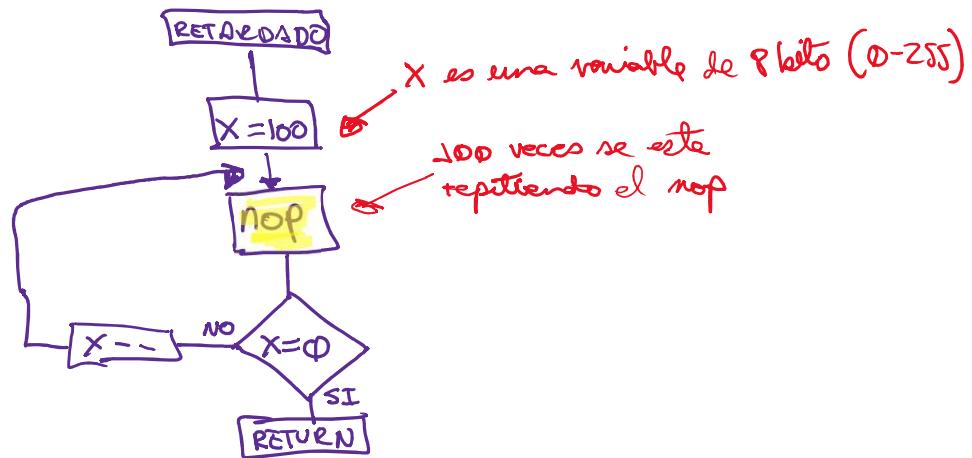
¿Cómo hago el antirrebote del botón sin tener que escribir tanto “nops”?

- Escribir 40000 nops no es práctico ni eficiente!



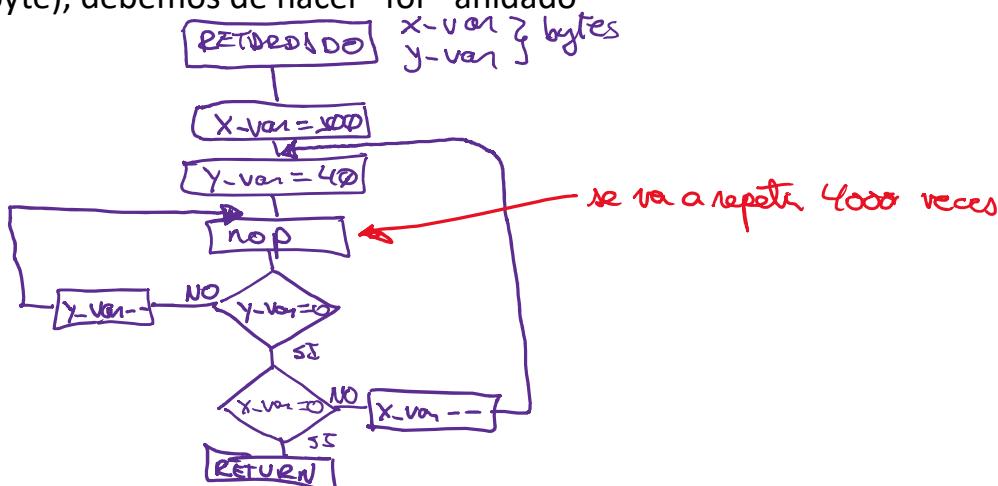
¿Cómo generamos el retardo de 40ms para el antirrebote del botón?

- Usar muchos nops ocupa demasiado espacio
- Se puede armar una rutina de bucle de repetición (for)



¿Cómo generamos el retardo de 40ms para el antirrebote del botón?

- Con un solo bucle de repetición solo llegamos a 255 (valor máximo en un byte), debemos de hacer "for" anidado



¿Cómo generamos el retardo de 40ms para el antirrebote del botón?

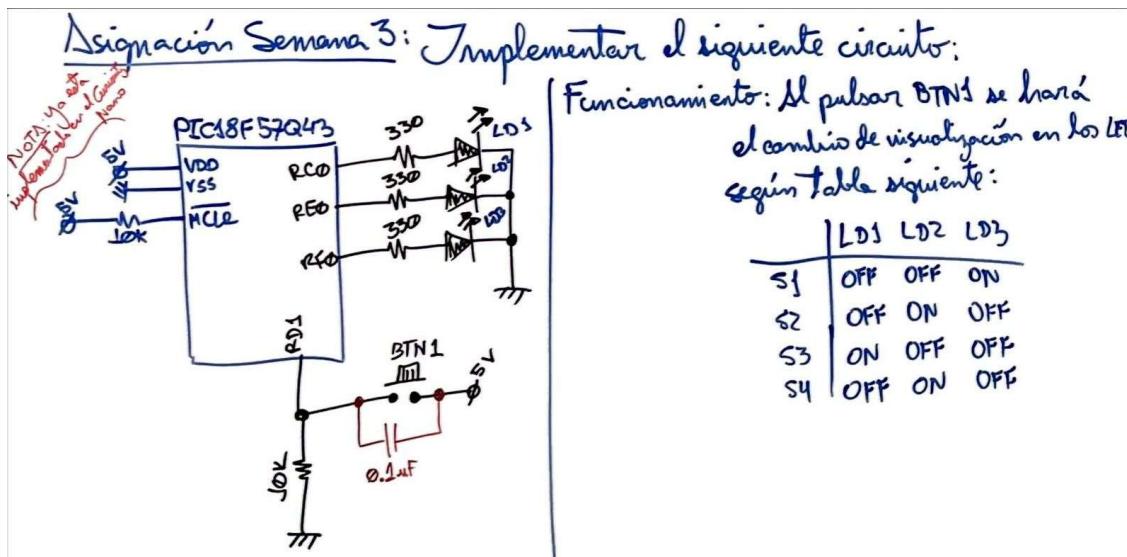
- **x_var** y **y_var** son etiquetas asignadas a posiciones GPR de la memoria de datos, recordando que los GPR empiezan a partir de la dirección 500H en el caso del PIC18F57Q43, para el PIC18F47Q10 las GPR empiezan desde la dirección 000H.
- El código fue obtenido del diagrama de flujo anterior, tener en cuenta que al usar BSR se tiene que estar cambiando de bancos ya que el registro STATUS se encuentra en el BANK4 (4D8H) y los GPRs **x_var** y **y_var** están ubicados en BANK5 (500H y 501H respectivamente). Deberá de modificarse dichos valores de banco de emplearse el PIC18F47Q10.
- Reemplazarlo en los códigos anteriores para obtener un mejor filtro antirrebote

```

40      retardado:
41      movlb 5H
42      movlw 100
43      movwf x_var, 1
44      otro3:
45      movlw 40
46      movwf y_var, 1
47      otro2:
48      nop                                ;se va a ejecutar 40000 veces
49      movf y_var, 1, 1
50      movlb 4H
51      btfss STATUS, 2, 1
52      bra y_var_noescero
53      movlb 5H
54      movf x_var, 1, 1
55      movlb 4H
56      btfss STATUS, 2, 1
57      bra x_var_noescero
58      return
59      y_var_noescero:
60      movlb 5H
61      decf y_var, 1, 1
62      bra otro2
63      x_var_noescero:
64      movlb 5H
65      decf x_var, 1, 1
66      bra otro3

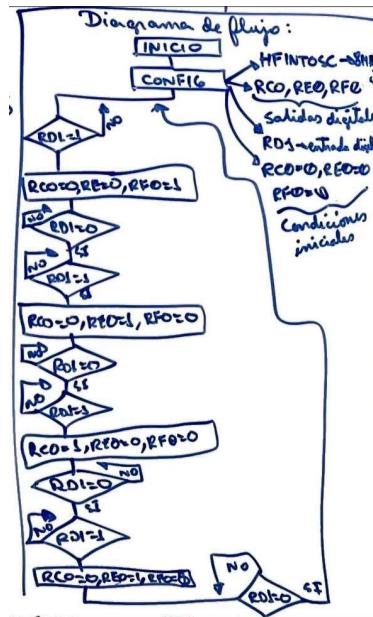
```

Ejemplo 2024-1



Ejemplo 2024-1

- Diagrama de flujo:



```

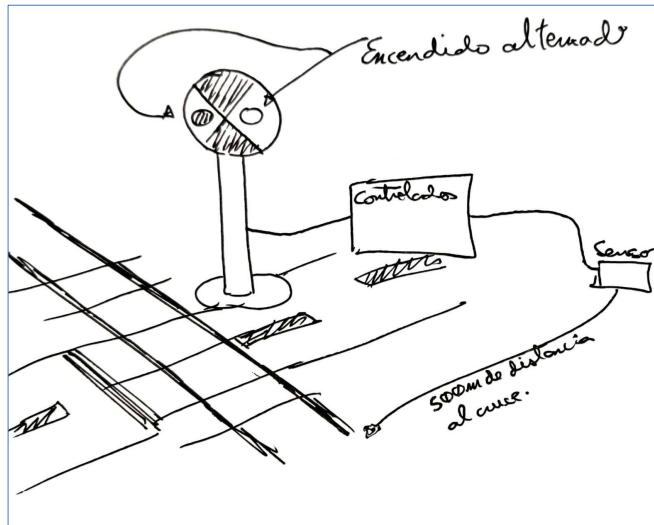
1  PROCESSOR 18F57Q43           ;directiva de procesador
2  #include "cabecera.inc"      ;
3
4  PSECT upcino, class=CODE, reloc=2, abs   ;apertura de program section
5  upcino:                      ;etiqueta upcino
6  ORG 000000H                  ;vector de reset
7  bra configuro                ;salto a etiqueta configuro
8
9  ORG 000100H                  ;zona de programa de usuario
10 configuro:                   ;etiqueta configuro
11  movlb 0H                     ;al BANK0
12  movlw 60H
13  movwf OSCCON1, 1
14  movlw 03H
15  movwf OSCFRQ, 1
16  movlw 40H
17  movwf OSCEN, 1
18  movlw 4H
19  bcf TRISD, 1, 1
20  bcf ANSELB, 1, 1
21  bcf TRISC, 0, 1
22  bcf ANSELC, 0, 1
23  bcf TRISE, 0, 1
24  bcf ANSELF, 0, 1
25  bcf TRISF, 0, 1
26  bcf LATC, 0, 1
27  bcf LATD, 0, 1
28  bcf LATE, 0, 1
29  bcf LATF, 0, 1
30
31 inicio:
32  btfss PORTD, 1, 1
33  bra inicio
34  bcf LATC, 0, 1
35  bcf LATD, 0, 1
36  bcf LATF, 0, 1
37 otrol:
38  btfsc PORTD, 1, 1
39  bra otrol
40 otro2:
41  btfss PORTD, 1, 1
42  bra otro2
43  bcf LATC, 0, 1
44  bcf LATD, 0, 1
45  bcf LATF, 0, 1
46 otro3:
47  btfss PORTD, 1, 1
48  bra otro3
49 otro4:
50  btfss PORTD, 1, 1
51  bra otro4
52  bcf LATC, 0, 1
53  bcf LATD, 0, 1
54  bcf LATF, 0, 1
55 otro5:
56  btfsc PORTD, 1, 1
57  bra otro5
58 otro6:
59  btfss PORTD, 1, 1
60  bra otro6
61  bcf LATC, 0, 1
62  bcf LATD, 0, 1
63  bcf LATF, 0, 1
64 otro7:
65  btfsc PORTD, 1, 1
66  bra otro7
67  bra inicio
68
69 end upcino
    
```

Ejemplo 2024-1

- Código:

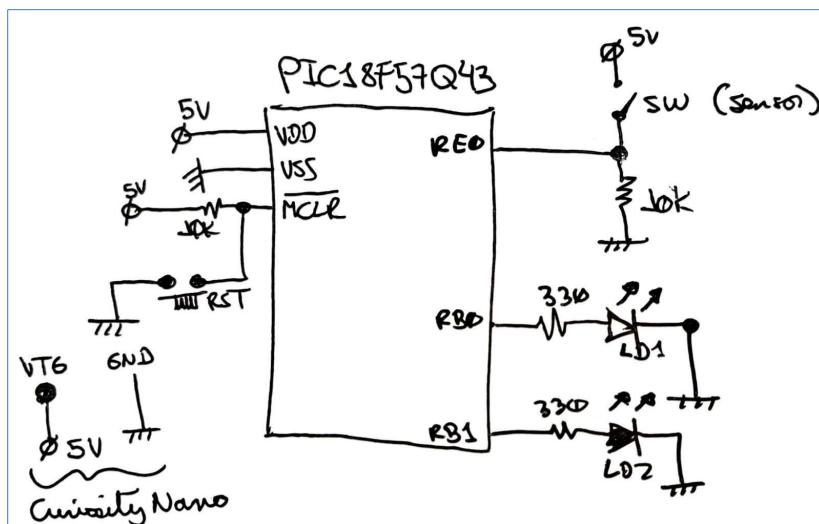
Ejemplo 2 – 2024-2

- Desarrollar una aplicación de una señal de cruce de tren con control de encendido a través de un pulsador.



