

Microcontroladores

Semana 3

Semestre 2025-1

Profesor: Kalun José Lau Gan

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:

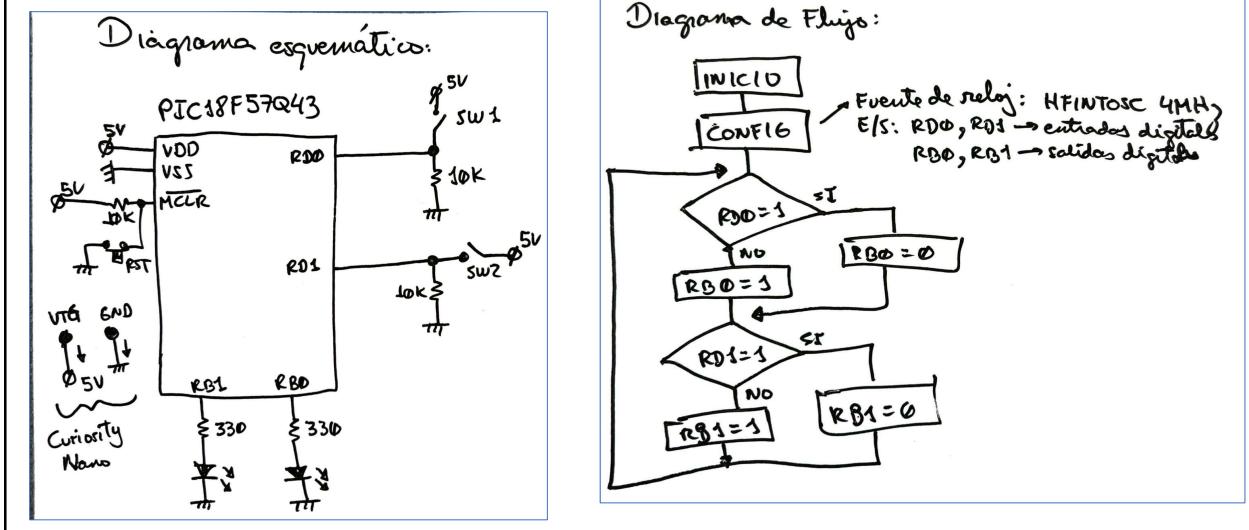
- He visto videos de youtube y algunos libros (el del pic16f84 supongo de José Angulo Usategui) donde hay algunas discrepancias en la representación de números en el Assembler
 - No hay discrepancias, un número en cualquiera de sus representaciones numéricas (hex, bin o dec) siempre es el mismo valor, 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, promediacióñ, 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 y/o en la ruta del proyecto.
 - Ej. Semana_2⁴ <- No debe de colocarse el punto en el nombre
 - Ej. Semana2_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?



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

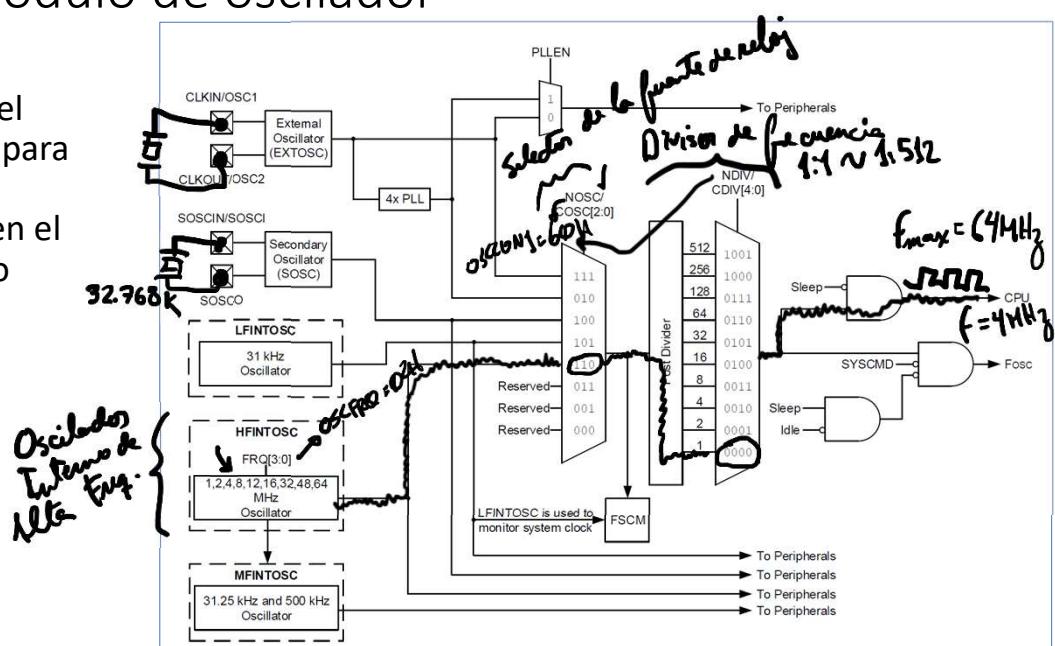
[Close](#)

Agenda:

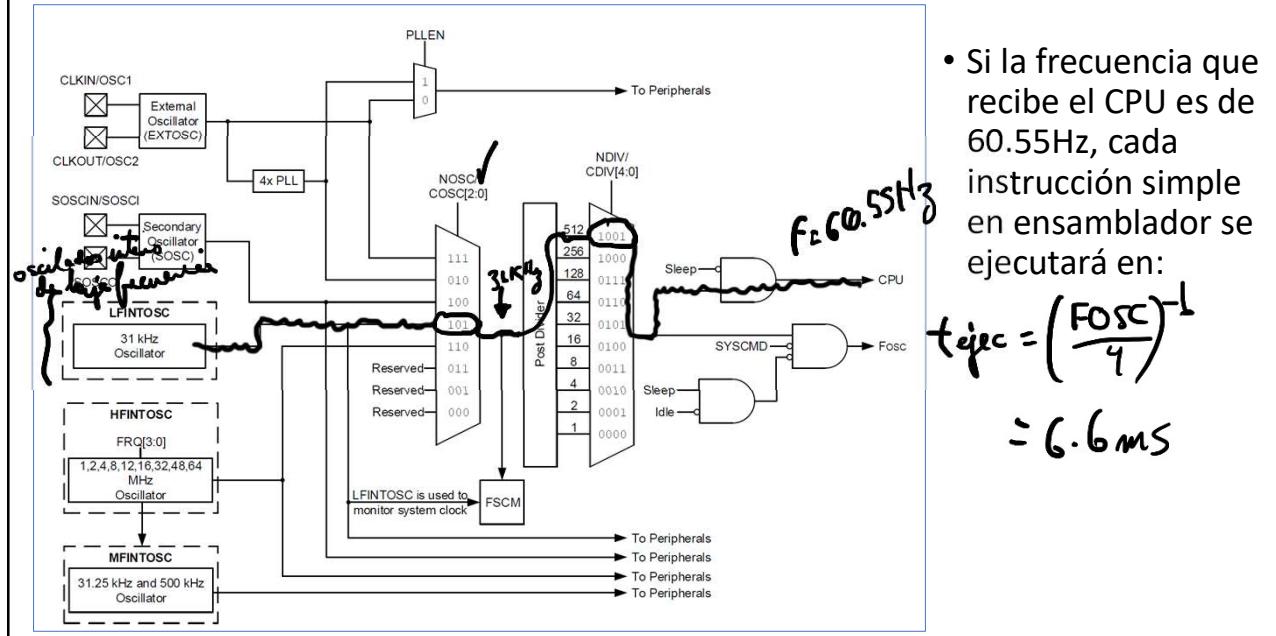
- Estructura interna de un microcontrolador
- Memoria de programa del microcontrolador PIC18F57Q43
- Instrucciones básicas en XC8 PIC Assembler para el PIC18F57Q43
- El contador de programa (PC) del CPU del microcontrolador PIC18F57Q43
- Acceso a datos almacenados en la memoria de programa empleando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

El módulo de oscilador

- Nosotros usaremos el HFINTOSC para nuestros ejemplos en el laboratorio

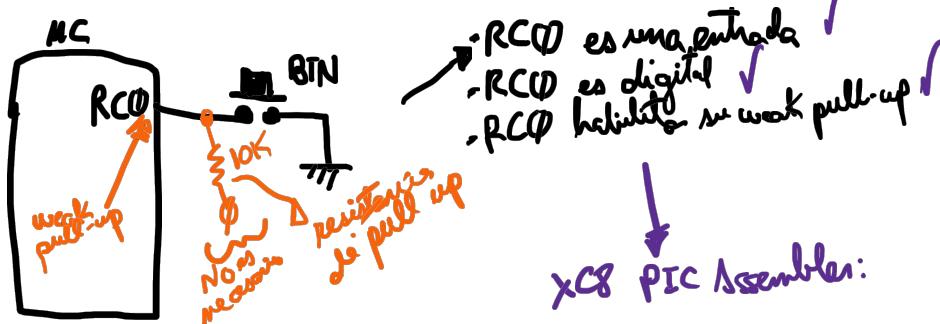


Camilo: ¿Cuál es la mínima frecuencia posible?



Recordando la configuración de los puertos de E/S:

- Definir las configuraciones al siguiente caso:

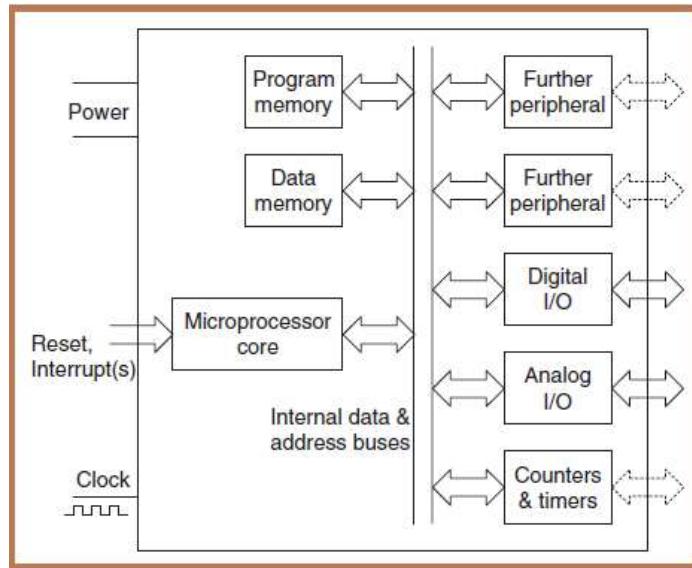


Según CLD: Las entradas no deben de estar al aire!

x8 PIC Assembler:

```
bsf TRISC, 0, 1
bcf ANSEL0, 0, 1
bsf WPUC, 0, 1
```

Estructura interna de un microcontrolador



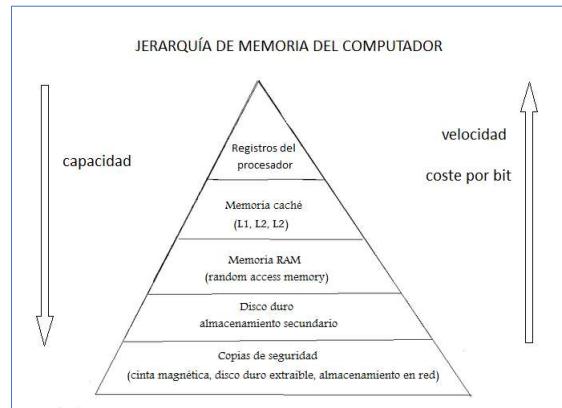
- CPU: Encargado de ejecutar las instrucciones
- Soporte del CPU: módulo de reloj, dispositivos de seguridad, etc.
- **Memoria de programa:** **No volátil**, almacena programa y datos constantes
- **Memoria de datos:** **Volátil**, almacena datos temporales y contiene los registros SFR
- Entradas y salidas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

¿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 PIC18F57Q43 de Microchip

- Estructura de la memoria de programa del PIC18F57Q43:
 - 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-3FFFFFFH (22 bits – 4Mbyte de direcciones)
- Recordar que la dirección 000000H es el vector de RESET

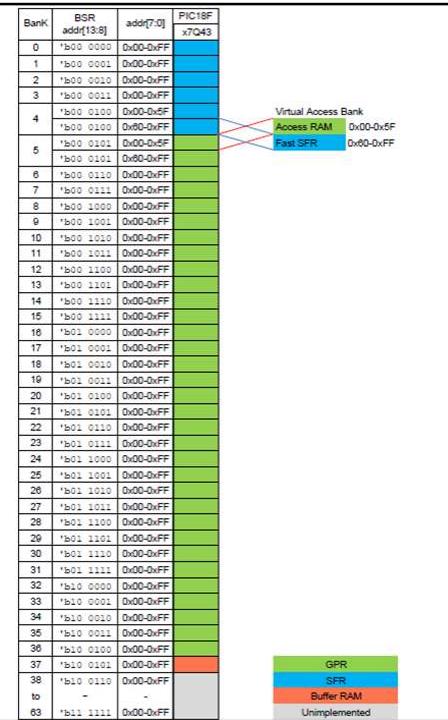
Address	Device
00 0000h to 00 3FFFh	PIC18F57Q43
00 4000h to 00 7FFFh	Program Flash Memory (64 KW) ⁽¹⁾
00 8000h to 00 FFFFh	
01 0000h to 01 FFFFh	
02 0000h to 1F FFFFh	Not Present ⁽²⁾
20 0000h to 20 003Fh	User ID (32 Words) ⁽³⁾
20 0040h to 2B FFFFh	Reserved
2C 0000h to 2C 00FFh	Device Information Area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

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 PIC18F57Q43 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			



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.
 - Por ejemplo: Deseo manipular el puerto RE0 como salida digital

```
ejemplo:  
    movlb 4H      ;me voy al Bank4  
    bcf TRISE, 0, 1 ;REO sea salida  
    bcf ANSEL0, 1, 1 ;REO 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++) {    ↑ límite → 255
    }                                     } se va a repetir 10 veces
}