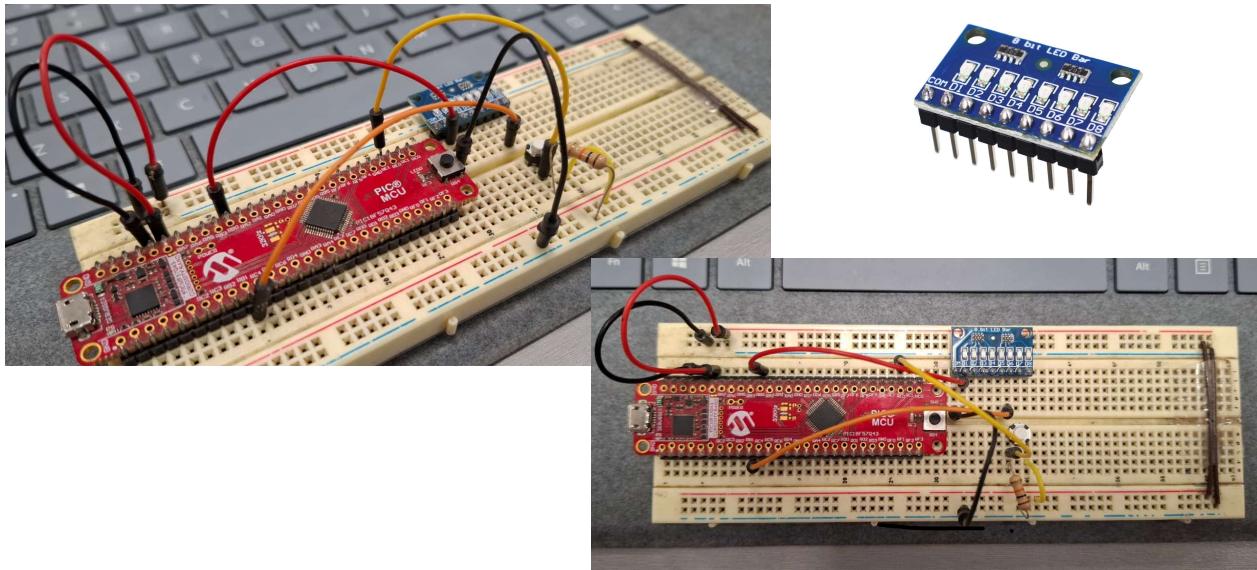
Ejemplo 2 – 2024-2

- Diseño del hardware – Diagrama esquemático



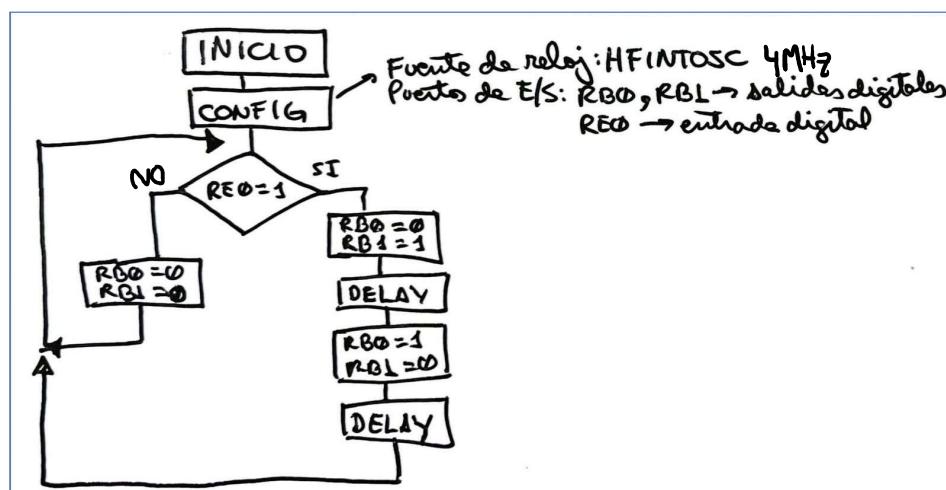
Ejemplo 2 – 2024-2

- Diseño del hardware – Implementación



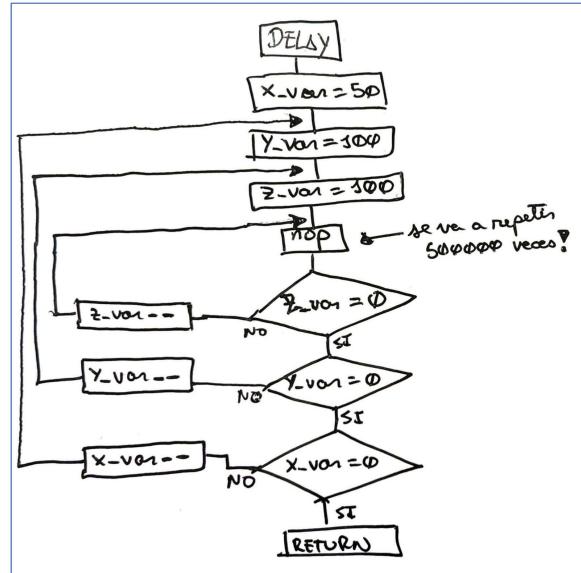
Ejemplo 2 – 2024-2

- Diseño de software: Diagrama de flujo



Ejemplo 2 – 2024-2

- Diseño de software: Diagrama de flujo



Ejemplo 2 – 2024-2

- Diseño de software: Codificación en lenguaje XC8 PIC Assembler

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  ;asignación de nombres a los registros GPR para la rutina de retardo
5  x_var EQU 500H           ;opcion 2 manipulacion numerica al registro
6  y_var EQU 501H           ;salto a label es_falso
7  z_var EQU 502H           ;Verdad (efecto de luces del cruce de tren)
8
9  PSECT upcino, class=CODE, reloc=2, abs
10 upcino:
11   ORG 000000H             ;vector de reset
12   bra configuro            ;salto a label configuro
13
14   ORG 000100H             ;zona de programa de usuario
15 configuro:
16   ;configuracion de la fuente de reloj
17   movlb 0H                 ;nos vamos al Bank0
18   movlw 60H
19   movwf OSCCON1, 1          ;NOSC=HFINTOSC, NDIV 1:1
20   movlw 02H
21   movwf OSCFRC, 1           ;HFINTOSC a 4MHz
22   movlw 40H
23   movwf OSCEN, 1             ;HFINTOSC habilitado
24   ;configuracion las E/S
25   movlb 4H
26   bcf TRISE, 0, 1           ;REO como entrada
27   bcf ANSELE, 0, 1           ;REO como digital
28   ;opcion 1 manipulacion individual de bits
29   bcf TRISB, 0, 1           ;RB0 como salida
30   bcf ANSELB, 0, 1           ;RB0 como digital
31   bcf ANSELB, 0, 1           ;RB1 como digital
32
33   ;opcion 2 manipulacion numerica al registro
34   movlw 0FCH
35   movwf TRISE, 1             ;RB0 y RB1 como salidas
36   movwf ANSELE, 1             ;RB0 y RB1 como digitales
37
38   inicio_train:
39   btfs PORTE, 0, 1           ;Pregunto si RE0 es uno
40   bra es_falso               ;Falso, salta a label es_falso
41   moviw 01H
42   movwf LATB, 1               ;RB1 apagado, RB0 encendido
43   call retardo               ;llamo rutina de retardo
44   moviw 02H
45   movwf LATB, 1               ;RB1 encendido, RB0 apagado
46   call retardo               ;llamo rutina de retardo
47   bra inicio_train           ;salta a label inicio_train
48
49   es_falso:
50   clrf LATB, 1               ;todo el RB en cero (apagado RB1 y RB0)
51   bra inicio_train           ;salta a label inicio_train
52
53   retardo:
54   movlb 5H                   ;nos vamos al Bank5
55   movlw 50
56   movwf x_var, 1
57
58   punto_2:
59   moviw 50
60   movwf y_var, 1
61
62   punto_1:
63   moviw 50
64   movwf z_var, 1
65
66   punto_0:
67   nop                         ;se va a repetir 125000 veces
68   movlw 0
69   cpfseq y_var, 1
70   bfa falso_y
71   movlw 0
72   cpfseq x_var, 1
73   bfa falso_x
74   movlb 4H                     ;regresamos al Bank0
75   return
76
77   falso_z:
78   decf z_var, 1, 1
79   bfa punto_0
80
81   falso_y:
82   decf y_var, 1, 1
83   bfa punto_1
84
85   falso_x:
86   decf x_var, 1, 1
87   bfa punto_2
88
89   end_upcino

```

Ejemplo 2 – 2024-2

- Diseño de software: Codificación en lenguaje XC8 PIC Assembler

```

1      PROCESSOR 18F57Q43          34    inicio_tren:
2      #include "cabecera.inc"     35        btfs PORTE, 0, 1 ;pregunto si RE0 es 1
3
4      ;Asignacion de labels a GPR 36        bra es_falsoz ;viene aqui cuando es Falso
5      x_var EQU 500H              37        bra es_verdad ;viene aqui cuando es Verdadero
6      y_var EQU 501H              38
7      z_var EQU 502H              39
8
9      PSECT upcino, class=CODE, reloc=2, abs 40        bsf LATB, 0, 1 ;enciendiendo RB0
10     upcino:                   41        bcf LATB, 1, 1 ;apago RB1
11         ORG 000000H ;vector de RESET 42        call retardo ;llamada a subrutina retardo
12         bra configuro ;salto a label configuro 43        bsf LATB, 0, 1 ;enciendiendo RB1
13
14         ORG 000100H ;zona de programa de usu 44        bcf LATB, 1, 1 ;apago RB0
15 configuro:                  45        call retardo ;llamada a subrutina retardo
16             ;configuracion la fuente de reloj 46        bra inicio_tren ;retorno al inicio_tren
17             movlb 0H ;nos vamos al Bank0 47
18             movlw 60H 48
19             movwf OSCCON1, 1 ;NOSC=HFINTOSC, NDIV 50        es_falsoz:
20             movlw 02H 51        bcf LATB, 0, 1 ;apago RB0
21             movwf OSCFRCQ, 1 ;HFINTOSC a 4MHz 52        bcf LATB, 1, 1 ;apago RB1
22             movlw 40H 53        bra inicio_tren ;retorno al inicio_tren
23             movwf OSCEN, 1 ;HFINTOSC habilitado 54
24             ;configuracion las E/S 55
25             movlb 4H ;saltamos al Bankd 56
26             bcf TRISE, 0, 1 ;RE0 es entrada 57        es_verdad:
27             bcf ANSELE, 0, 1 ;RE0 es digital 58        bcf LATB, 0, 1 ;enciendiendo RB0
28             bcf TRISB, 0, 1 ;RB0 es salida 59        bcf LATB, 1, 1 ;apago RB1
29             bcf TRISB, 1, 1 ;RB1 es salida 60        call retardo ;llamada a subrutina retardo
30             ;Recordando bcf [registro], #bit, access 61        bsf LATB, 0, 1 ;enciendiendo RB1
31             bcf ANSELB, 0, 1 ;RB0 es digital 62        bcf LATB, 1, 1 ;apago RB0
32             bcf ANSELB, 1, 1 ;RB1 es digital 63        call retardo ;llamada a subrutina retardo
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
766
767
768
768
769
769
770
771
772
773
774
775
776
776
777
778
778
779
779
780
781
782
783
784
785
786
786
787
788
788
789
789
790
791
792
793
794
795
795
796
797
797
798
798
799
799
800
801
802
803
804
805
805
806
807
807
808
808
809
809
810
811
812
813
814
815
815
816
817
817
818
818
819
819
820
821
822
823
824
824
825
826
826
827
827
828
828
829
829
830
831
832
833
834
834
835
836
836
837
837
838
838
839
839
840
841
842
843
843
844
845
845
846
846
847
847
848
848
849
849
850
851
852
853
853
854
855
855
856
856
857
857
858
858
859
859
860
861
862
863
863
864
865
865
866
866
867
867
868
868
869
869
870
871
872
873
873
874
875
875
876
876
877
877
878
878
879
879
880
881
882
883
883
884
885
885
886
886
887
887
888
888
889
889
890
891
892
893
893
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
903
904
905
905
906
906
907
907
908
908
909
909
910
911
912
913
913
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
923
924
925
925
926
926
927
927
928
928
929
929
930
931
932
933
933
934
935
935
936
936
937
937
938
938
939
939
940
941
942
943
943
944
945
945
946
946
947
947
948
948
949
949
950
951
952
953
953
954
955
955
956
956
957
957
958
958
959
959
960
961
962
963
963
964
965
965
966
966
967
967
968
968
969
969
970
971
972
973
973
974
975
975
976
976
977
977
978
978
979
979
980
981
982
983
983
984
985
985
986
986
987
987
988
988
989
989
990
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
```

Ejemplo 2 – 2024-2

- Diseño de software: Codificación en lenguaje XC8 PIC Assembler

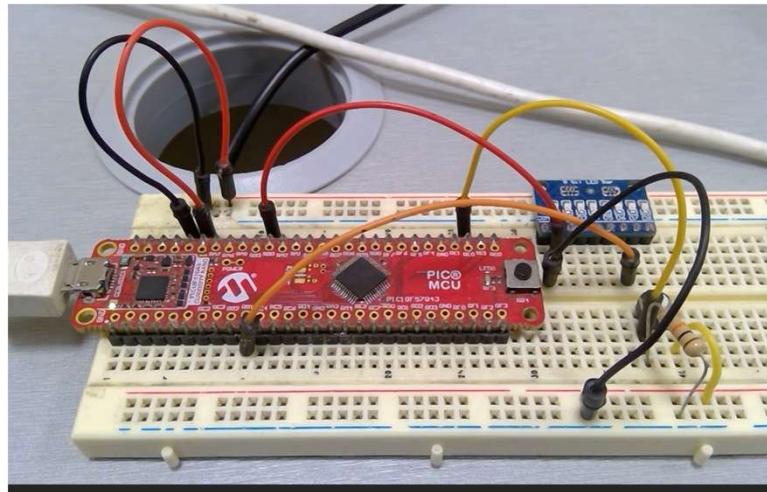
```