```

¿Cómo hago para hacer mas 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++) {
        }                                     } ¿Cuántos veces se repite? → 40000
    }
}

```

Recordando saltos a sub-rutinas

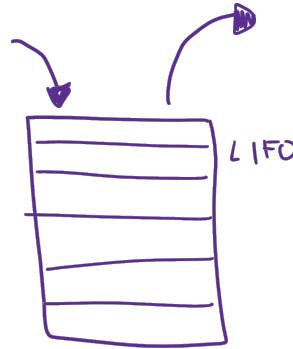
- ¿Cómo sabe el CPU que debe de retornar de donde saltó?
 - Existe una memoria el cual almacena la dirección de retorno y que le va a permitir al CPU regresar una vez atendido la eventualidad.
 - Dicha memoria se le conoce como "stack" o de pila (acción de apilar).

Memorias “Stack” o de pila (apilamiento)

- Estructura FIFO
(first in – first out)



- Estructura LIFO
(last in – first out)



Aspectos relacionados con el MPASM y el 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

<p>MPASM:</p> <pre> list p=18f4550 #include<p18f4550.inc> ; aqui declaramos los bi CONFIG FOSC = XT_XT CONFIG PWRT = ON CONFIG BOR = OFF CONFIG WDT = OFF CONFIG PBADEN = OFF CONFIG LVP = OFF org 0x0000 goto configuru org 0x0020 configru: bsf TRISB, 0 bcf TRISD, 0 principal: btfss PORTB, 0 goto principal btg LATD, 0 otro: btfsc PORTB, 0 goto otro goto principal end </pre>	<p>XC8 PIC ASM:</p> <pre> PROCESSOR 18F4550 #include "cabecera.inc" PSECT rstVect,class=CODE, reloc=2, abs ORG 0000H rstVect: goto configuru ORG 0020H configru: 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 </pre>
---	--

Annotations in red:

- A bracket labeled "Labels" points to the label "configru:" in the MPASM code.
- A bracket labeled "Directivas" points to the "#include" and "CONFIG" lines in the MPASM code.
- A bracket labeled "Configuraciones" points to the "CONFIG" section and the "configru:" label in the MPASM code.
- A bracket labeled "Programa" points to the "principal:" and "otro:" sections in the MPASM code.

Note at the bottom right:

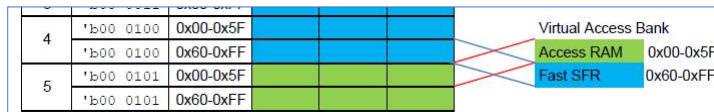
Nota: Los bits de configuración se alojaron en un archivo header llamado "cabecera.inc"

¿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

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
ADDFWF	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	0110	101a	ffff	ffff	Z
COMF	f, d, a	Complement f	1	0000	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	0000	00da	ffff	ffff	Z, N 1
MOVWF	f, d, a	Move f to WREG or f	1	0101	00da	ffff	ffff	Z, N 1
MOVFF	f _s , f _d	Move f _s (16-bit source) to f _d (16-bit destination)	2	1100	$\sum f_s$	$\sum f_d$	$\sum f_d$	None 1, 3, 4
MOVFF	f _s , f _d	Move f _s (16-bit source) to f _d (16-bit destination)	3	1111	$\sum f_s$	$\sum f_d$	$\sum f_d$	None 1, 3
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None 1
NEGf	f, a	Negate f	1	0110	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
RJLNF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N 1
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N 1
RJNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N 1
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None
SUBWF	f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N 1
SUBWF	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
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None 1
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	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:

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

Detalle de una instrucción con opción {a}

- Ejemplo:

Trabajando con Access bank:
movwf TRISB, 0

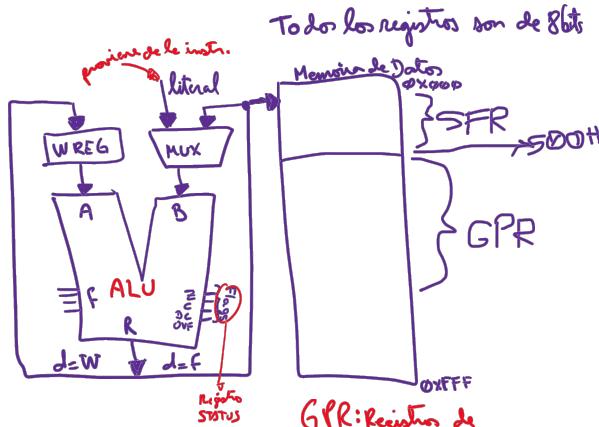
Trabajando con BSR:
(previamente especificar en el BSR cuál es el banco de acceso)
movwf TRISB, 1

movwf TRISB, a *access bank*
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
Decode	Read W
Q3	Process Data
Q4	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



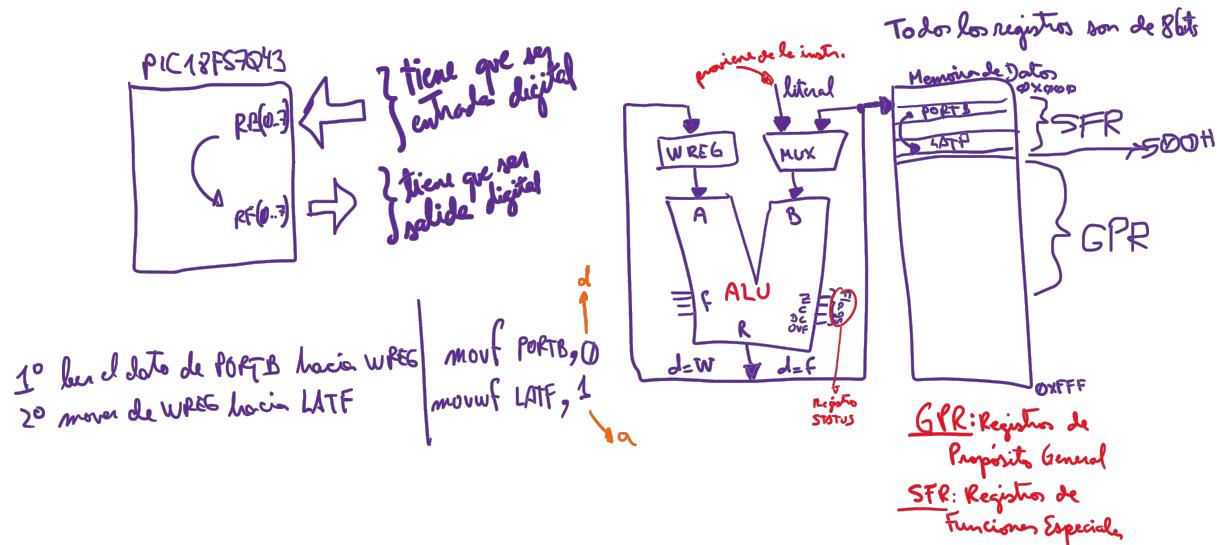
Nota: La configuración y uso de los puertos y periféricos se hace a través de registros SFR.

GPR: Registro de Propósito General
SFR: Registro de Funciones Especiales.

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)

Ejemplo: Pasar un dato del puerto B a 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	C
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/decremento 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)

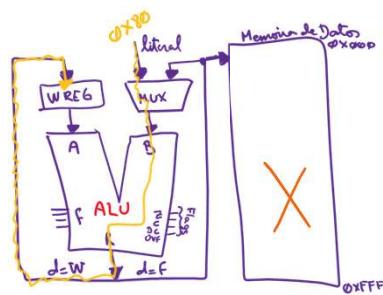
Instrucciones básicas en XC8 PIC Assembler

- Instrucciones de movimiento de datos

movlw [literal]

- mover un literal hacia WREG

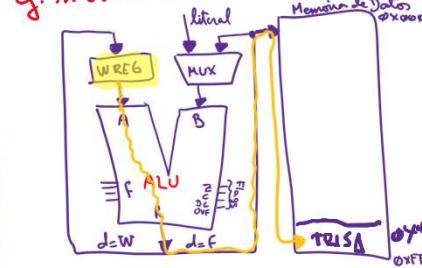
Ej: movlw 0x80



movwf [registro], a

- mover el contenido de WREG hacia [registro]

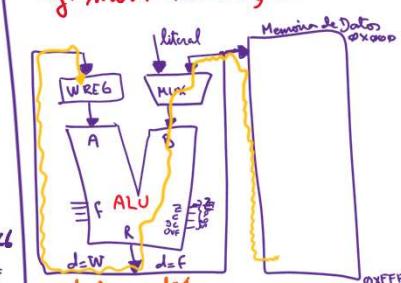
Ej: movwf TRISA



movf [registro], d, a

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

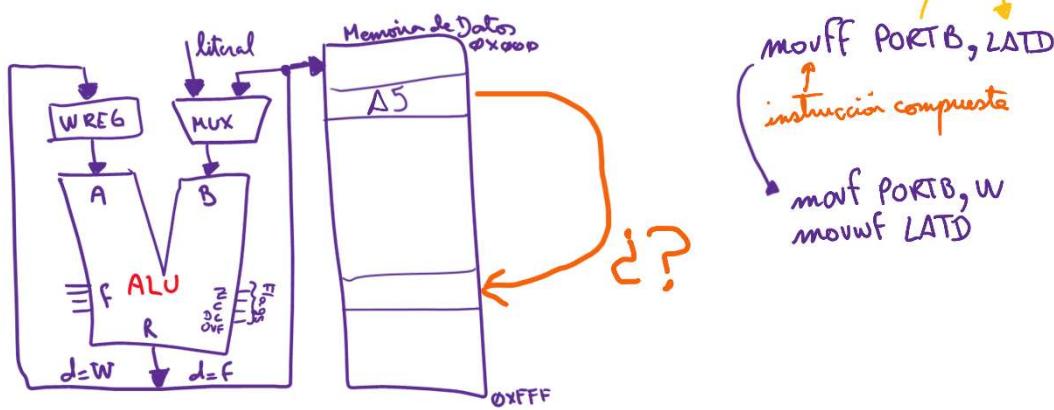
Ej: movf PORTC, W



Nota: movf [registro], f aviso para act. flag

Instrucción movff [registro1], [registro2]

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



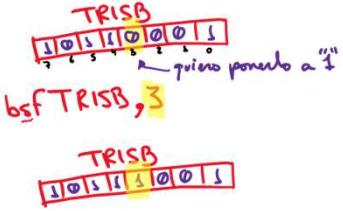
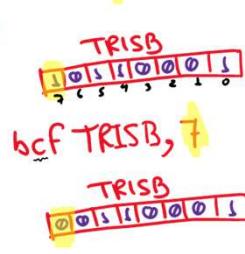
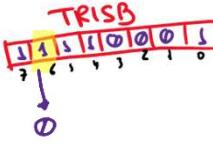
¿Instrucción movfw?

- ¡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.

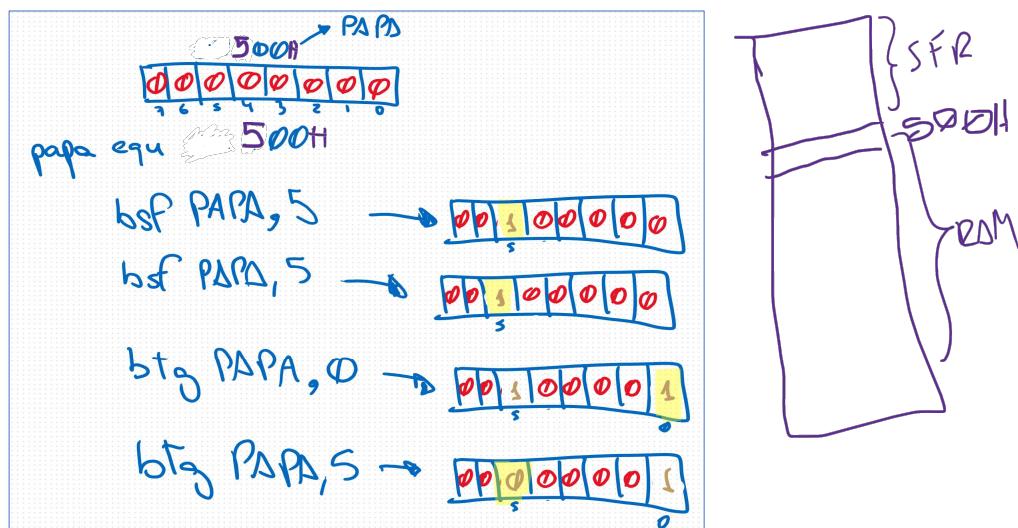
Ej: **movfw TRISA** *simila* **movf TRISA, W**

Instrucciones básicas en XC8 PIC Assembler

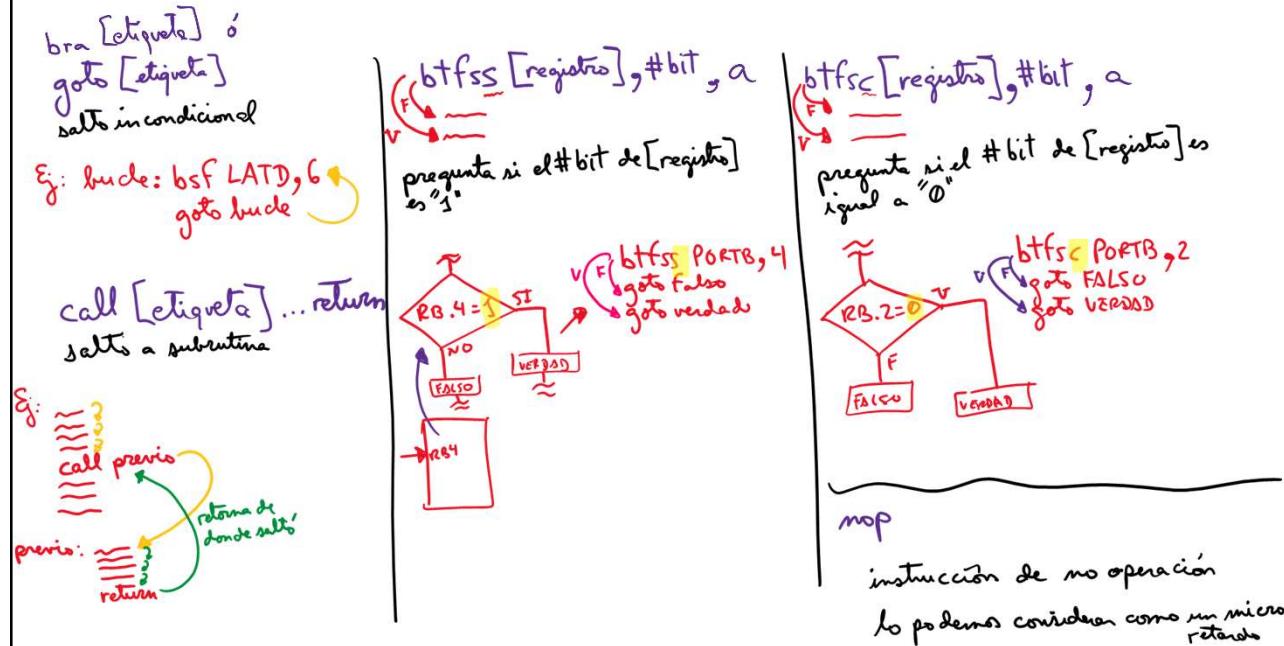
- Instrucciones de manipulación de un bit determinado, en un registro

<p><u>bsf</u> [registro], #bit, ^ posición coloca a "1" el #bit de [registro]</p> <p>Ej:  bsf TRISB, 3</p>	<p><u>bcf</u> [registro], #bit, ^ coloca a "0" el #bit de [registro]</p> <p>Ej: </p>	<p><u>btg</u> [registro], #bit, ^ aplica complemento al #bit de [registro]</p> <p>Ej: btg TRISB, 6 </p>
---	--	---

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



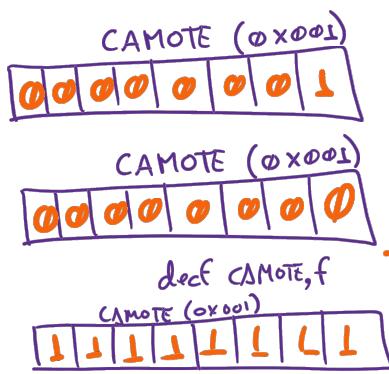
Instrucciones básicas en XC8 PIC Assembler



Instrucciones básicas en XC8 PIC Assembler

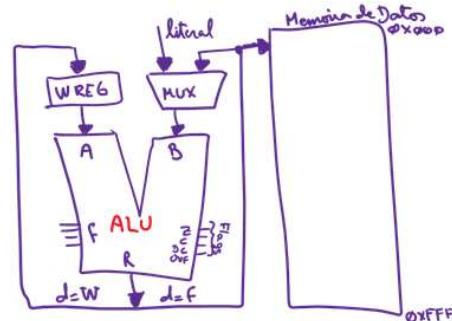
- Instrucción decf / incf
 - Decremento (decf) o incremento (incf) de registro, ambos de **uno en uno**

Ej: decf [registro], d, a incf [registro], d, a
"d" puede ser: f ó w



decf CAMOTE, f

Registro STATUS: Z = 1



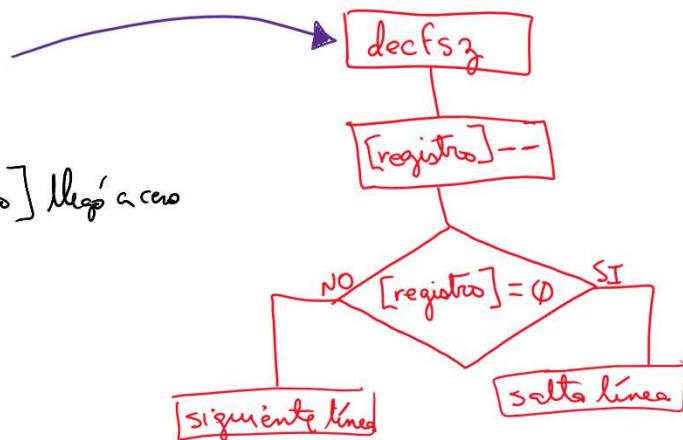
Instrucciones básicas en XC8 PIC Assembler

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:

TRISB	1	0	1	1	0	0	0	1
-------	---	---	---	---	---	---	---	---

`setf TRISB`

TRISB	1	1	1	1	1	1	1	1
-------	---	---	---	---	---	---	---	---

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

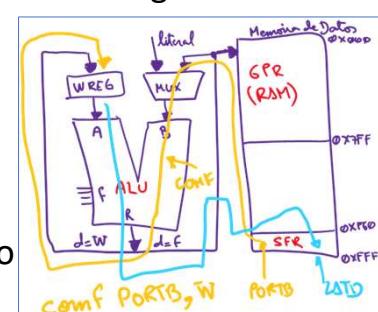
ejemplo:

`clrf TRISB`

TRISB	1	0	1	1	0	0	0	1
-------	---	---	---	---	---	---	---	---

TRISB	0	0	0	0	0	0	0	0
-------	---	---	---	---	---	---	---	---

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



Ejemplo: Puerto D todos sus bits como salida digital:

Opción 1: usando bcf	Opción 2: usando valor numérico	Opción 3: usando clrf
$\text{bcf TRISD}, 0$ $\text{bcf TRISD}, 1$ \vdots $\text{bcf TRISD}, 7$ ocho instrucciones	movlw 00H movwf TRISD, 1 <small>parametro '1'</small> <small>lo trabajan</small>	clrf TRISD, 1 <small>una sola instrucción!</small>

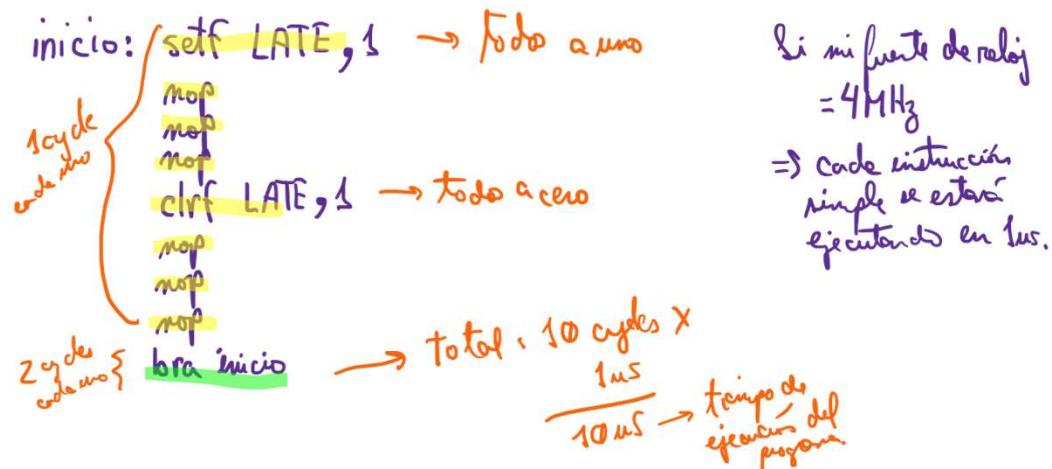
Tiempo de ejecución de las instrucciones en XC8 PIC Assembler

- Se utiliza la siguiente fórmula:

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

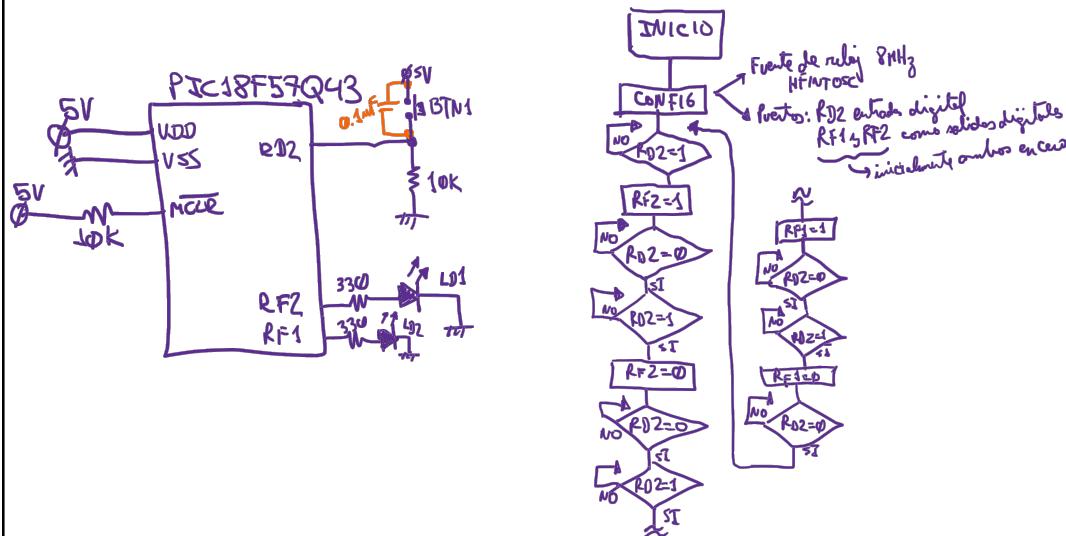
- Hay instrucciones simples, dobles y especiales (revisar 44.0 de la datasheet)
- Recordando la relación periodo vs frecuencia: $f = \frac{1}{T}$

Ejemplo de tiempo de ejecución



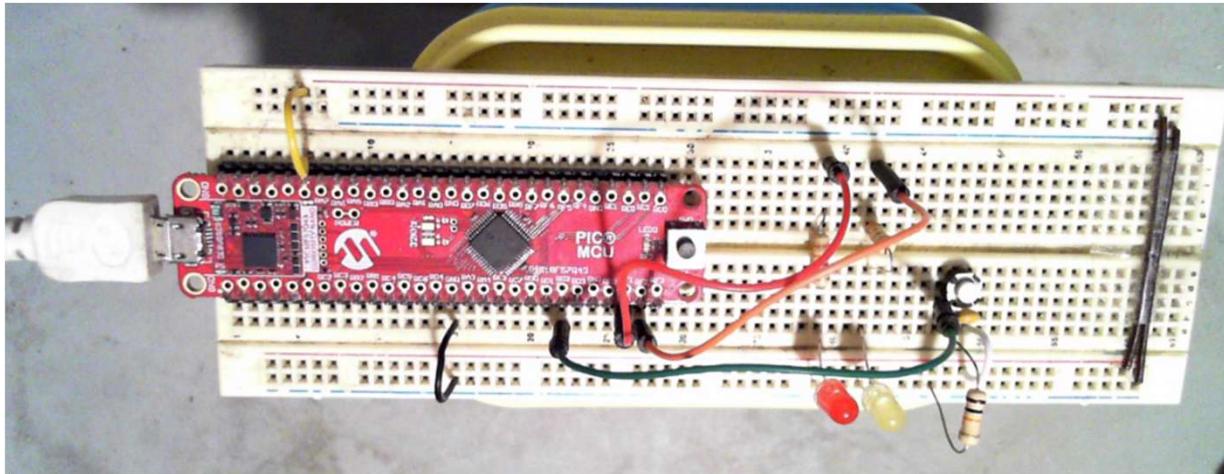
Ejemplo 02: Desarrollar un dispositivo LATCH con el Curiosity Nano PIC18F57Q43

- 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

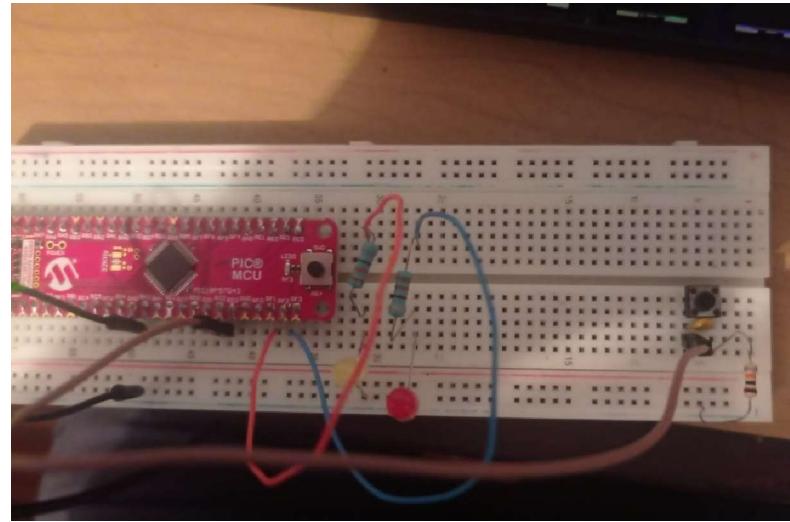
```