1  PROCESSOR 18F57Q43          ;RE0 com
2  #include "cabecera.inc"
3
4  x_var EQU 500H
5  y_var EQU 501H
6  z_var EQU 502H
7
8      PSECT upcino, class=CODE, reloc=2, abs
9  upcino:
10     ORG 000000H      ;vector de reset
11     bra configurando
12
13     ORG 000100H      ;zona de usuario
14 configurando:
15     ;configuracion de la fuente de reloj
16     movlw 0H           ;al Bank0
17     movlw 60H
18     movwf OSCCON1, 1   ;NOSC=HFINTOSC NDIV
19     movlw 02H
20     movwf OSCFRQ, 1    ;HFINTOSC 4MHz
21     movlw 40H
22     movwf OSCEN, 1     ;HFINTOSC enabled
23     ;configuracion de las E/S
24     movlb 4H           ;al Bank4
25     movlw 0xF0          ;en binario 1111110
26     movwf TRISB, 1      ;RB0 y RB1 como sal
27     movwf ANSELB, 1     ;RB0 y RB1 como dig
28     bsf TRISE, 0         ;RE0 como entrada
29
30
31 inicio_tren:
32     btfss PORTE, 0, 1  ;pregunta
33     bra es_falso        ;viene a
34     bra es_verdad       ;viene a
35 es_falso:
36     clrf LATB, 1        ;apagando
37     bra inicio_tren
38 es_verdad:
39     movlw 01H
40     movwf LATB, 1        ;RB1=0, 69
41     call retardado       ;llamada
42     movlw 02H
43     movwf LATB, 1        ;RB1=1, 72
44     call retardado       ;llamada
45     bra inicio_tren
46 retardado:
47     movlb 5H             ;al Bank
48     movlw 30
49     movwf x_var, 1
50 punto_tres:
51     movlw 30
52     movwf y_var, 1
53 punto_dos:
54     movlw 30
55     movwf z_var, 1
56 punto_uno:
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470

```

Ejemplo 2 – 2024-2

- Diseño de software: Funcionamiento, al presionar el botón iniciará la secuencia de encendido de los LEDs



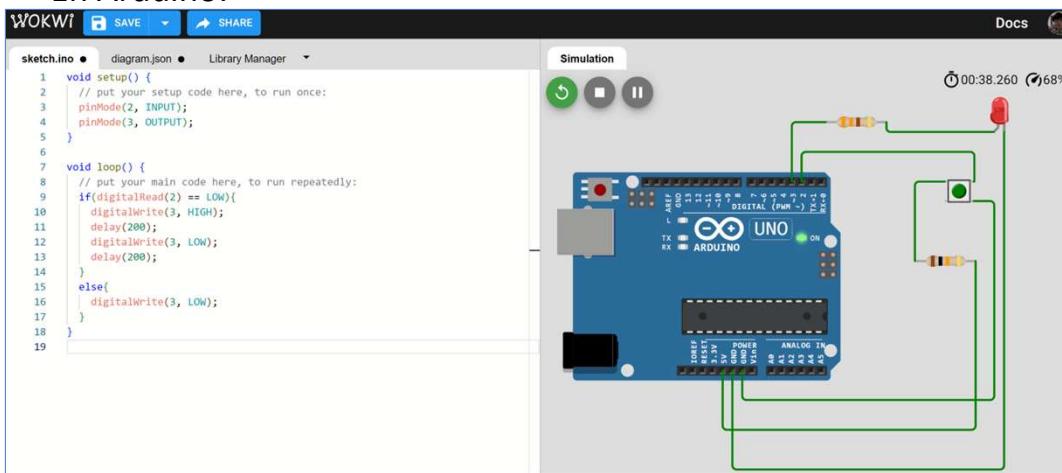
Ejemplo 2 – 2024-2

- Asignación: Mejorar el diseño agregando una entrada para que se puede manipular la velocidad de la alternancia de encendido de los LEDs

Ejemplo 2 (2025-1)

- Prender y apagar un LED, controlando su parpadeo con un pulsador activo en bajo, el parpadeo debe de activarse solo cuando se mantenga presionado el pulsador.

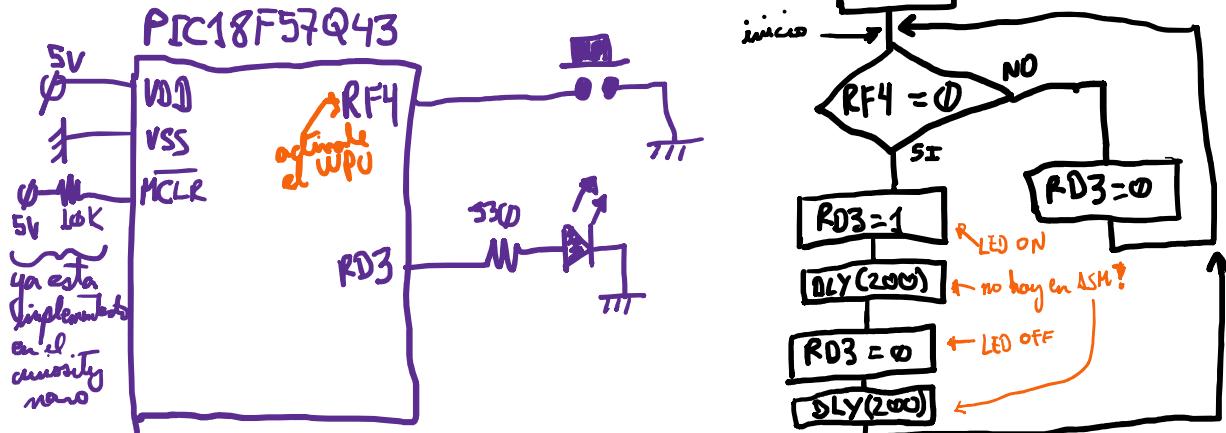
En Arduino:



Ejemplo 2 (2025-1)

- Prender y apagar un LED, controlando su parpadeo con un pulsador activo en bajo, el parpadeo debe de activarse solo cuando se mantenga presionado el pulsador.

Empleando el PIC18F57Q43:



Ejemplo 2 (2025-1)

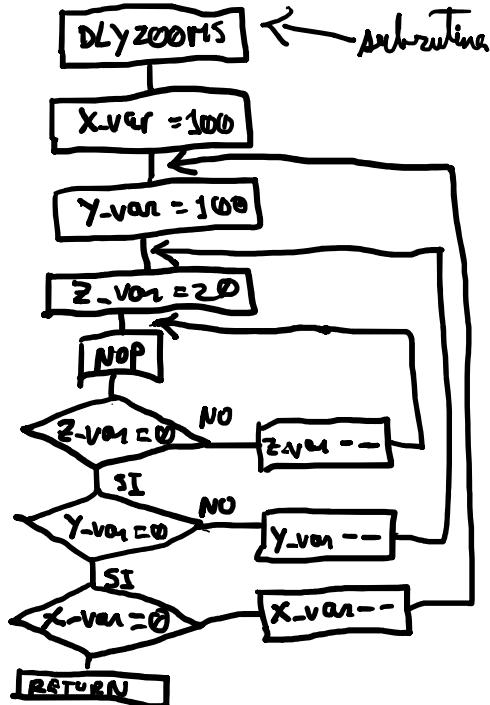
- Tenemos un pequeño inconveniente: en assembler no hay rutina de retardo implementado.

Se tiene la instrucción "nop"

Si Fosc = 4MHz ese nop se ejecutará en 1us.

Tu necesitas 200000us, 200000 nops?

Tenemos que construir una estructura de repetición anidada para ese nop

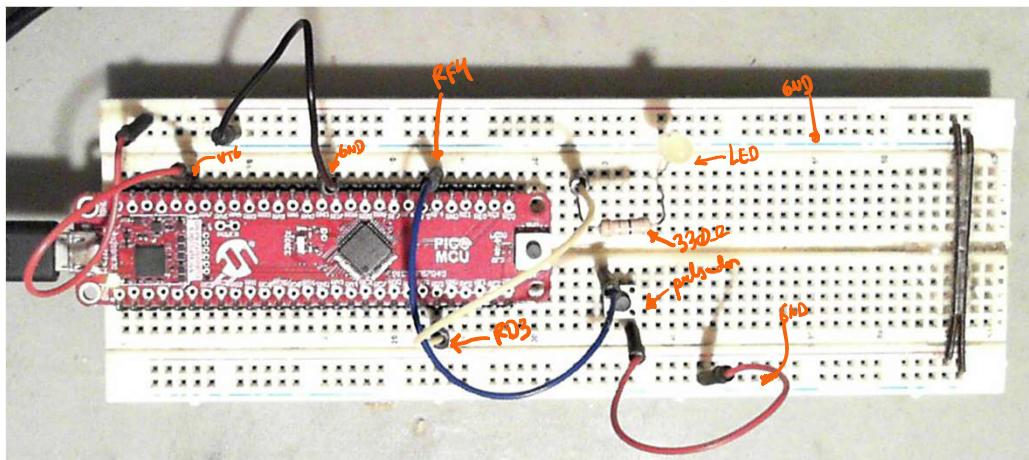


Ejemplo 2 (2025-1)

- Prender y apagar un LED, controlando su parpadeo con un pulsador activo en bajo, el parpadeo debe de activarse solo cuando se mantenga presionado el pulsador.

<pre> 1 ;Programa del ejemplo de la semana 2 - 2025-1 2 PROCESSOR 18F57Q43 3 #include "cabecera.inc" 4 5 PSECT upcino, class=CODE, reloc=2, abs 6 ;declarar etiquetas para los GPR 7 x_var EQU 500H 8 y_var EQU 501H 9 z_var EQU 502H 10 11 upcino: 12 ORG 000000H 13 bra configuro 14 15 ORG 000100H 16 configuro: 17 ;Configuracion del modulo de oscilador 18 movlb 0H 19 movlw 60H 20 movwf OSCCON1, 1 ;NOSC=HFINTOSC, NDIV 21 movlw 02H 22 movwf OSCFRO, 1 ;HFINTOSC a 4MHz 23 bsf OSCEN, 6, 1 ;HFINTOSC enabled 24 25 ;Configuracion de los puertos de E/S 26 movlb 4H 27 ;RF4 como entrada digital con pullup activado 28 bsf TRISF, 4, 1 ;RF4 entrada 29 bcf ANSELF, 4, 1 ;RF4 digital 30 bcf WPFF, 4, 1 ;RF4 pullup enabled 31 ;RD3 como salida digital 32 bcf TRI3D, 3, 1 ;RD3 salida 33 bcf ANSELD, 3, 1 ;RD3 digital </pre>	<pre> 35 inicio: 36 btfs PORTF, 4, 1 ;pregunto si presione el botón 37 bra sopresione ;falso, salta a noperse 38 bra siperseone ;verdad, salta a siperse 39 40 noperse: 41 bcf LATD, 3, 1 ;apago el LED 42 bra inicio ;salto de retorno a inicio 43 44 siperse: 45 bsf LATD, 3, 1 ;enciendiendo el LED 46 call dly200ms ;salto a subrutina dly200ms 47 bcf LATD, 3, 1 ;apago el LED 48 call dly200ms ;salto a subrutina dly200ms 49 bra inicio ;salto de retorno a inicio 50 51 ;subrutina de retardo 52 dly200ms: 53 movlb 5H 54 movlw 50 55 movwf x_var, 1 56 llegada_uno: 57 movlw 50 58 movwf y_var, 1 59 llegada_dos: 60 movlw 20 61 movwf z_var, 1 </pre>	<pre> 62 llegada_tres: 63 nop 64 ;pregunta si z_var llegó a cero 65 movf z_var, 1, 1 66 movlb 4H 67 btfs STATUS, 2, 1 ;pregunta si se levanto bandera Z 68 bra noselevantol 69 movlb 5H 70 ;pregunta si y_var llegó a cero 71 movf y_var, 1, 1 72 movlb 4H 73 btfs STATUS, 2, 1 ;pregunta si se levanto bandera Y 74 bra noselevantol2 75 movlb 5H 76 ;pregunta si x_var llegó a cero 77 movf x_var, 1, 1 78 movlb 4H 79 btfs STATUS, 2, 1 ;pregunta si se levanto bandera X 80 bra noselevantol3 81 82 noselevantol: 83 movlb 5H 84 decf z_var, 1, 1 ;decremento de z_var 85 bra llegada_tres 86 noselevantol2: 87 movlb 5H 88 decf y_var, 1, 1 ;decremento de y_var 89 bra llegada_dos 90 noselevantol3: 91 movlb 5H 92 decf x_var, 1, 1 ;decremento de x_var 93 bra llegada_uno 94 95 end upcino </pre>
--	--	---

Ejemplo 2 (2025-1)



Ejercicios complementarios de manipulación de puertos con el PIC18F57Q43:

Tener en consideración el procedimiento para el desarrollo de los ejercicios (análisis del problema y requerimientos, diseño de hardware, diagrama de flujo, código en XC8 PIC Assembler y pruebas)

1. Colocar LEDs en todos los pines de RD y de RB (con su respectiva resistencia de 330Ω). Escribir 5AH en RD y 0A5H en RB.
2. Implementar una XOR de un bit empleando RD0 y RD4 como entradas y RB2 como la salida.
3. Recibir un dato de 8 bits en RD y replicarlo en complemento por RB
4. Con el algoritmo antirrebote basado en la generación de retardo de 40ms con bucles anidados visto en el último ejemplo 04. Ampliar dicho retardo hasta 500ms aproximadamente y realizar una aplicación de parpadeo de un LED conectado en el RA0.

Recomendaciones al momento de implementar el circuito en físico con el Curiosity Nano PIC18F57Q43 y el Curiosity Nano PIC18F47Q10:

- Verificar continuidad en los cables jumper antes de ser utilizados en el circuito.
- Verificar que el cable USB-microUSB tenga capacidad de transferencia de datos.
- Verificar que la PC haya detectado correctamente el Curiosity Nano PIC18F57Q43 o el Curiosity Nano PIC18F47Q10.
- Verificar que el proyecto creado en el MPLABX se haya seleccionado el Curiosity Nano PIC18F57Q43 o el Curiosity Nano PIC18F47Q10 (ventana de propiedades del proyecto y "connected hardware tool")
- No olvidar que el header file tiene extensión *.inc y el source file tiene extensión *.s
- Revisar que se haya configurado el Curiosity Nano PIC18F57Q43 o Curiosity Nano PIC18F47Q10 para que entregue voltaje de 5V a través del pin VTG (ventana de propiedades del PKOB nano dentro de propiedades del proyecto y opción Power)
- Revisar siempre los mensajes en la ventana de "output" por posibles fallos en el evento de compilación y/o evento de programación.
- Tener a la mano un multímetro para verificar voltajes y continuidad de conexiones en el prototipo implementado

Ejercicio 2026-0:

- Al ejercicio del auto fantástico agregarle tres efectos luminosos adicionales (total cuatro), el cambio se hará mediante un solo pulsador ubicado en RB0.

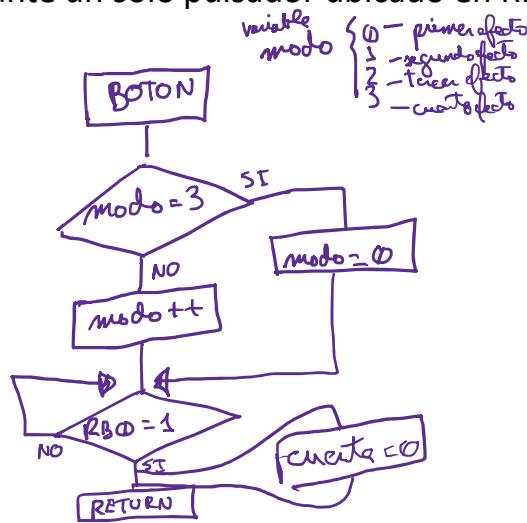
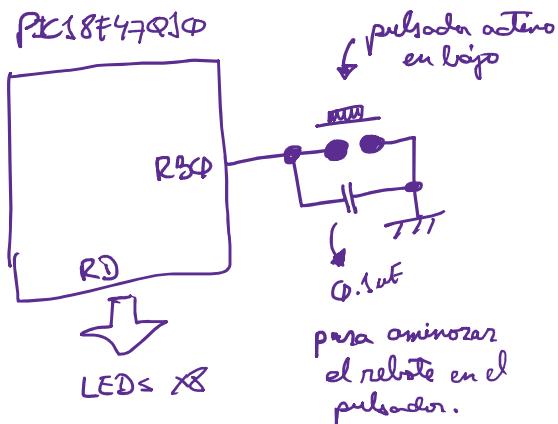
	L0	L1	L2	L3	L4	L5	L6	L7	
T0									03H
T1									06H
T2									0CH
T3									18H
T4									30H
T5									60H
T6									0COH
T7									60H
T8									30H
T9									18H
T10									0CH
T11									06H

	L0	L1	L2	L3	L4	L5	L6	L7	
T0									07H
T1									0EH
T2									1CH
T3									38H
T4									70H
T5									0E0H
T6									70H
T7									38H
T8									1CH
T9									0EH

	L0	L1	L2	L3	L4	L5	L6	L7	
T0									81H
T1									42H
T2									24H
T3									18H
T4									24H
T5									42H

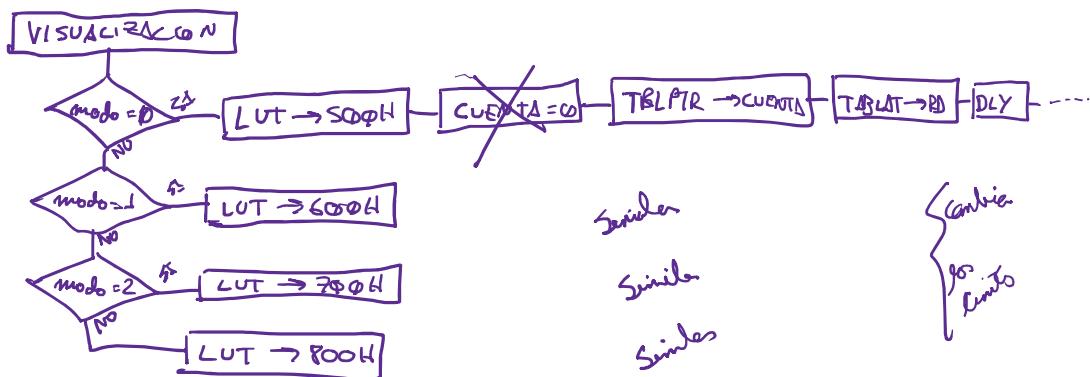
Ejercicio 2026-0:

- Al ejercicio del auto fantástico agregarle tres efectos luminosos adicionales (total cuatro), el cambio se hará mediante un solo pulsador ubicado en RBO.
- Funcionamiento del pulsador en RBO:



Ejercicio 2026-0:

- Al ejercicio del auto fantástico agregarle tres efectos luminosos adicionales (total cuatro), el cambio se hará mediante un solo pulsador ubicado en RBO.
- Selección de la visualización



Ejercicio adicional 2026-0

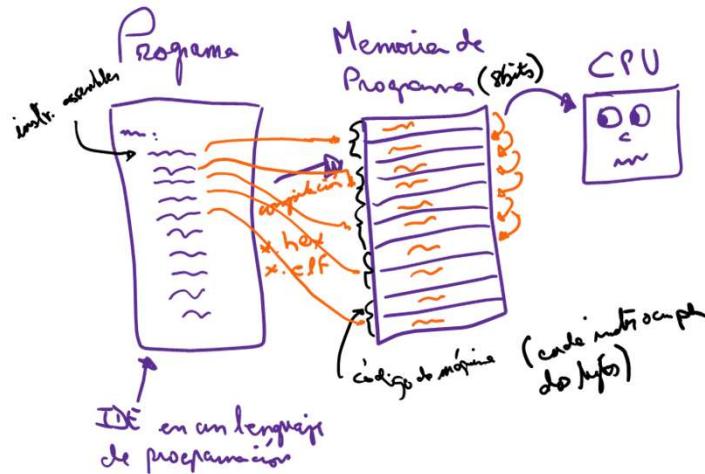
- Agregar un pulsador adicional para cambiar la velocidad de lento a rápido.

Agenda

- El contador de programa
- Los displays de siete segmentos
- Datos constantes en la memoria de programa
- El puntero de tabla (TBLPTR)
- Instrucciones de comparación numérica

Recordando:

- La ejecución de un programa en el microcontrolador es de manera **secuencial** (una instrucción a la vez) y **ordenada** (uno detrás de otro).



La ejecución es ordenada y secuencial

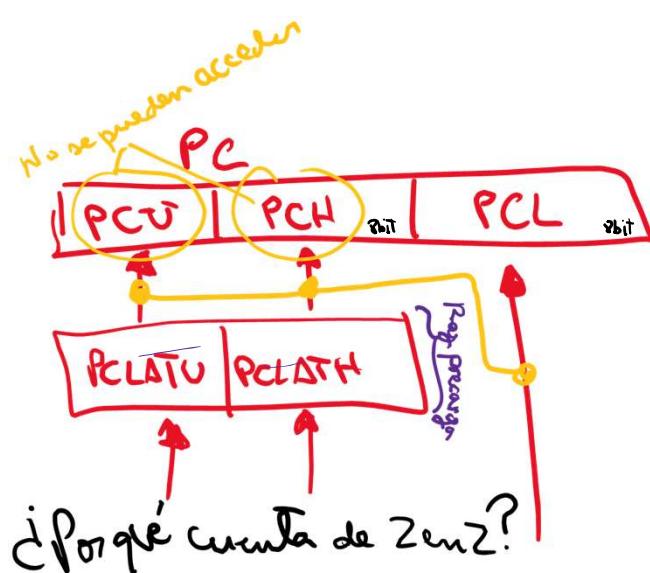
```

inicio:          ;etiqueta llamada inicio
    f°→ bsf LATF, 6, 1 ;RF6 en uno lógico
    2°→ nop            ;retardo de 1us si Fosc=4MHz
    3°→ bcf LATF, 6, 1 ;RF6 en cero lógico
    4°→ nop            ;retardo de 1us si Fosc=4MHz
    5°→ bra inicio     ;salto a etiqueta inicio
  
```

Cuando el programa anterior se pasa a la memoria de programa del microcontrolador

Memoria de programa	
Contenido	Dirección
bsf LATF, 6, 1 <i>(4CH)</i>	000000H
nop	000001H
	000002H
	000003H
bcf LATF, 6, 1 <i>(4CH)</i>	000004H
nop	000005H
	000006H
	000007H
bra inicio <i>(000000H)</i>	000008H
	000009H
	00000AH
	00000BH

El Contador de Programa (PC)



- Indispensable para la ejecución secuencial de las instrucciones.
- Su función es de alojar la dirección de la siguiente instrucción que el CPU va a ejecutar.
- Según hoja técnica, consta de 21 bits separados en tres registros: PCU, PCH y PCL.
- PCU y PCH no son accesibles, para que se pueda escribir la dirección de 21bits es necesario seguir paso previo que es la de precargar los datos en los registros previos PCLATH y PCLATU, y solamente se transferirán hacia PCU y PCH respectivamente cuando PCL le cargue un dato.
- Al escribir en PCL se subirán PCLATH y PCLATU para así subir los 21 bits a la vez
- En el PIC18F57Q43, se tiene solamente 17 bits ya que la memoria es de 128Kbyte

NOTA: El banco activo debe de ser el 4

Ejemplo

- Cargar dirección 00288AH en PC:

```

    clrf PCLATH      ← Precarga
    movlw 28H
    movwf PCLATH      ← Precarga
    movlw 8AH
    movwf PCL          ← Se carga PCL
    PCLATH            hacia el PC
  
```



9.8 Register Summary - Memory Organization

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x04D9	FSR2	7:0								FSRH[5:0]
0x04DB	PLUSW2	15:8								PLUSW[7:0]
0x04DC	PREINC2	7:0								PREINC[7:0]
0x04DD	POSTDEC2	7:0								POSTDEC[7:0]
0x04DE	POSTINC2	7:0								POSTINC[7:0]
0x04DF	INDF2	7:0								INDF[7:0]
0x04E0	BSR	7:0								BSR[5:0]
0x04E1	FSR1	15:8								FSRH[5:0]
0x04E3	PLUSW1	7:0								PLUSW[7:0]
0x04E4	PREINC1	7:0								PREINC[7:0]
0x04E5	POSTDEC1	7:0								POSTDEC[7:0]
0x04E6	POSTINC1	7:0								POSTINC[7:0]
0x04E7	INDF1	7:0								INDF[7:0]
0x04E8	WREG	7:0								WREG[7:0]
0x04E9	FSR0	15:8								FSRH[5:0]
0x04EB	PLUSW0	7:0								PLUSW[7:0]
0x04EC	PREINC0	7:0								PREINC[7:0]
0x04ED	POSTDEC0	7:0								POSTDEC[7:0]
0x04EE	POSTINC0	7:0								POSTINC[7:0]
0x04EF	INDF0	7:0								INDF[7:0]
0x04F0	Reserved									
0x04F8	PCL	7:0								PCL[7:0]
0x04FA	PCLAT	15:8								PCLATH[7:0]
0x04FC	STKPTR	7:0								STKPTR[8:0]
0x04FD	TOS	7:0								TOS[7:0]
		23:16								TOS[16:8]
										TOS[20:16]

El mismo código de abajo pero sin etiquetas:

Ejemplo

- Si ejecutamos "goto inicio" en el siguiente programa:

```

13      org 0x0000
14      goto init_conf
15
16      org 0x0020
17      init_conf: clrf TRISD
18      inicio:    comf PORTB, W
19      movwf LATD
20      goto inicio
21      end
  
```

↓ hace un salto a 0x0020

↓ haz un salto a 0x0022

modifica el PC

Direcciones de la memoria de programación

```

org 0x0000
goto 0x0020
org 0x0020
clrf 0xF95
org 0x0022
comf 0xF81, 0
movwf 0xF8C
goto 0x0022
end
  
```

¿Cómo funciona el PC?

- Un programa simple:

```

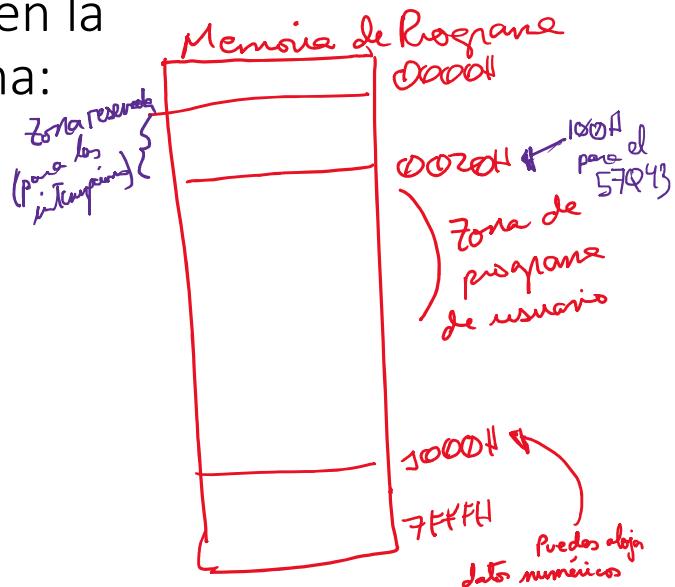
org 0x0000
inicio: clrf TRISB
loop:   setf LATB
        nop
        clrf LATB
        nop
        goto loop
end
    
```

Nota: PC va de dos en dos, nunca será impar



Información alojada en la memoria de programa:

- En la memoria de programa podemos alojar no solamente instrucciones de un programa, sino también datos que serán constantes (no se podrán modificar cuando el microcontrolador entre en operación)



Otro ejemplo de PC:

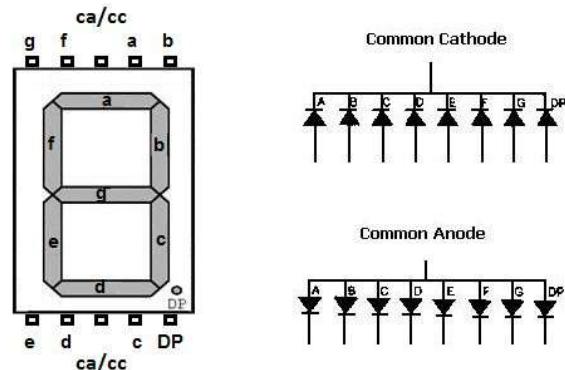
- Las últimas 4 instrucciones emulan un salto a una posición de memoria, como si fuera una instrucción *bra* o *goto*

bra 86H {

ORG 000000H	
bra 000080H	;00H
ORG 000080H	
movlb 4H	;80H
bcf TRISA, 0, 1	;82H
bcf ANSELA, 0, 1	;84H
btg LATA, 0, 1	;86H
nop	;88H
clrf PCLATU	;
clrf PCLATH	;
movlw 86H	;
movwf PCL	; hace un salto a 86H

El display de siete segmentos

- Dos tipos: ánodo común y cátodo común
- Tablas de decodificación diferentes entre ellos.
- Es necesario colocarle resistencias limitadoras de corriente para cada segmento que compone el display (100ohm a 1kohm).
- Evitar usar solo una resistencia en el pin común del display ya que al cambiar de carácter mostrado se tendrá un efecto de cambio de intensidad luminosa.

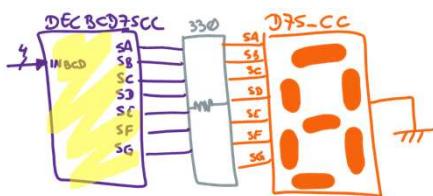


Los precursores a los displays de siete segmentos: Tubos Nixie

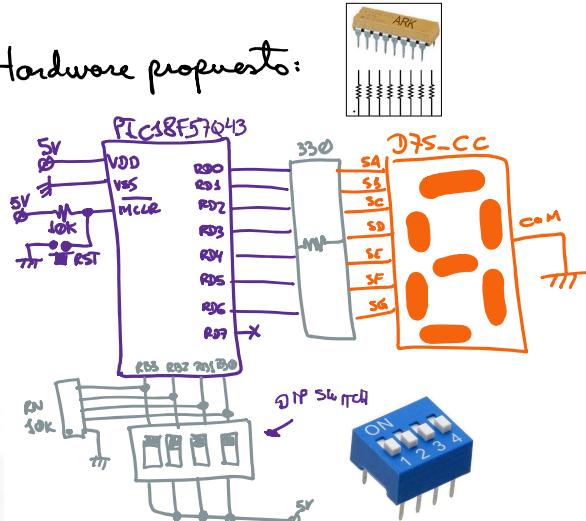


Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:

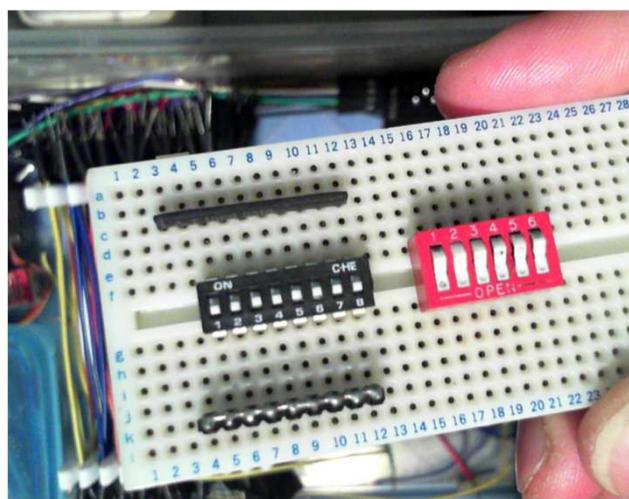


Hardware propuesto:



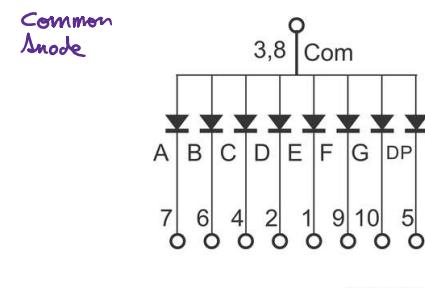
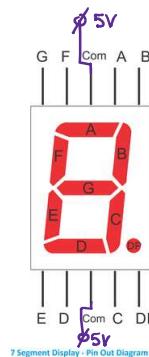
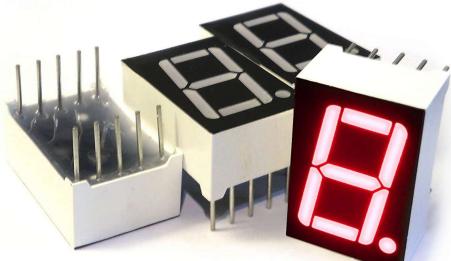
Uso de DIP Switches y Chip Resistors

- Nos ayudan a hacer el proceso de prototipaje mas rápido, ordenado y simple



Observación:

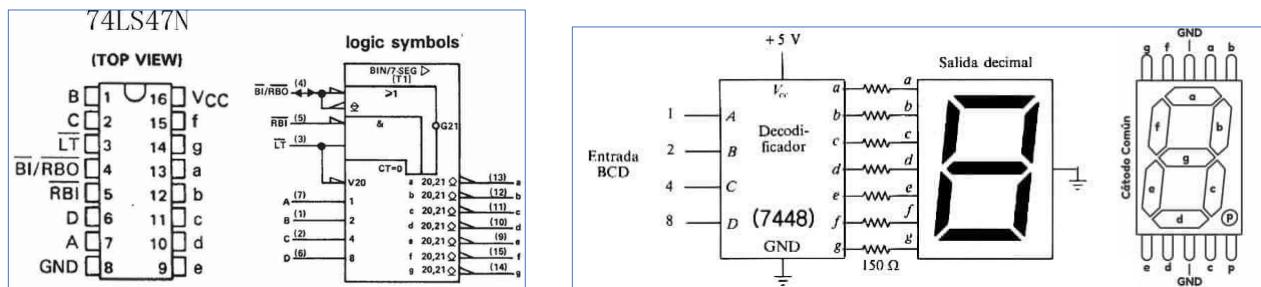
- Los pines denominados “comunes” (Com) son el mismo nodo, es la misma conexión!



Desarrollo de la tabla de decodificación BCD para display de 7 segmentos cátodo común:

CC	X	SG	SF	SE	SD	SC	SB	SA	HEX
	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	0	1	1	1	0x67

Circuitos TTL decodificadores de siete segmentos

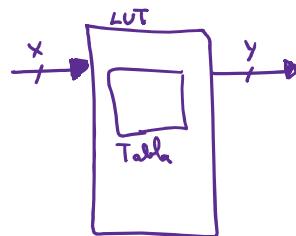


Recordando acerca de la memoria de programa

- En esta memoria no solamente permite el alojamiento de instrucciones.
- También se puede alojar datos constantes, estos datos constantes solo se graban en un evento de programación del microcontrolador, luego en operación no se pueden modificar ni borrar.

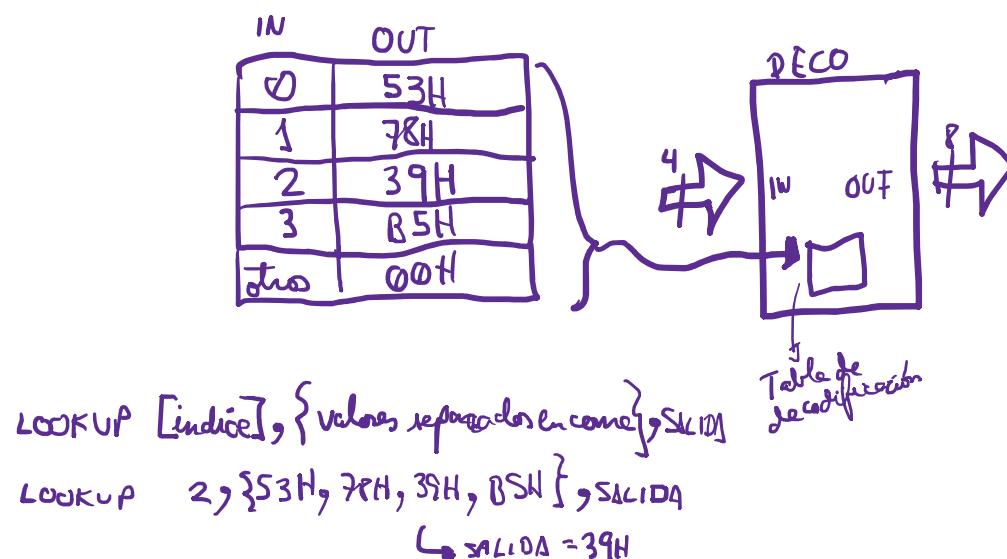
Tablas de búsqueda (lookup tables - LUT)

- Decodificadores implementados en software



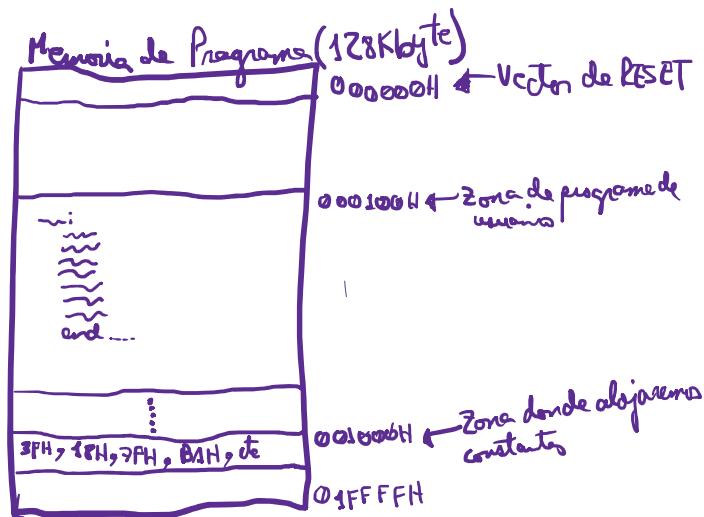
- Dos maneras de implementar LUT en MPASM-PIC18
 - Utilizando la funcionalidad del PC (preferible no usar)
 - Utilizando el TBLPTR

¿Cómo funciona la tabla de búsqueda?



¿En dónde guardo los datos de la tabla de decodificación?

- En la memoria de programa, alejado de la zona donde está tu programa:



¿Cómo funciona la tabla de búsqueda?

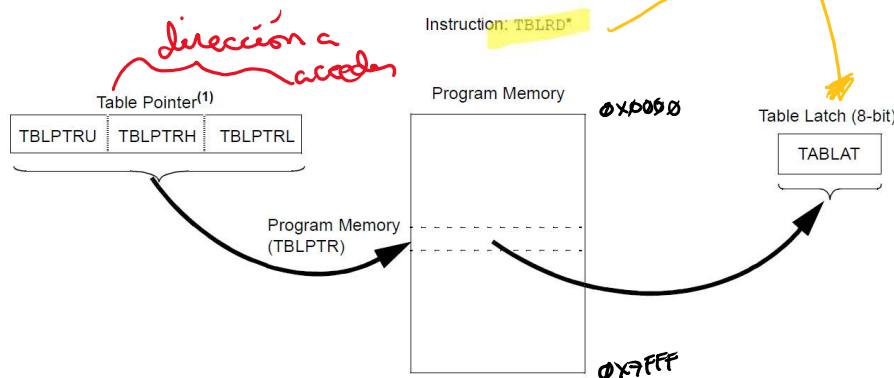
```