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 OSCEN, 1          ;HFINTOSC habilitado
17   movlb 4H               ;al BANK4
18   bcf 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   bra inicio              ;falso, no presione el boton y vuelvo a preguntar
29   bsf LATF, 1, 1          ;verdad, enciendo primer LED
30
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             ;verdad, apago el LED
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             ;verdad, apago el LED
53     bra inicio              ;regreso a label inicio
54
55   end upcino
56

```

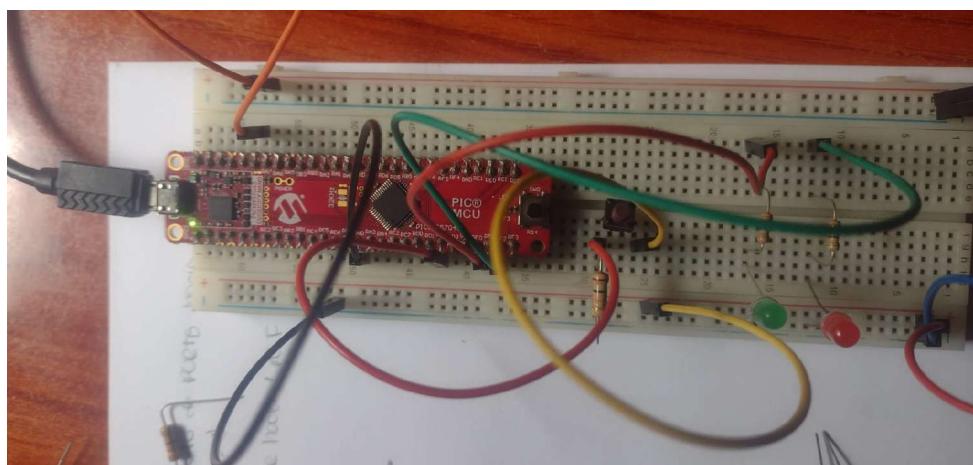
Error en la implementación

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



Error en la implementación

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

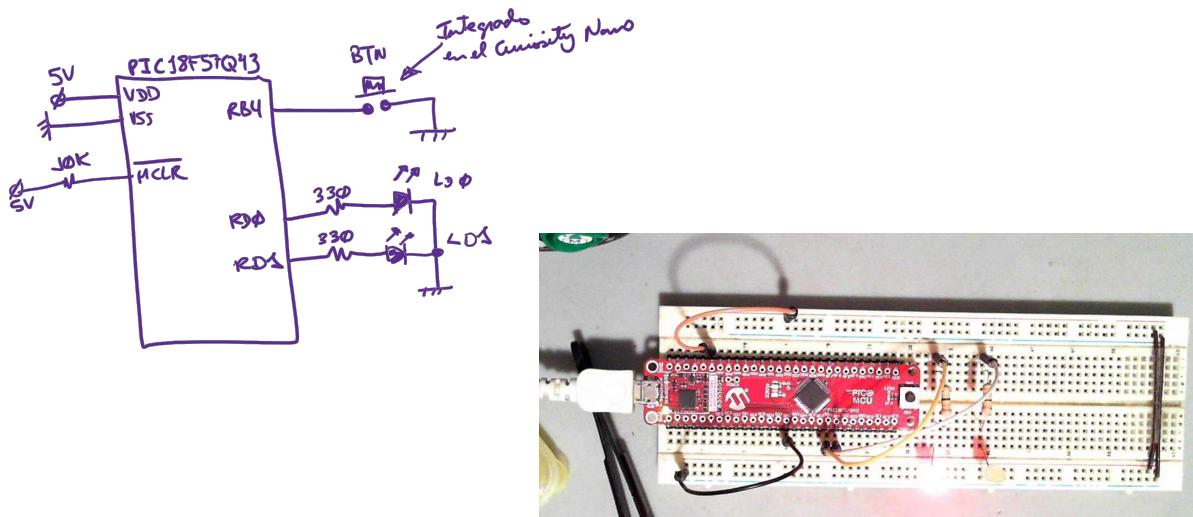


Problemas encontrados

- 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 ANSELD, 1         ;RD0 y RD1 como digitales
26     movlw 01H
27     movwf LATD, 1           ;RD0 encendido y RD1 apagado

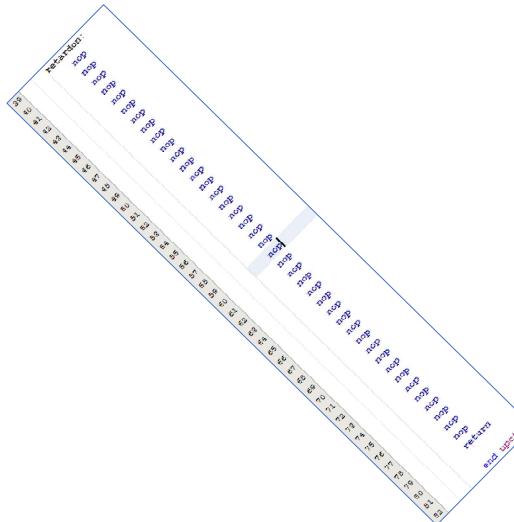
29     inicio:
30         btfsc PORTB, 4, 1  ;Pregunta si RB4 es cero (si presione el boton)
31         bra inicio          ;falso, regresa a preguntar
32         comf LATD, 1, 1      ;complemento a registro
33         call retardon
34     otro:
35         btfss PORTB, 4, 1  ;Pregunta si RB4 es uno (si solte el boton)
36         bra otro             ;falso, vuelvo a preguntar
37         bra inicio

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