//índice: 0   1   2   3   4   5   6
unsigned char tablota[] = {0x10, 0x35, 0x44, 0x55, 0xBE, 0xAF, 0x99};
unsigned char salida = 0;

void main(void){
    salida = tablota[6];           //al ejecutarse salida=99H
}

```

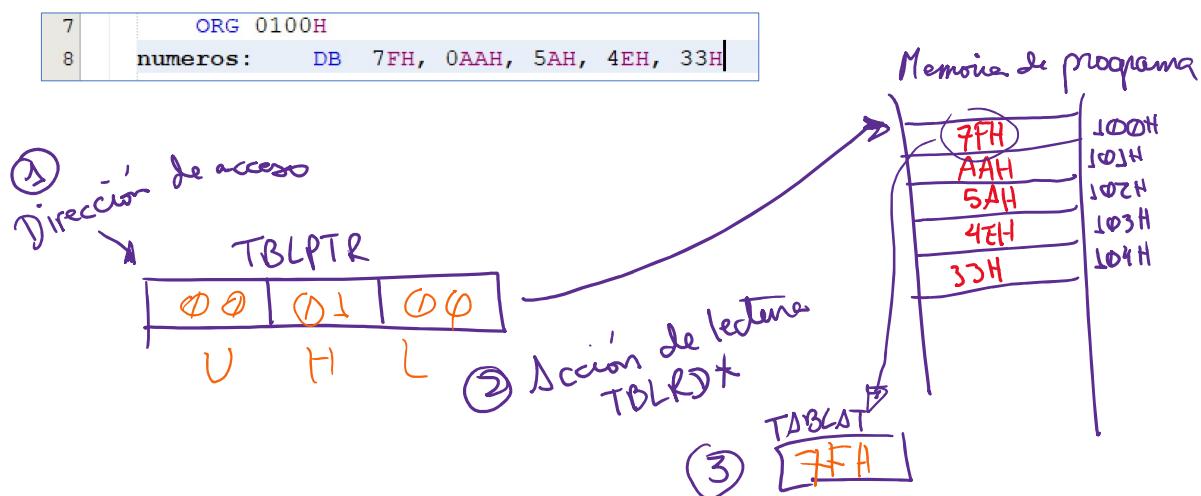
El puntero de tabla (TBLPTR)



Note 1: Table Pointer register points to a byte in program memory.

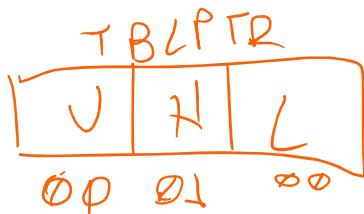
- Al igual que el PC, el TBLPTR también es de 21 bits (para el caso del PIC18F4550 15 bits)
- Se emplea como única herramienta para acceder a la memoria de programa y leer (también escribir pero es mas complicado) su contenido están en operación el microcontrolador.
- El puntero debe de tener la dirección de apunte antes de hacer el proceso de lectura con TBLRD*. Luego de la acción de lectura, el contenido de la celda apuntada se alojará en el registro TABLAT

Operación con el TBLPTR



Operación con el TBLPTR

- En este programa se busca obtener el contenido de 102H en la memoria y trasladarlo al puerto D



```

1      PROCESSOR 18F4550
2      #include "cabecera.inc"
3
4      PSECT principal, class=CODE, reloc=2, abs
5
6      principal:
7          ORG 0100H
8          numeros:   DB 7FH, 0AAH, 5AH, 4EH, 33H
9
10         ORG 0000H
11         goto configuracion
12         ORG 0020H
13
14         configuracion:
15             clrf TRISD      ;Puerto D como salida
16             clrf TBLPTRU
17             movlw HIGH numeros
18             movwf TBLPTRH
19             movlw LOW numeros
20             movwf TBLPTRL    ;TBLPTR apuntando a 000100H
21
22         loop:
23             movlw 02H
24             addwf TBLPTRL, 1    ;Dirección de apunte modificada a 000102H
25             TBLRD*              ;Leo contenido apuntado por TBLPTR
26             movf TABLAT, 0        ;Muevo el contenido de TABLAT hacia WReg
27             movwf LATD            ;Muevo contenido de WReg hacia LATD
28             goto loop
29         end principal

```

Modificación del ejemplo del decodificador anterior empleando TBLPTR

```

1      ...
2      #include "cabecera.inc"
3
4      PSECT rstVect, class=CODE, reloc=2, abs
5
6          ORG 0500H
7          cadena: DB 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x79, 0x79, 0x79, 0x79, 0x79
8
9          ORG 0000H
10         rstVect:   goto configuracion
11
12         ORG 0020H
13         configuracion: movlw 80H
14             movwf TRISD      ;RD6:RD0 como salidas
15             clrf TBLPTRU
16             movlw HIGH cadena
17             movwf TBLPTRH
18             movlw LOW cadena
19             movwf TBLPTRL      ;Asignamos dirección a TBLPTR (500H)
20         inicio:
21             movf PORTB, w
22             andlw 0FH
23             movwf TBLPTRL
24             TBLRD*
25             movff TABLAT, LATD
26             goto inicio
27
28         end rstVect

```

- Como segunda opción para implementar una LUT es empleando el puntero de tabla (TBLPTR) donde accederá a determinado dato ubicado en la memoria de programa dependiendo de la dirección asignada.

Ejemplo (PIC18F57Q43):

- Desarrollar un programa donde se tenga almacenado los siguientes datos en la **memoria de programa**:
 - 00500H: 04H
 - 00501H: AFH
 - 00502H: BEH
 - 00503H: 89H
- Elaborar un algoritmo que permita leer los datos anteriores y arrojarlas de manera secuencial a través de RD con periodo de NOP

Desarrollo del ejemplo:

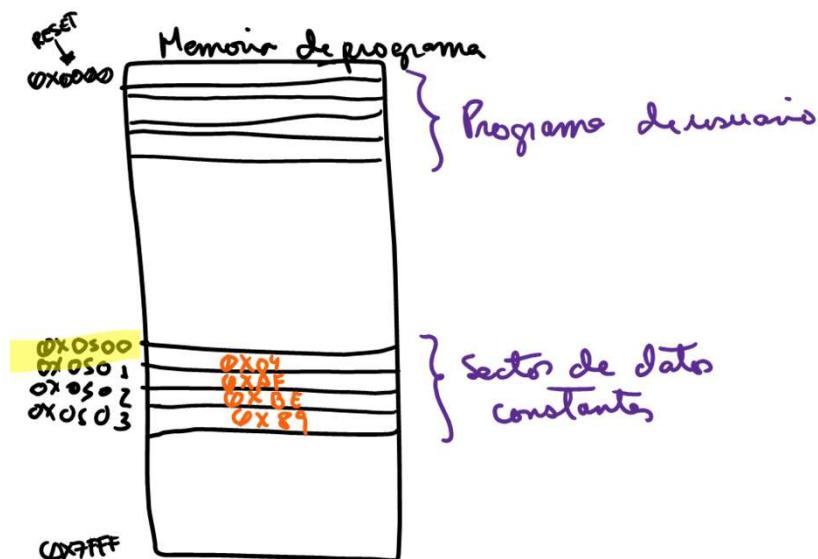
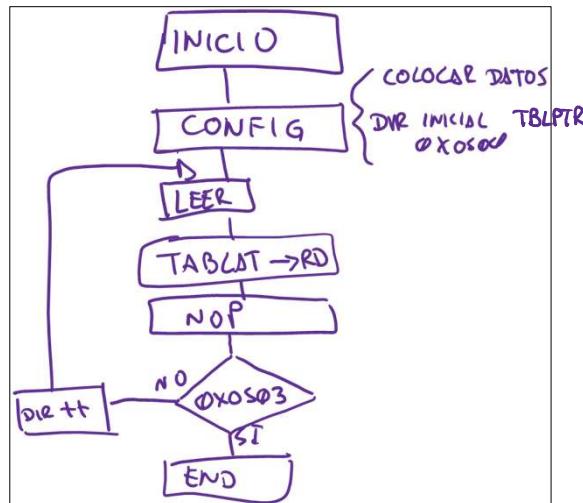


Diagrama de flujo:



Resumen de instrucciones con toma de decisión:

Formato:

Instrucción [registro]

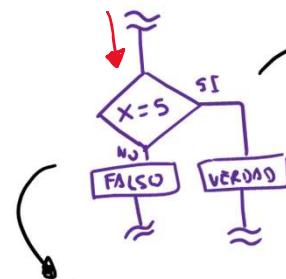
goto nnn

- | | |
|--------------------------|---|
| • btfss, btfsc | -Prueba el #bit de [registro] si es 0 ó 1 |
| • decfsz, incfsz | -Decrementa o incrementa [registro] y pregunta si es cero. |
| • dcfsnz, icfsnz | -Decrementa o incrementa [registro] y pregunta si no es cero |
| • cpfsgt, cpfseq, cpfslt | -Comparaciones numéricas entre [registro] y W |
| • tstfsz | -Prueba [registro] y salta si es cero |
| • Saltos Branch | -Saltos condicionales |

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

$\text{CPFSEQ} \rightarrow f = W_{\text{reg}}$
 $\text{CPFS LT} \rightarrow f < W_{\text{reg}}$
 $\text{CPFS GT} \rightarrow f > W_{\text{reg}}$

Nota: El valor a comparar debe de estar previamente en Wreg



En lenguaje de alto nivel:

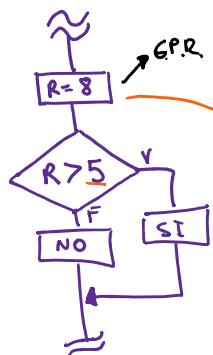
```