```

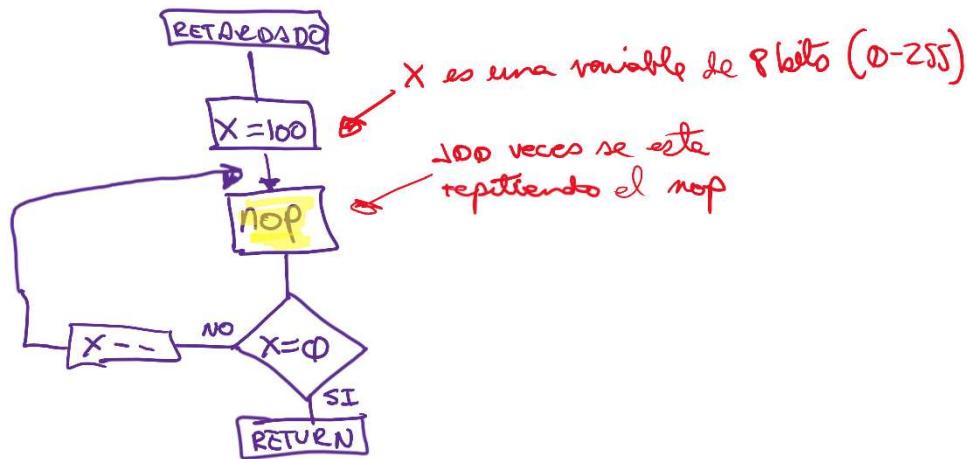
¿Cómo hago el antirrebote del botón sin tener que escribir tanto “nops”?

- Escribir 40000 nops no es práctico ni eficiente!