if (x = 5) {
    [VERDAD]
} else {
    [FALSO]
}
  
```

XC8 ASM:
 En MPASM:
 Usaremos CPFSEQ:
 CPFSEQ [reg]

$(f) = w$
 $x = 5$
 movlw 5
 cpfseq x-reg
 goto F
 goto Falso
 goto Verdad

Ejercicio: Pasar a XC8 ASM el siguiente diagrama de flujo:



$R_reg \text{ EQU } 0x0000$
 :
 movlw d'8'
 movwf R.reg
 movlw d'5'
 cpfsge R.reg
 goto NO
 goto SI

Comparando si:
 $R_reg > 5$

Nota: En las instrucciones cpfs-- el segundo parámetro de la comparación debe de estar en Wreg

Ejercicio:

- Implementar un comparador de magnitud de dos números de 8 bits:

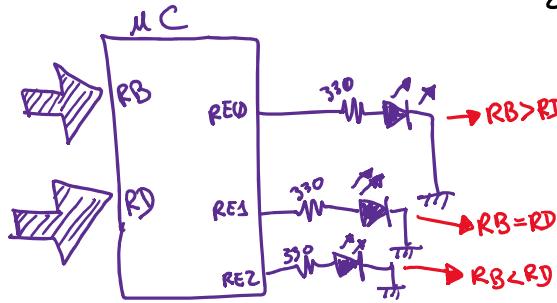
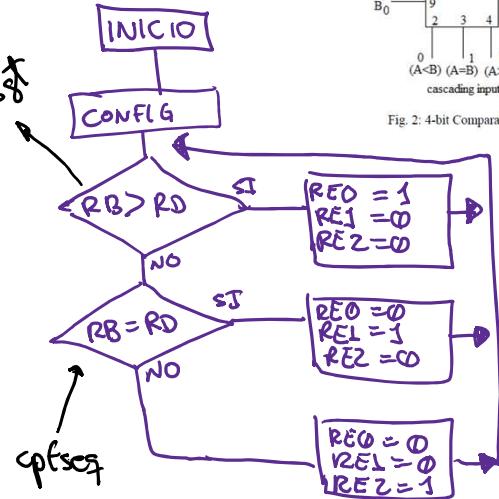
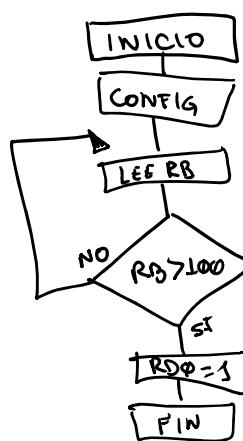
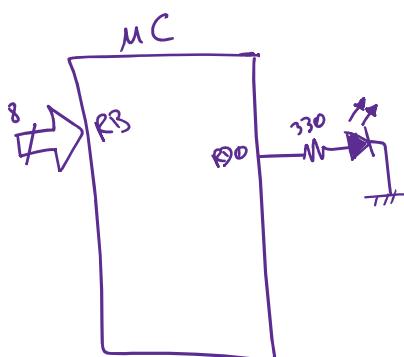


Fig. 2: 4-bit Comparator



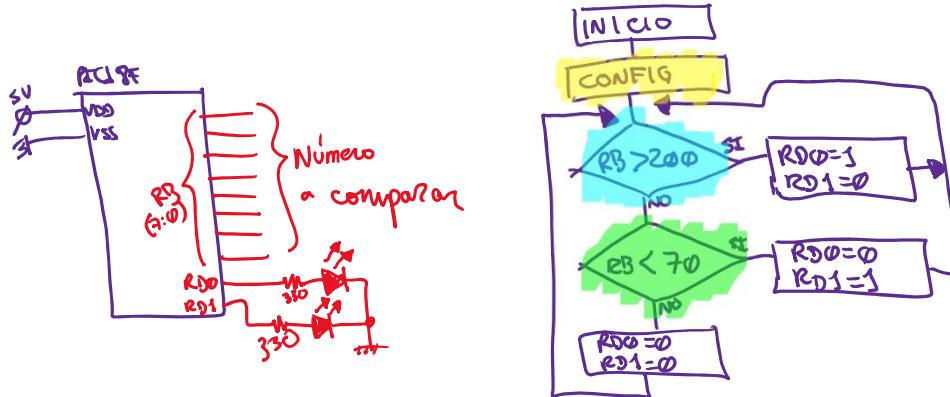
Ejercicio: Leer el valor de RB y colocar a uno el puerto RD0 únicamente cuando dicho valor sea mayor de 100.



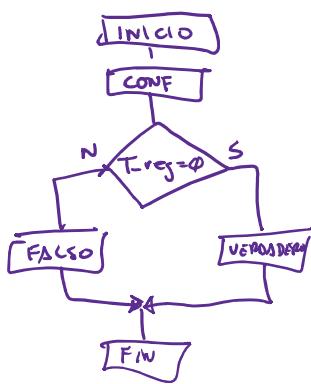
$f > w_{reg}$
 bucle:
 movlw .100
 cpfsgt PORTB
 goto bucle
 bsf LATD, 0

Ejercicio:

- Desarrollar un programa para que compare lo que se está ingresando en RB y arroje lo siguiente: RD0=1 cuando RB>200 y RD1=1 cuando RB<70, cuando no se cumplan las dos condiciones las dos salidas permanecerán en cero.



Ejemplo: Pregunta si T_reg (GPR) es igual a cero, si es cierto va a etiqueta verdadero, si no es cierto va a etiqueta falso



Opción 1:

inicio: movlw .0
cpfsg T-reg
v goto FALSO
d goto VERDADERO

Opción 2:

inicio: tstfsz T-reg
v goto FALSO
d goto VERDADERO

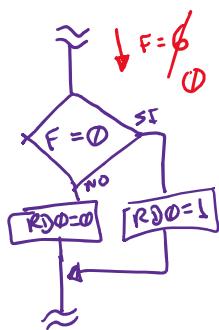
Opción 3:

inicio: movf T-reg, W
sublw .0
btfs STATUS, Z
v goto FALSO
d goto VERDADERO

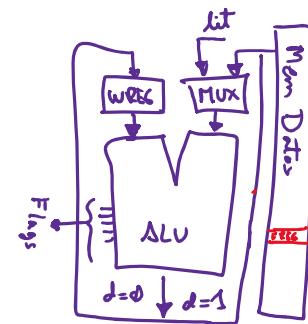
Opción 4:

inicio: movf T-reg, W
sublw .0
b3 VERDADERO
FALSO: =
VERDADERO: =

Aplicación de sublw para tomas de decisión:



morf F-reg, 0
 sublw 0011
~~btfss STATUS, 2~~
 bit 2 (Z)
 F-reg
~~-F-reg~~
~~0-~~
~~-6~~
 ¿Se levantará algún flag?
~~Z=0 N=1~~



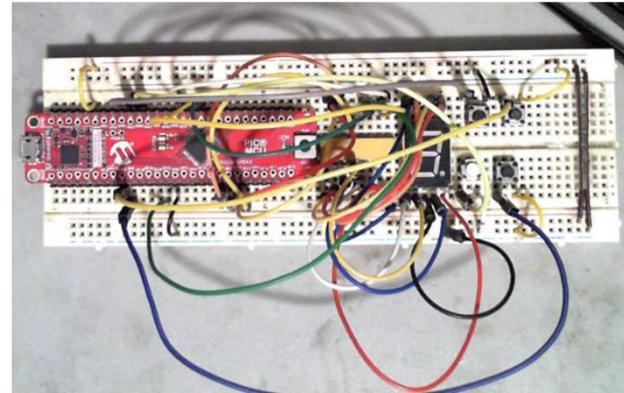
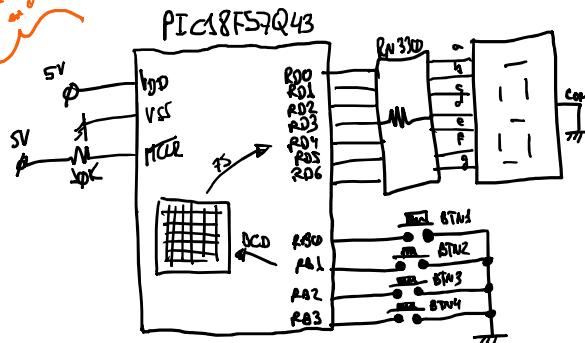
Los saltos BRANCH

- Son saltos cortos condicionales en base a los flags del registro STATUS

BC	n	Branch if Carry
BN	n	Branch if Negative
BNC	n	Branch if Not Carry
BNN	n	Branch if Not Negative
BNOV	n	Branch if Not Overflow
BNZ	n	Branch if Not Zero
BOV	n	Branch if Overflow
BRA	n	Branch Unconditionally
BZ	n	Branch if Zero

Ejemplo 2024-1 Sem4

Nota: Yo dejo implementado el circuito para el desarrollo

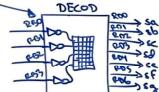


Ejemplo 2024-1 Sem4

Ejemplo de aplicación: Decodificador BCD → SegCC

Observaciones del hardware:

- 1: Entradas del decodificador: RB0 ~ RB3
Entradas digitales con pullup activo
- 2: Salidas del decodificador: RD0 ~ RD6
Salidas digitales



⇒ Tenemos que hacer dos operaciones antes de hacer el proceso de la tabla de decodificación

- complemento
- enmascaramiento

Procediendo "LOOK UP TABLES" o tablas de verdad.

$\text{uint}_8 \text{ tabla}[] = \{0x3F, 0x06, 0x5B, 0x4F, \dots\}$

$\text{uint}_8 \text{ salida};$

$= \text{salida} = \text{tabla}[(\text{RA0}) \& (\text{RA1})];$

vector de RST → 0H

Zona de programación de 0x0FF

Memoria de datos

TBLPTR (Table pointer)

00 00 00 00

clrf TBLPTRU
movlw 00H
movwf TBLPTRL

clrf TBLPTRL
movlw 00H
movwf TBLPLH

tblrdx *
tblrdx

Ejemplo 2024-1 Sem4

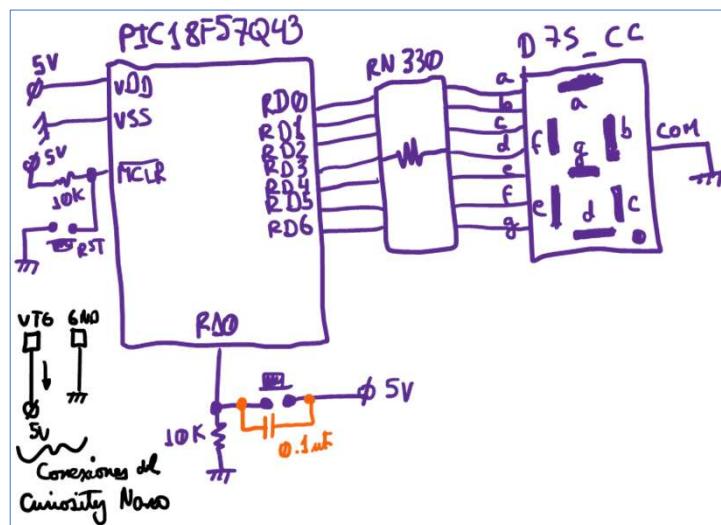
```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6      ORG 000000H
7      bra configuro
8
9      ORG 001000H
10     tabla_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12     ORG 000100H
13 configuro:
14     movlb 0H
15     movlw 60H
16     movwf OSCCON1, 1
17     movlw 03H
18     movwf OSCFRQ, 1
19     movlw 40H
20     movwf OSCEN, 1
21     movlb 4H
22     movlw 0FFH
23     movwf TRISB, 1
24     movlw 0FOH
25     movwf ANSELB, 1
26     movlw 0FH
27     movwf WPUB, 1
28     movlw 80H
29         movwf TRISD, 1
30         movwf ANSELD, 1
31         movlw 00H
32         movwf TBLPTRU, 1
33         movlw 10H
34         movwf TBLPTRH, 1
35         movlw 00H
36         movwf TBLPTRL, 1
37
38 inicio:
39         comf PORTB, 0, 1
40         andlw 0FH
41         movwf TBLPTRL, 1
42         TBLRD*
43         movff TABLAT, LATD
44         bra inicio
45
46 end upcino

```

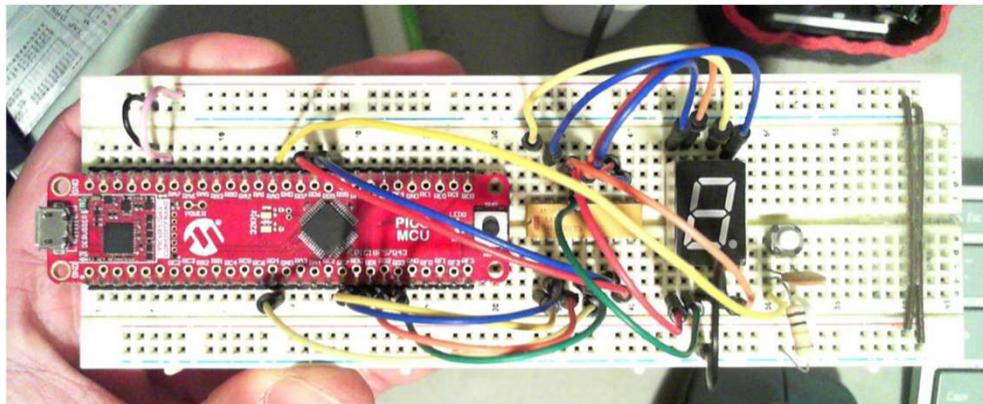
Ejemplo 2024-2

- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.



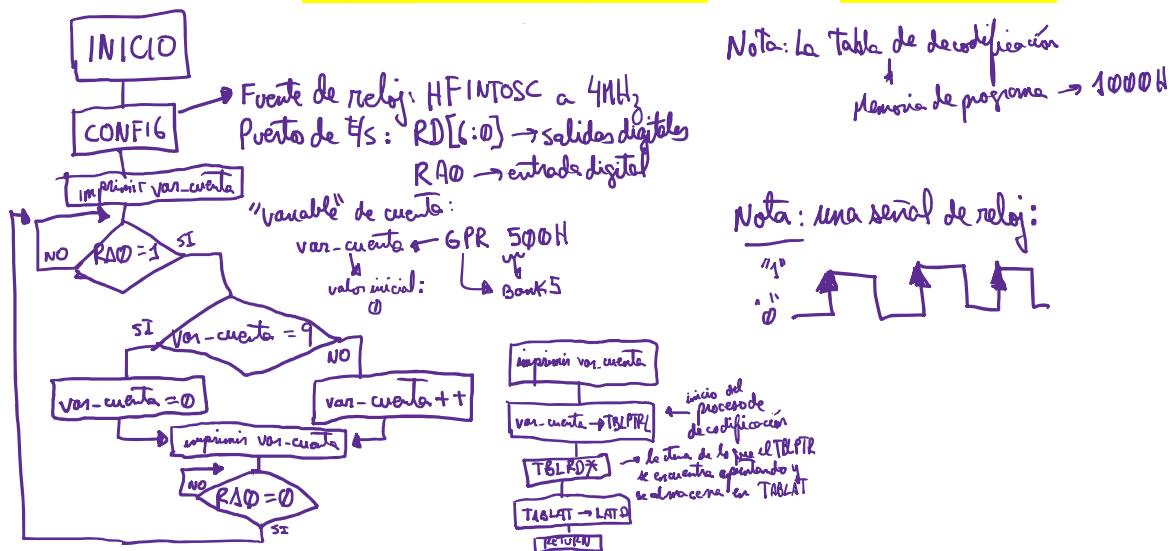
Ejemplo 2024-2

- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.



Ejemplo 2024-2

- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.



Ejemplo 2024-2

- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  var_cuenta EQU 500H ;un label al GPR 500H
5
6  PSECT upcino, class=CODE, reloc=2, abs
7  upcino:
8      ORG 000000H
9      bra configuro
10
11     ORG 000800H
12     decod_bcd: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
13     ;   0   1   2   3   4   5   6   7   8   9
14
15     ORG 000100H
16 configuro:
17     ;configuracion de la fuente de reloj
18     movlb 0H
19     movlw 60H
20     movwf OSCCON1, 1
21     movlw 02H
22     movwf OSCFRC, 1
23     movlw 40H
24     movwf OSCEN, 1
25     ;configuracion de las E/S
26     movlb 4H
27     movlw 80H
28     movwf TRISD, 1 ;RD[6:0] como salidas
29     movwf ANSELDA, 1 ;RD[6:0] como digital
30     bsf TRISA, 0, 1 ;RA0 como entrada
31     bcf ANSELA, 0, 1 ;Ra0 como digital
32
33     ;configuraciones adicionales
34     clrf TBLPTRU, 1
35     movlw 08H
36     movwf TBLPTRH, 1
37     clrf TBLPTRL, 1 ;TBLPTR direccionado a 0800H en la mem de p
38     movlb 5H
39     clrf var_cuenta, 1 ;var_cuenta con valor inicial cero
40     movlw 4H
41     call imprimir_var_cuenta
42
43 inicio_contador:
44     btfs PORTA, 0, 1 ;pregunto si presione boton
45     bbra inicio_contador ;falso, no presione boton
46     movlb 5H
47     movlw 9
48     cpfseq var_cuenta, 1 ;pregunto si var_cuenta=9
49     bra no_es_nueva ;falso
50     bra si_es_nueva ;verdad
51     no_es_nueva:
52     incf var_cuenta, 1, 1 ;incremento var_cuenta
53     bra siguiente
54     si_es_nueva:
55     clrf var_cuenta, 1 ;mando a cero var_cuenta
56     siguiente:
57     call imprimir_var_cuenta
58     aun_no:
59     btfs PORTA, 0, 1 ;pregunto si dejaste de presionar boton
60     bra aun_no
61     bbra inicio_contador
62
63 imprimir_var_cuenta:
64     movlb 5H
65     movwf var_cuenta, 0, 1 ;muevo var_cuenta a Wreg
66     movlb 4H
67     movwf TBLPTRL, 1 ;direccion del TBLPTR
68     TBLRD*
69     movff TABLAT, LATD ;accion de lectura
70
71     return
72
73 end upcino

```

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6      ORG 000000H
7      bra configuro
8
9      ORG 001000H
10     deco7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11     ;   0   1   2   3   4   5   6   7   8   9
12
13     ORG 000100H
14 configuro:
15     ;conf fuente de reloj
16     movlb 0H ;voy al Bank0
17     movlw 60H
18     movwf OSCCON1, 1
19     movlw 02H
20     movwf OSCFRC, 1
21     movlw 40H
22     movwf OSCEN, 1
23     ;conf puertos E/S
24     movlb 4H ;voy al Bank4
25     setf TRISB, 1 ;RB[3:0] como entradas
26     movlw 0FOH
27     movwf ANSELB, 1 ;RB[3:0] como digitales
28     bsf TRISE, 0, 1 ;RE0 como entrada
29     bcf ANSELE, 0, 1 ;RE0 como digital
30     movlw 80H
31     movwf TRISD, 1 ;RD[6:0] como salida
32     movwf ANSELD, 1 ;RD[6:0] como digital
33
34     ;conf adicional
35     clrf TBLPTRU, 1
36     movlw 10H
37     movwf TBLPTRH, 1 ; U H L
38     clrf TBLPTRL, 1 ;TBLPTR apuntando a 001000H
39
40 inicio_deco:
41     movwf PORTB, 0, 1 ;Leo el puerto RB[3:0] y lo almaceno en Wreg
42     andlw 0FH ;Enmascaramiento de los cuatro LSB de WReg
43     movwf TBLPTRL, 1 ;asignamos el valor de Wreg hacia TBLPTRL
44     TBLRD*
45     movff TABLAT, LATD ;mueve contenido leido y alojado en TABLAT hacia RD
46     bra inicio_deco
47
48 end upcino

```

Códigos de ejemplos 2024-2

- Decodificador BCD-7Segments

```

1      PROCESSOR 18F57Q43
2      #include "cabecera.inc"
3
4      PSECT upcino, class=CODE, reloc=2, abs
5      upcino:
6          ORG 000000H
7          bra configuro
8
9          ORG 001000H
10         decod: DB 3FH, 06H, 5BH, 4FH, 66h, 6DH, 7DH, 07H, 7FH, 67H
11         ;      0   1   2   3   4   5   6   7   8   9
12
13         ORG 000100H
14         configuro:
15             ;configuracion de la fuente de reloj
16             movlb OH
17             movlw 60H
18             movwf OSCCON1, 1
19             movlw 02H
20             movwf OSCFRQ, 1
21             movlw 40H
22             movwf OSCEN, 1
23             ;configuracion de las E/S
24             movlb 4H
25             movlw 80H
26             movwf TRISB, 1      ;RB[6:0] como salidas
27             movwf ANSELB, 1     ;RB[6:0] como digitales
28             setf TRISD, 1      ;RD como entradas
29             movlw OFOH
30             movwf ANSELD, 1     ;RD[3:0] como digitales
31
32             ;configuraciones adicionales
33             clrf TBLPTRU, 1
34             movlw 10H
35             movwf TBLPTRH, 1
36             clrf TBLPTRL, 1      ;direccion asignada a TBLPTR=001000H
37             inicio_decod:
38                 ;tengo que leer el puerto de entrada RD[3:0]
39                 movf PORTD, 0, 1    ;leo RD y lo mando a Wreg
40                 andlw OFH           ;enmascaramiento a Wreg con el valor OFH
41                 ;tengo que decodificar usando el TBLPTR
42                 movwf TBLPTRL, 1      ;muevo contenido de Wreg a TBLPTRL
43                 TBLRD*               ;lectura de lo que apunta TBLPTR y se graba en TABLAT
44                 ;tengo que escribir TABLAT a RB[6:0]
45                 movff TABLAT, LATB     ;muevo contenido de TABLAT a RB
46                 bra inicio_decod
47
48             end upcino|

```

Dígitos de ejemplos 2024-2

ecodificador BCD-7Segmentos

```

1      PROCESSOR 18F57Q43
2      #include "cabecera.inc"
3
4      PSECT upcino, class=CODE, reloc=2, abs
5      upcino:
6          ORG 000000H
7          bra configuro
8
9          ORG 001000H      ;sector de mem de prog de datos del dec
10         dec7s: DB 3FH, 06H, 5BH, 4FH, 66H
11         ;      0   1   2   3   4
12         ;      DB 6DH, 7DH, 07H, 7FH, 67H
13         ;      5   6   7   8   9
14
15         ORG 000100H
16         configuro:
17             ;conf fuente de reloj
18             movlb OH
19             movlw 60H
20             movwf OSCCON1, 1
21             movlw 02H
22             movwf OSCFRQ, 1
23             movlw 40H
24             movwf OSCEN, 1
25             ;conf de E/S
26             movlb 4H
27             movlw 80H
28             movwf TRISD, 1      ;RD[6:0] como salidas
29             movwf ANSELD, 1     ;RD[6:0] como digitales
30             setf TRISE, 1      ;RB[3:0] como entradas
31
32             movlw OFOH
33             movwf ANSELB, 1     ;RB[3:0] como digitales
34             bsf TRISE, 0, 1     ;RE0 como entrada
35             bcf ANSELE, 0, 1     ;RE0 como digital
36             ;conf adicionales
37             ;asignacion de direccion a apuntar por el TBLPTR
38             clrf TBLPTRU, 1
39             movlw 10H
40             movwf TBLPTRH, 1
41             clrf TBLPTRL      ;TBLPTR apunta a 001000H
42             inicio_decod:
43                 movf PORTB, 0, 1    ;lectura de RB
44                 andlw OFH           ;enmascaramiento para pasar RB[3:0]
45                 movwf TBLPTRL, 1      ;muevo Wreg a TBLPTRL para accion de decodificacion
46                 TBLRD*               ;accion de lectura
47                 movff TABLAT, LATD     ;muevo TABLAT a LATD
48                 bra inicio_decod
49
50             end upcino|

```

Códigos de ejemplos 2024-2

• Decodificador BCD-7Segmentos

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6  ORG 000000H
7  bra configuro
8
9  ORG 001000H
10 deco_7s:   DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12 ORG 000100H
13 configuro:
14     ;conf fuente de reloj:
15     movlb 0H
16     movlw 60H
17     movwf OSCCON1, 1
18     movlw 02H
19     movwf OSCFRQ, 1
20     movlw 40H
21     movwf OSCEN, 1
22     ;conf puertos E/S:
23     movlb 4H
24     movlw 80H
25     movwf TRISD, 1      ;RD[6:0] como salidas
26     movwf ANSEL0, 1      ;RD[6:0] como digitales
27     setf TRISB, 1        ;RB[3:0] como entradas
28     movlw 0FOH
29     movwf ANSELB, 1      ;RB[3:0] como digitales
30     bsf TRISE, 0, 1      ;RE0 como entrada
31     bcf ANSELE, 0, 1      ;RE0 como digital
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

Códigos de ejemplos 2024-2

- Decodificador BCD-7Segmentos

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  cuenta EQU 500H      ;asignacion de label a GPR 500H
5
6  PSECT upcino, class=CODE, reloc=2, abs
7  upcino:
8  ORG 000000H
9  bra configuro
10
11 ORG 001000H
12 decod: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
13 ;      0 1 2 3 4 5 6 7 8 9
14
15 ORG 000100H
16 configuro:
17     ;configuracion de la fuente de reloj
18     movlb 0H
19     movlw 60H
20     movwf OSCCON1, 1
21     movlw 02H
22     movwf OSCFRQ, 1
23     movlw 40H
24     movwf OSCEN, 1
25     ;configuracon de las E/S
26     movlb 4H
27     movlw 80H
28     movwf TRISB, 1      ;RB[6:0] como salidas
29     movwf ANSELB, 1      ;RB[6:0] como digitales
30     setf TRISD, 1        ;RD como entradas
31     movlw 0FOH
32     movwf ANSEL0, 1      ;RD[3:0]] como digitales
33     bsf TRISE, 0, 1      ;RE0 como entrada
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

Códigos de ejemplos 2024-2

- Contador BCD-7Segmentos

```

34     bcf ANSELE, 0, 1      ;RE0 como digital
35     ;configuraciones adicionales
36     clrf TBLPTRU, 1
37     movlw 10H
38     movwf TBLPTRH, 1
39     clrf TBLPTRL, 1      ;TBLPTR esta apuntando a 001000H
40     inicio_deco:
41         movf PORTB, 0, 1      ;leo RB y lo mando a Wreg
42         andlw 0FH
43         movwf TBLPTRL, 1      ;enmascaramiento para que pase solo RB[3:0]
44         TBLRD*
45         movff TABLAT, LATD
46         bra inicio_deco
47         end upcino
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65     imp_7s:
66         movlb 5H
67         movf cuenta, 0, 1      ;mando cuenta a Wreg
68         movlb 4H
69         movwf TBLPTRL, 1      ;muevo Wreg a TBLPTRL
70         TBLRD*
71         movff TABLAT, LATB
72         return
73
74     end upcino

```

```

1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  var_cuenta EQU 500H ;asignando etiqueta a GPR 500H
5
6  PSECT upcino, class=CODE, reloc=2, abs
7  upcino:
8    ORG 000000H
9    bra configuro
10
11   ORG 001000H
12  deco_7s:   DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
13
14   ORG 000100H
15  configuro:
16    ;conf fuente de reloj:
17    movlw 0H
18    movlw 60H
19    movwf OSCCON1, 1
20    movlw 02H
21    movwf OSCFRQ, 1
22    movlw 40H
23    movwf OSCEN, 1
24    ;conf puertos E/S:
25    movlb 4H
26    movlw 80H
27    movwf TRISB, 1
28    movwf ANSELB, 1
29    setf TRISB, 1
30    movlw 0FOH
31    movwf ANSELB, 1
32    bcf TRISE, 0, 1
33    bcf ANSELE, 0, 1
34
35    ;RD[6:0] como salidas
36    ;RD[6:0] como digitales
37    ;RB[3:0] como entradas
38
39    ;conf fuente de reloj:
40    movlw 0H
41    movwf OSCCON1, 1
42    call imp_7s
43
44    ;conf adicional
45    clrf TBLPTRD, 1
46    movlw 10H
47    movwf TBLPTRH, 1
48    clrf TBLPTRL, 1
49    movlb 5H
50    clrf var_cuenta, 1
51    movlb 4H
52    call inicio_conta
53
54    ;pregunto si presione boton
55    btfss PORTE, 0, 1
56    bra $-2
57    movlb 5H
58    movlw 9
59    cpfseq var_cuenta, 1
60    bra no_es_nueve
61    clrf var_cuenta, 1
62    bra siguiente
63    no_es_nueve:
64    incf var_cuenta, 1, 1
65    siguiente:
66    call imp_7s
67    btfsc PORTE, 0, 1
68    bra $-2
69    bra inicio_conta
70
71    imp_7s:
72    movlb 5H
73    movf var_cuenta, 0, 1
74    movlb 4H
75    movwf TBLPTRL, 1
76    TBLRD*
77    movff TABLAT, LATD
78    return
79
80  end upcino

```

Códigos de ejemplos 2024-2

- Contador BCD-7Segmentos

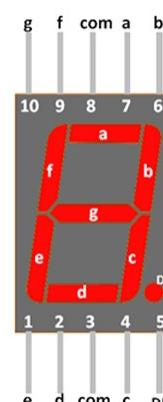
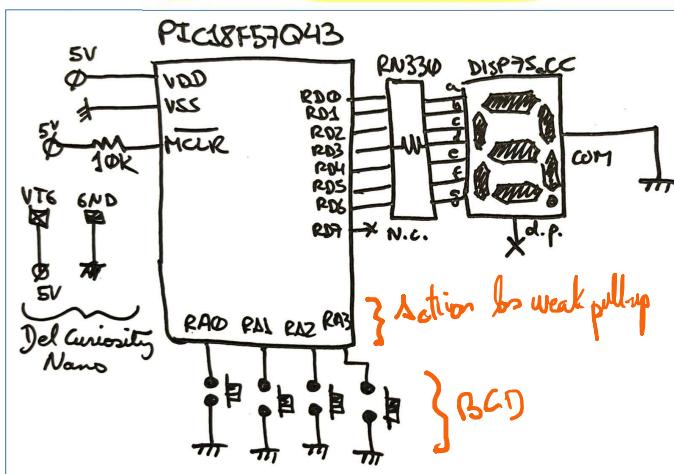
```