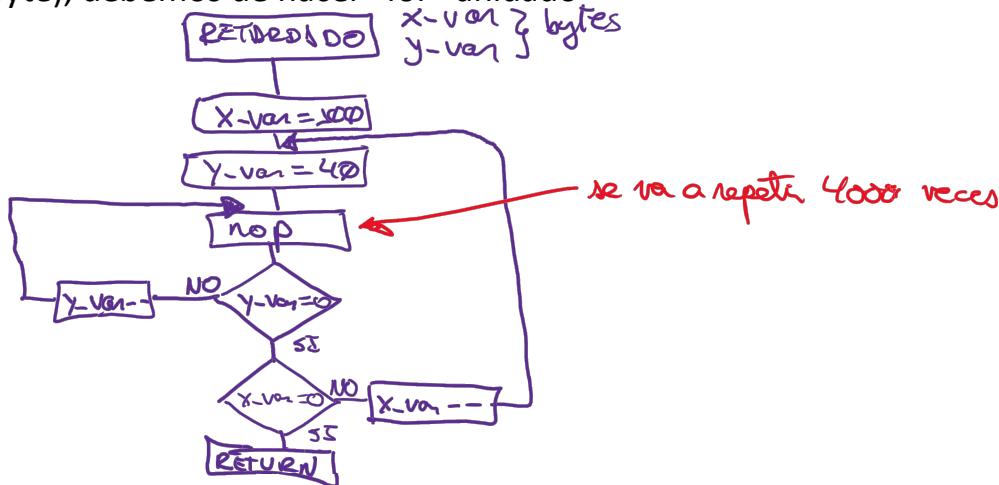
¿Cómo generamos el retardo de 40ms para el antirrebote del botón?

- 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?

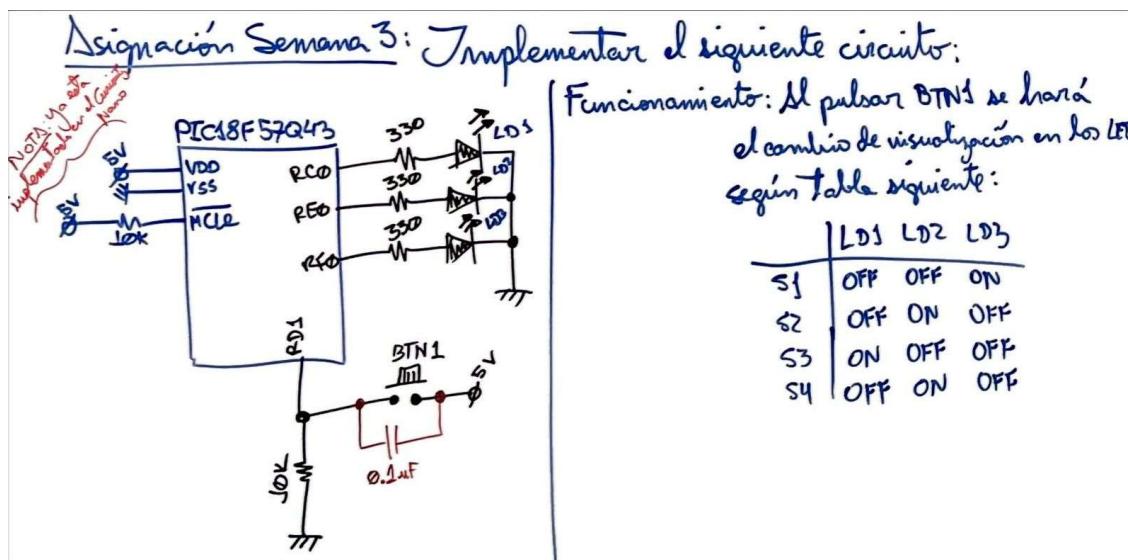
- **x_var** y **y_var** son etiquetas asignadas a posiciones GPR de la memoria de datos, recordando que los GPR empiezan a partir de la dirección 500H
- El código fue obtenido del diagrama de flujo anterior, tener en cuenta que al usar BSR se tiene que estar cambiando de bancos ya que el registro STATUS se encuentra en el BANK4 (4D8H) y los GPRs **x_var** y **y_var** están ubicados en BANK5 (500H y 501H respectivamente)
- 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
45  otro3:
46      movlw 40
47      movwf y_var, 1
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