34    ;conf adicional
35    clrf TBLPTRD, 1
36    movlw 10H
37    movwf TBLPTRH, 1
38    clrf TBLPTRL, 1
39    movlb 5H
40    clrf var_cuenta, 1
41    movlb 4H
42    call imp_7s
43
44    ;conf adicional
45    clrf TBLPTRD, 1
46    movlw 10H
47    movwf TBLPTRH, 1
48    clrf TBLPTRL, 1
49    movlb 5H
50    clrf var_cuenta, 1
51    movlb 4H
52    call inicio_conta
53
54    ;pregunto si presione boton
55    btfss PORTE, 0, 1
56    bra $-2
57    movlb 5H
58    movlw 9
59    cpfseq var_cuenta, 1
60    bra no_es_nueve
61    clrf var_cuenta, 1
62    bra siguiente
63    no_es_nueve:
64    incf var_cuenta, 1, 1
65    siguiente:
66    call imp_7s
67    btfsc PORTE, 0, 1
68    bra $-2
69    bra inicio_conta
70
71    imp_7s:
72    movlb 5H
73    movf var_cuenta, 0, 1
74    movlb 4H
75    movwf TBLPTRL, 1
76    TBLRD*
77    movff TABLAT, LATD
78    return
79
80  end upcino

```

Ejercicio 2025-1

- Desarrollar un decodificador de BCP a siete segmentos para display de siete segmentos de cátodo común donde las entradas son activos en bajo (deberán de activar las weak pullup de los pines)

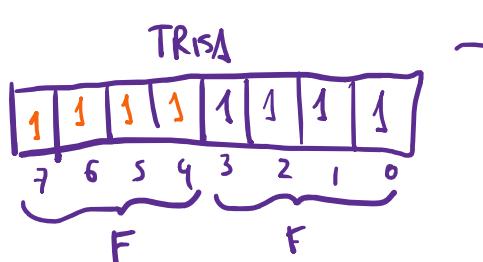


Configuración de los pines RA

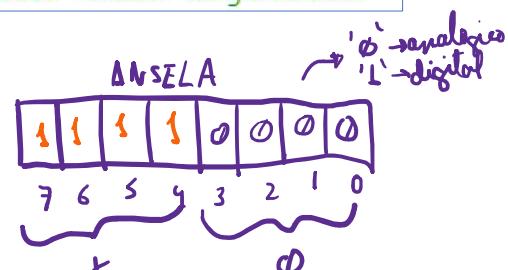
```

movlw 0FFH
movwf TRISA, 1 ;RA3 al RA0 como entradas
movlw 0FOH
movwf ANSELA, 1 ;RA3 al RA0 como digitales

```



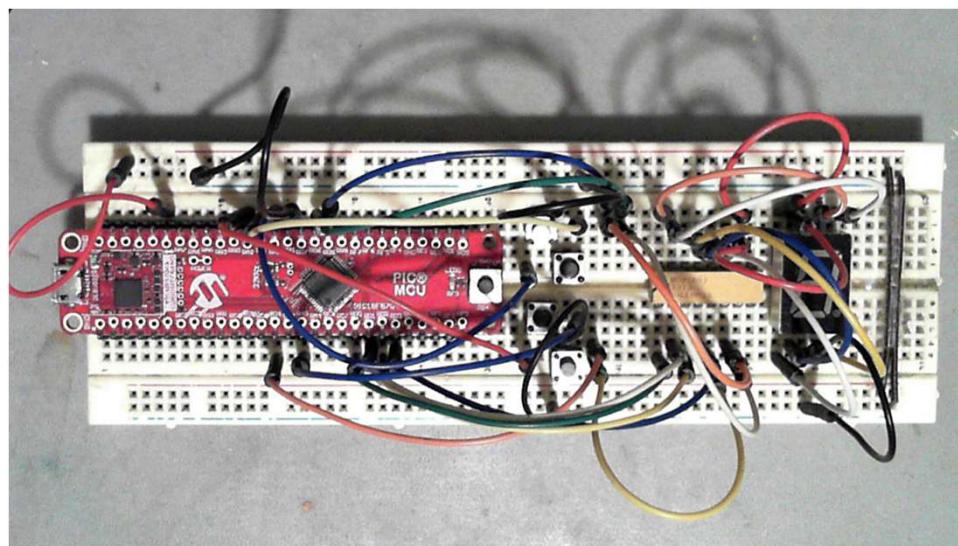
→ '0' → salida
'1' → entrada



→ '0' → analógico
'1' → digital

Ejercicio 2025-1

- Implementar el siguiente circuito



Ejercicio 2025-1

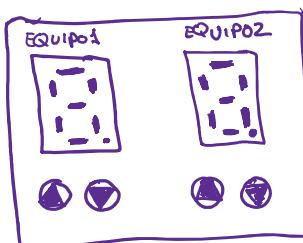
```

1 ;Programa de decodificador BCD a display de siete segmentos cátodo común
2 PROCESSOR 18F57Q43
3 #include "cabecera.inc"
4
5 PSECT upcino, class=CODE, reloc=2, abs
6 upcino:
7
8     ORG 000000H
9     bra configuro
10
11    ORG 001000H
12 ;Aquí estarán los datos de la tabla de decodificación
13 datos_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
14
15    ORG 000100H
16 configuro:
17 ;configuración de la fuente de reloj
18     movlb 0H          ;al Bank0
19     movlw 60H
20     movwf OSCCON1, 1  ;NOSC=HFINTOSC, NDIV=1:1
21     movlw 02H
22     movwf OSCFRQ, 1   ;HFINTOSC a 4MHz
23     movlw 40H
24     movwf OSCEN, 1    ;HFINTOSC enabled
25 ;configuración de los puertos
26     movlb 4H
27     movlw 0FFH
28     movwf TRISA, 1    ;RA3 al RA0 como entradas
29     movlw 0FOH
30     movwf ANSELA, 1   ;RA3 al RA0 como digitales
31     movlw 0FH
32     movwf WPUA, 1      ;RA3 al RA0 con pullups
33     movlw 80H
34     movwf TRISD, 1      ;RD6 al RD0 como salidas
35     movwf ANSEL0, 1      ;RD6 al RD0 como digitales
36 ;configuración del puntero de tabla ó TBLPTR
37     clrf TBLPTRU, 1      ;TBLPTRU = 00H
38     movlw 10H
39     movwf TBLPTRH, 1      ;TBLPTRH = 10H
40     clrf TBLPTRL, 1      ;TBLPTRL = 00H ---> dirección completa 001000H
41
42 inicio:
43 ;Zona de la aplicación
44     comf PORTA, 0, 1    ;Leo el puerto A y le aplico complemento y el resultado va a wreg
45     andlw 0FH            ;operación de enmascaramiento para que solo estén RA3 al RA0
46     movwf TBLPTRL, 1      ;movemos el contenido de Wreg hacia el puntero de tabla
47     TBLRD*                ;acción de lectura del puntero de tabla y lo aloja en TABLAT
48     movff TABLAT, LATD    ;movemos contenido de TABLAT hacia LATD (donde está el display)
49     bra inicio
50
51 end upcino

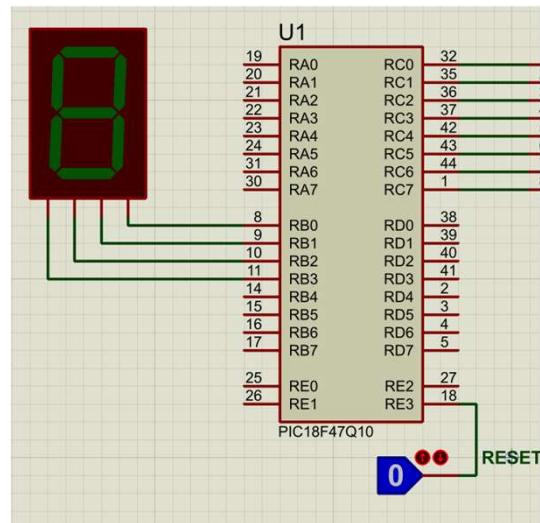
```

Ejercicios adicionales:

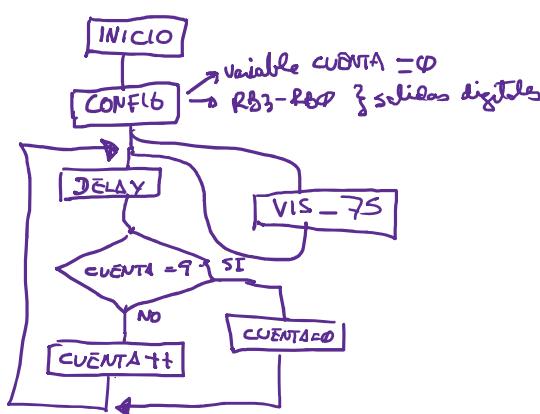
- Desarrollar un visualizador de mensaje “SOY FELIZ” a través de un display de siete segmentos del tipo cátodo común a razón de una letra a la vez y con periodo de cambio de 500ms.
- Desarrollar un tablero de score para una cancha deportiva, el cual se tenga dos displays y cuatro pulsadores clasificados en un display y dos pulsadores para cada equipo, un pulsador será para incrementar la cuenta y el otro para decrementar la cuenta.



Ejercicio 2026-0 Contador 0-9 con salida BCD autoincremental con periodo de tiempo 200ms entre incrementos



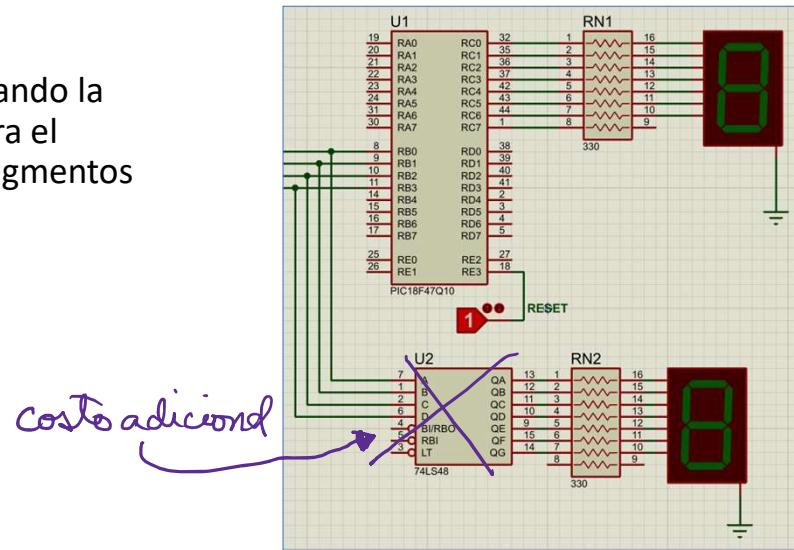
Ejercicio 2026-0 Contador 0-9 con salida BCD autoincremental con periodo de tiempo 200ms entre incrementos



- No será necesario implementar en esta primera etapa el decodificador ya que el display empleado en la simulación en Proteus ya tiene el decodificador implementado en el mismo display.

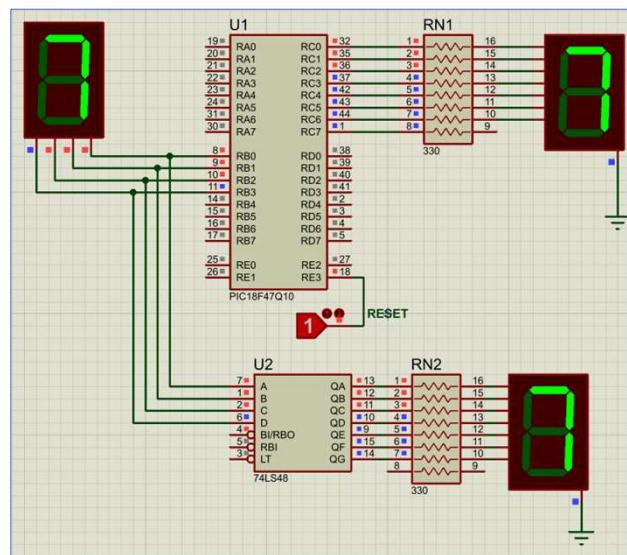
Ejercicio 2026-0 Contador 0-9 con salida BCD autoincremental con periodo de tiempo 200ms entre incrementos

- Ahora implementando la decodificación para el display de siete segmentos regular



Ejercicio 2026-0 Contador 0-9 con salida BCD autoincremental con periodo de tiempo 200ms entre incrementos

- Implementación final con los tres casos

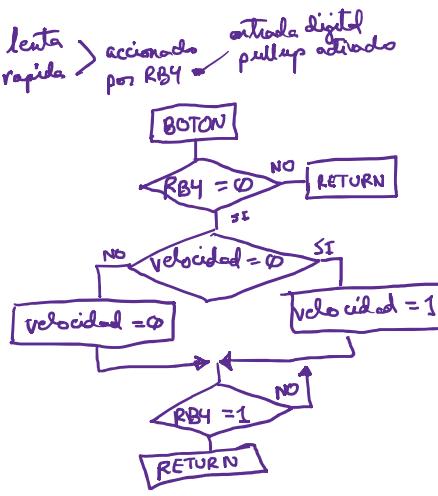
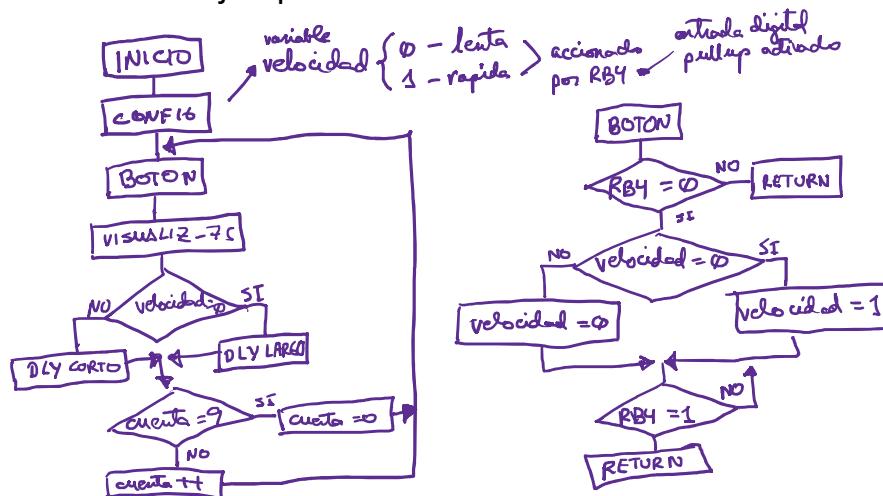


Ejercicios adicionales:

- Agregar un pulsador en RB4 cambiar la velocidad entre lento y rápido en la cuenta del ejemplo anterior
- Agregar un pulsador en RB5 para que al pulsarlo, la cuenta inicie en 4
- Agregar un segundo display que que el rango de cuenta sea de 00-99 en decimal preservando las opciones de botones anteriores.

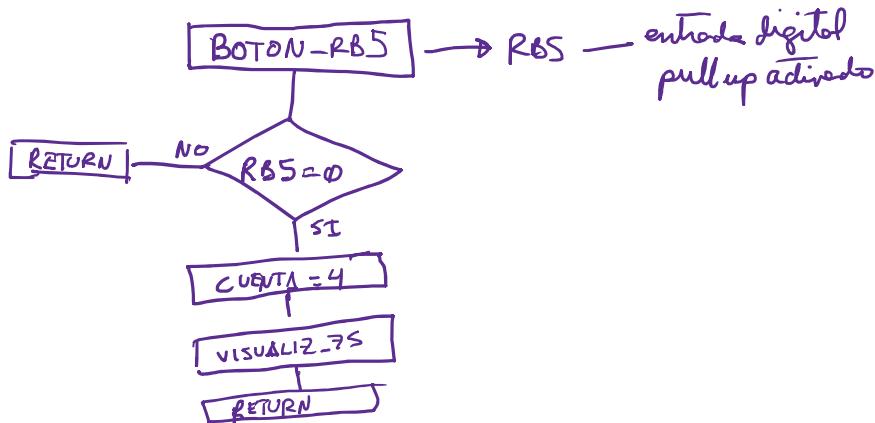
Ejercicios adicionales:

- Agregar un pulsador en RB4 cambiar la velocidad entre lento y rápido en la cuenta del ejemplo anterior



Ejercicios adicionales:

- Agregar un pulsador en RB5 para que al pulsarlo, la cuenta inicie en 4



Fin de la sesión