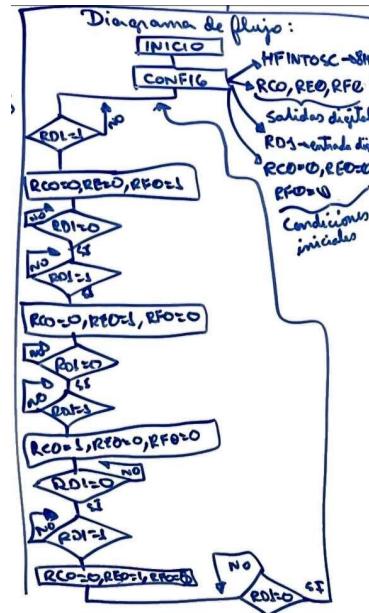
```

Ejemplo 2024-1



Ejemplo 2024-1

- Diagrama de flujo:



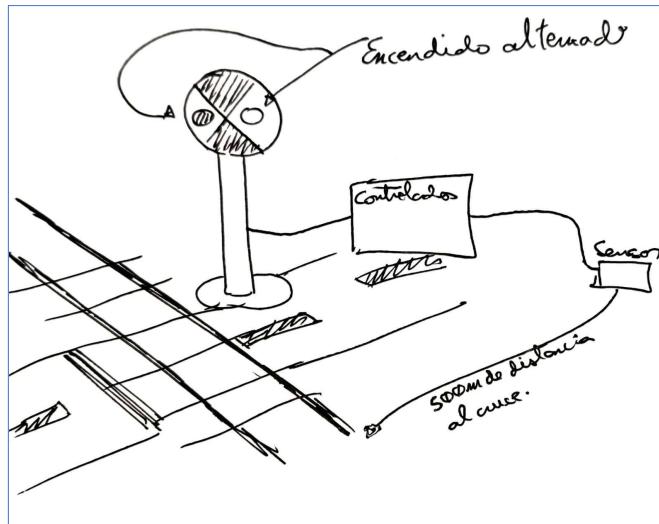
1	PROCESSOR 18F57Q43	;directiva de procesador
2	#include "cabecera.inc"	
3		
4	PSECT upcino, class=CODE, reloc=2, abs	;apertura de program section
5	upcino:	;etiqueta upcino
6	ORG 00000H	;vector de reset
7	bra configuro	;salto a etiqueta configuro
8		
9	ORG 000100H	;zona de programa de usuario
10	configuro:	;etiqueta configuro
11	movlw 0H	;al BANK0
12	movlw 60H	
13	movwf OSCCON1, 1	;NOSC=110 (HFINTOSC), NDIV=0000 (1:1)
14	movlw 03H	
15	movwf OSCFRQ, 1	;HFINTOSC a 8MHz
16	movlw 40H	
17	movwf OSCEN, 1	;HFINTOSC enabled
18	movlw 4H	;al BANK4
19	baf TRISD, 1, 1	;RDI como entrada
20	baf ANSEL0, 1, 1	;RDI como digital
21	baf TRISC, 0, 1	;RC0 como salida
22	baf ANSELC, 0, 1	;RC0 como digital
23	baf TRISE, 0, 1	;REO como salida
24	baf ANSEL0, 0, 1	;REO como digital
25	baf TRISE, 0, 1	;RFO como salida
26	baf ANSELF, 0, 1	;RFO como digital
27	baf LATC, 0, 1	;RC0 en cero
28	baf LATE, 0, 1	;REO en cero
29	baf LATF, 0, 1	;RFO en cero
30		
31	inicio:	
32	btfss PORTD, 1, 1	;Pregunto si pulse el BTN1
33	bra inicio	;no pulse BTN1 y me regreso
34	baf LATC, 0, 1	;RC0 en cero
35	baf LATE, 0, 1	;REO en cero
36	baf LATF, 0, 1	;RFO en uno
37	otro1:	
38	btfsc PORTD, 1, 1	;Pregunto si deje de pulsar BTN1
39	bra otro1	;sigue BTN1 pulsado y regreso
40	otro2:	
41	btfss PORTD, 1, 1	;Pregunto si pulse el BTN1
42	bra otro2	;no pulse BTN1 y me regreso
43	baf LATC, 0, 1	;RC0 en cero
44	baf LATE, 0, 1	;REO en uno
45	baf LATF, 0, 1	;RFO en cero
46	otro3:	
47	btfsc PORTD, 1, 1	;Pregunto si deje de pulsar BTN1
48	bra otro3	;sigue BTN1 pulsado y regreso
49	otro4:	
50	btfss PORTD, 1, 1	;Pregunto si pulse el BTN1
51	baf LATC, 0, 1	;no pulse BTN1 y me regreso
52	baf LATE, 0, 1	;RC0 en uno
53	baf LATF, 0, 1	;REO en cero
54	otro5:	
55	btfsc PORTD, 1, 1	;Pregunto si deje de pulsar BTN1
56	bra otro5	;sigue BTN1 pulsado y regreso
57	otro6:	
58	btfss PORTD, 1, 1	;Pregunto si pulse el BTN1
59	bra otro6	;no pulse BTN1 y me regreso
60	baf LATC, 0, 1	;RC0 en cero
61	baf LATE, 0, 1	;REO en uno
62	baf LATF, 0, 1	;RFO en cero
63	otro7:	
64	btfsc PORTD, 1, 1	;Pregunto si deje de pulsar BTN1
65	bra otro7	;sigue BTN1 pulsado y regreso
66	baf LATC, 0, 1	
67	baf LATF, 0, 1	
68	end upcino	

Ejemplo 2024-1

- Código:

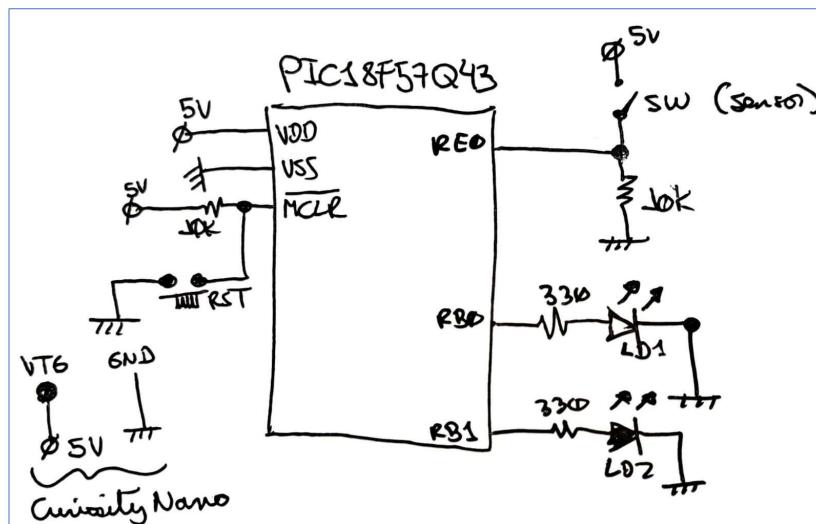
Ejemplo 2 – 2024-2

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



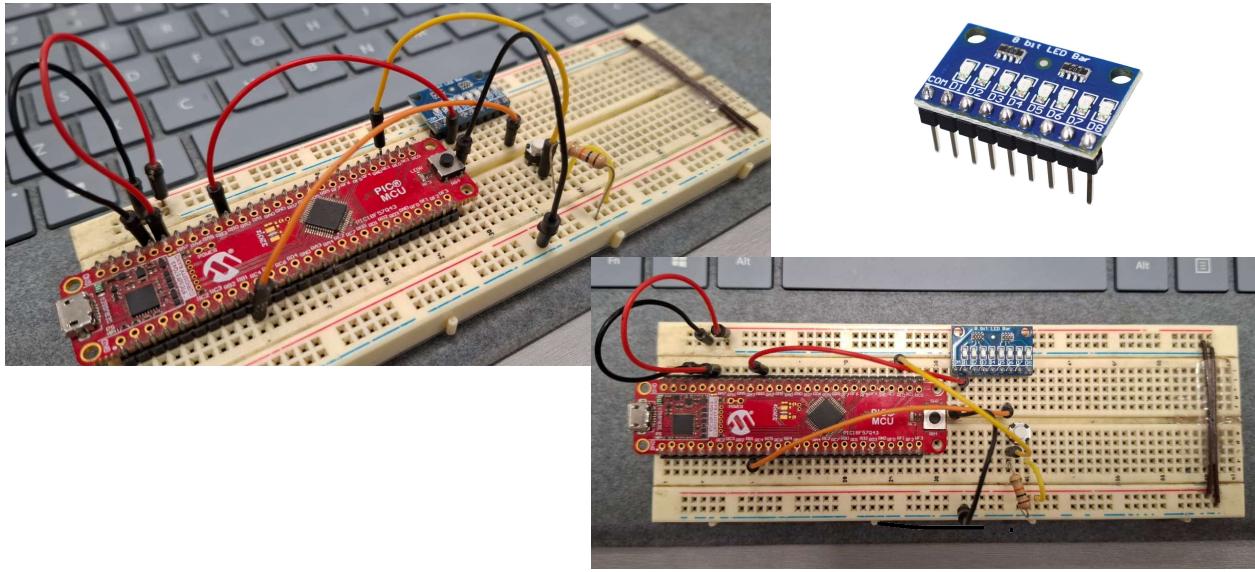
Ejemplo 2 – 2024-2

- Diseño del hardware – Diagrama esquemático



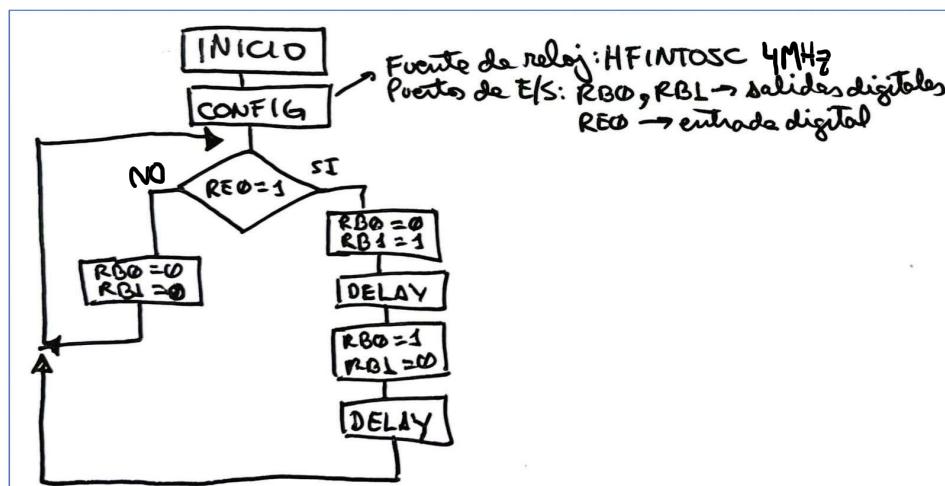
Ejemplo 2 – 2024-2

- Diseño del hardware – Implementación



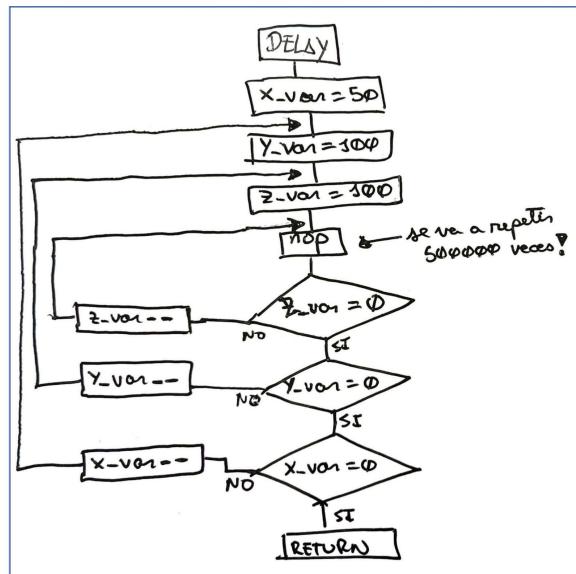
Ejemplo 2 – 2024-2

- 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  ;asignacion de nombres a los registros GPR para la rutina de retardo
5  x_var EQU 500H
6  y_var EQU 501H
7  z_var EQU 502H
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 ANSELB, 0, 1   ;REO como digital
28     ;opcion 1 manipulacion individual de bits
29     ;bcf TRISE, 0, 1   ;RB0 como salida
30     ;bcf TRISB, 1, 1   ;RB1 como salida
31     ;bcf ANSELB, 0, 1   ;RB0 como digital
32     ;bcf ANSELB, 1, 1   ;RB1 como digital
33     ;opcion 2 manipulacion numerica al registro
34     movlw 0FCH
35     movwf TRISB, 1      ;RB0 y RB1 como salidas
36     movwf ANSELB, 1      ;RB0 y RB1 como digitales
37
38 inicio_train:
39     btfss PORTE, 0, 1  ;Pregunto si RE0 es uno
40     bra es_falso        ;Falso, salta a label es_falso
41     movlw 01H            ;Verdad (efecto de luces del cruce de tren)
42     movwf LATB, 1         ;RB1 apagado, RB0 encendido
43     call retardo        ;llamo rutina de retardo
44     movlw 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 es_falso:
49     clrf LATB, 1          ;todo el RB en cero (apagado RB1 y RB0)
50     bra inicio_train    ;salta a label inicio_train
51
52 retardo:
53     movlb 5H              ;nos vamos al Bank5
54     movlw 50
55     movwf x_var, 1
56 punto_2:
57     movlw 50
58     movwf y_var, 1
59 punto_1:
60     movlw 50
61     movwf z_var, 1
62 punto_0:
63     nop                  ;se va a repetir 125000 veces
64     movlw 0
65     cpfseq z_var, 1
66     bra falso_z
67     movlw 0
68     cpfseq y_var, 1
69     bra falso_y
70     movlw 0
71     cpfseq x_var, 1
72     bra falso_x
73     movlb 4H              ;regresamos al Bank0
74     return
75
76     falso_z:
77     deef z_var, 1, 1
78     bra punto_0
79     falso_y:
80     deef y_var, 1, 1
81     bra punto_1
82     falso_x:
83     deef x_var, 1, 1
84     bra punto_2
85 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        btfss PORTE, 0, 1 ;pregunto si RE0 es 1
3      ;Asignacion de labels a GPR 36        bra es_falsozao ;viene aqui cuando es Falso
4      x_var EQU 500H              37        bra es_verdad ;viene aqui cuando es Verdadero
5      y_var EQU 501H              38        es_verdad:
6      z_var EQU 502H              39            bsf LATB, 0, 1 ;enciende RB0
7      ;PSECT upcino, class=CODE, reloc=2, abs 40            bcf LATB, 1, 1 ;apago RB1
8      upcino:                   41            call retardado ;llamada a subrutina retardado
9      ORG 000000H ;vector de RESET 42            bcf LATB, 0, 1 ;apago RB0
10     bra configuro ;salto a label configuro 43            bsf LATB, 1, 1 ;enciende RB1
11     ORG 000100H ;zona de programa de usu 44            call retardado ;llamada a subrutina retardado
12     configuro:                45            brc inicio_tren ;retorno al inicio_tren
13     ;configuracion la fuente de reloj 46        es_falsozao:
14     movlb 0H ;nos vamos al Bank0 47            bcf LATB, 0, 1 ;apago RB0
15     movlw 60H                  48            bra inicio_tren ;retorno al inicio_tren
16     movwf OSCCON1, 1 ;NOSC=HFINTOSC, NDIV 49            es_verdad:
17     movlw 02H                  50            movlw 50
18     movwf OSCFRC, 1 ;HFINTOSC a 4MHz 51            movwf x_var, 1 ;x_var valor d
19     movlw 40H                  52            punto_tres:
20     movwf OSCCEN, 1 ;HFINTOSC habilitado 53            movlw 50
21     ;configuracion las E/S 54            movwf y_var, 1 ;y_var valor d
22     movlb 4H ;saltamos al Bank4 55            punto_dos:
23     bcf TRISE, 0, 1 ;RE0 es entrada 56            movlw 50
24     bcf ANSELB, 0, 1 ;RB0 es digital 57            movwf z_var, 1 ;z_var valor d
25     bcf ANSELB, 1, 1 ;RB1 es digital 58            punto_uno:
26     nop ;instruccion no operacion 59            btfss STATUS, 2, 1 ;pregunto si Z=1
27     movf z_var, 1, 1 ;evaluar z_var 60            bra z_var_noescero ;Falso salta a label z_var_noescero
28     btfss STATUS, 2, 1 ;pregunto si Z=1 61            movf x_var, 1, 1 ;evaluo x_var
29     bra z_var_noescero ;Falso salta a label x_var_noescero 62            btfss STATUS, 2, 1 ;preguntar si Z=1
30     ;Recordando bcf [registro], #bit, access 63            bra x_var_noescero ;Falso salta label x_var_noescero
31     btfss STATUS, 2, 1 ;pregunto si Z=1 64            return
32     movf y_var, 1, 1 ;Verdad, evaluar y_var 65            z_var_noescero:
33     btfss STATUS, 2, 1 ;preguntar si Z=1 66            decf z_var, 1, 1 ;decrementamos z_var
34     movf y_var, 1, 1 ;evaluar y_var 67            bra punto_uno ;salto a label punto_uno
35     es_falsozao:               68            y_var_noescero:
36     clrf LATB, 1 ;apagamo 69            decf y_var, 1, 1 ;decrementamos y_var
37     bra inicio_tren ;retorn 70            bra punto_dos ;salto a label punto_dos
38     es_verdad:                71            x_var_noescero:
39     movlw 01H                  72            decf x_var, 1, 1 ;decremeno x_var
40     movwf LATB, 1 ;RB1=0, 73            bra punto_tres ;salto a label punto_tres
41     call retardado ;llamada 74            end upcino
42     movlw 02H                  75
43     movwf LATB, 1 ;RB1=1, 76
44     call retardado ;llamada 77
45     bra inicio_tren ;retorn 78
46     retardado:               79
47     movlb 5H ;al Bank5 80
48     movlw 30                  81
49     movwf x_var, 1 82
50     punto_tres:               83
51     movlw 30                  84
52     movwf y_var, 1 85
53     punto_dos:               86
54     movlw 30                  87
55     movwf z_var, 1 88
56     punto_ultimo:             89
57     nop 90
58     movf z_var, 1, 1 91
59     movlb 4H ;Al Bank4 92
60     btfss STATUS, 2, 1 ;pregunto si z_var llego a c 93
61     bra zvar_noescero ;falso zvar no llego a cero 94
62     movlb 5H ;Al bank5 95
63     movf y_var, 1, 1 ;verdad y prosigo a ls sigui 96
64     movlb 4H ;Al Bank4 97
65     btfss STATUS, 2, 1 ;pregunto si y_var llego a c 98
66     bra yvar_noescero ;falso yvar no llego a cero 99
67     movlb 5H ;Al bank5 100
68     movf x_var, 1, 1 101
69     movlb 4H ;Al Bank4 102
70     btfss STATUS, 2, 1 ;pregunto si x_var llego a c 103
71     bra xvar_noescero ;falso xvar no llego a cero 104
72     return 105
73     zvar_noescero: 106
74     decf z_var, 1, 1 ;decremento z_var 107
75     bra punto_ultimo 108
76     xvar_noescero: 109
77     movlb 5H ;Al bank5 110
78     decf y_var, 1, 1 ;decremento y_var 111
79     bra punto_dos 112
80     xvar_noescero: 113
81     movlb 5H ;Al bank5 114
82     decf x_var, 1, 1 ;decremtno x_var 115
83     bra punto_tres 116
84     end upcino 117

```

Ejemplo 2 – 2024-2

- Diseño de software: Codificación en lenguaje XC8 PIC Assembler

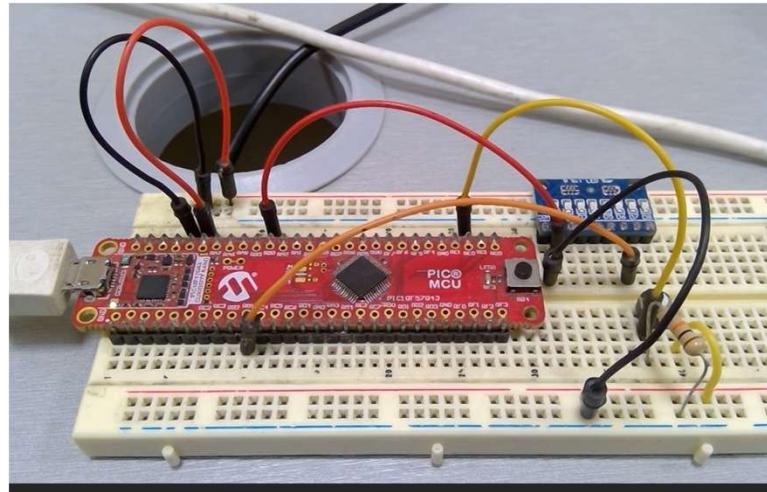
```

1      PROCESSOR 18F57Q43          29    bcf ANSELB, 0, 1 ;RE0 com 57    nop
2      #include "cabecera.inc"    30        movf z_var, 1, 1 58
3      ;Asignacion de labels a GPR 31        movlb 4H ;Al Bank4
4      x_var EQU 500H              32        btfss PORTB, 0, 1 ;pregunto si z_var llego a c 59
5      y_var EQU 501H              33        bra zvar_noescero ;falso zvar no llego a cero 60
6      z_var EQU 502H              34        bra es_verdad ;viene a 61
7      ;PSECT upcino, class=CODE, reloc=2, abs 35        es_verdad:
8      upcino:                   36        clrf LATB, 1 ;apagamo 62
9      ORG 000000H ;vector de reset 37        bra inicio_tren ;retorn 63
10     bra configuro ;salto a label configuro 38        es_falso:
11     ORG 000100H ;zona de usuario 39        movlw 01H 64
12     configuro:                40        btfss PORTB, 0, 1 ;pregunto si y_var llego a c 65
13     ;configuracion de la fuente de reloj 41        bra es_falso ;viene a 66
14     movlb 0H ;al Bank0 42        es_verdad:
15     movlw 60H                  43        movlw 01H 67
16     movwf OSCCON1, 1 ;NOSC=HFINTOSC NDIV 44        btfss PORTB, 0, 1 ;pregunto si x_var llego a c 68
17     movlw 02H                  45        call retardado ;llamada 69
18     movwf OSCFRC, 1 ;HFINTOSC 4MHz 46        movlw 02H 70
19     movlw 40H                  47        movwf LATB, 1 ;RB1=0, 71
20     movwf OSCCEN, 1 ;HFINTOSC enabled 48        call retardado ;llamada 72
21     ;configuracion las E/S 49        movwf LATB, 1 ;RB1=1, 73
22     movlb 4H ;al Bank4 50        retardado:
23     movlw 0xfc ;en binario 1111110 51        movlb 5H ;al Bank5 74
24     movwf TRISB, 1 ;RB0 y RB1 como sal 52        movlw 30 75
25     movwf ANSELB, 1 ;RB0 y RB1 como dig 53        movwf x_var, 1 76
26     bcf TRISE, 0, 1 ;RE0 como entrada 54        punto_tres:
27     btfss STATUS, 2, 1 ;pregunto si Z=1 55        movlw 30 77
28     movf z_var, 1, 1 ;evaluar z_var 56        movwf y_var, 1 78
29     btfss STATUS, 2, 1 ;pregunto si Z=1 57        punto_dos:
30     movf y_var, 1, 1 ;evaluar y_var 58        movlw 30 79
31     btfss STATUS, 2, 1 ;pregunto si Z=1 59        movwf z_var, 1 80
32     movf x_var, 1, 1 ;evaluar x_var 60        punto_ultimo:
33     btfss STATUS, 2, 1 ;pregunto si Z=1 61        nop 81
34     movf z_var, 1, 1 ;evaluar z_var 62        end upcino 82
35     es_falso:                 63
36     clrf LATB, 1 ;apagamo 64
37     bra inicio_tren ;retorn 65
38     es_verdad:               66
39     movlw 01H 67
40     btfss PORTB, 0, 1 ;pregunto si y_var llego a c 68
41     bra es_falso ;viene a 69
42     es_verdad:               70
43     movlw 01H 71
44     btfss PORTB, 0, 1 ;pregunto si x_var llego a c 72
45     call retardado ;llamada 73
46     retardado:               74
47     movlb 5H ;al Bank5 75
48     movlw 30 76
49     movwf LATB, 1 ;RB1=0, 77
50     call retardado ;llamada 78
51     movwf LATB, 1 ;RB1=1, 79
52     retardado:               80
53     movlb 5H ;al Bank5 81
54     movlw 30 82
55     movwf x_var, 1 83
56     punto_dos:               84
57     movlw 30 85
58     movwf y_var, 1 86
59     punto_tres:               87
60     movlw 30 88
61     movwf z_var, 1 89
62     punto_ultimo:             90
63     nop 91
64     zvar_noescero: 92
65     decf z_var, 1, 1 ;decremento z_var 93
66     bra punto_ultimo 94
67     yvar_noescero: 95
68     decf y_var, 1, 1 ;decremento y_var 96
69     bra punto_dos 97
70     xvar_noescero: 98
71     decf x_var, 1, 1 ;decremtno x_var 99
72     bra punto_tres 100
73     end upcino 101

```

Ejemplo 2 – 2024-2

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



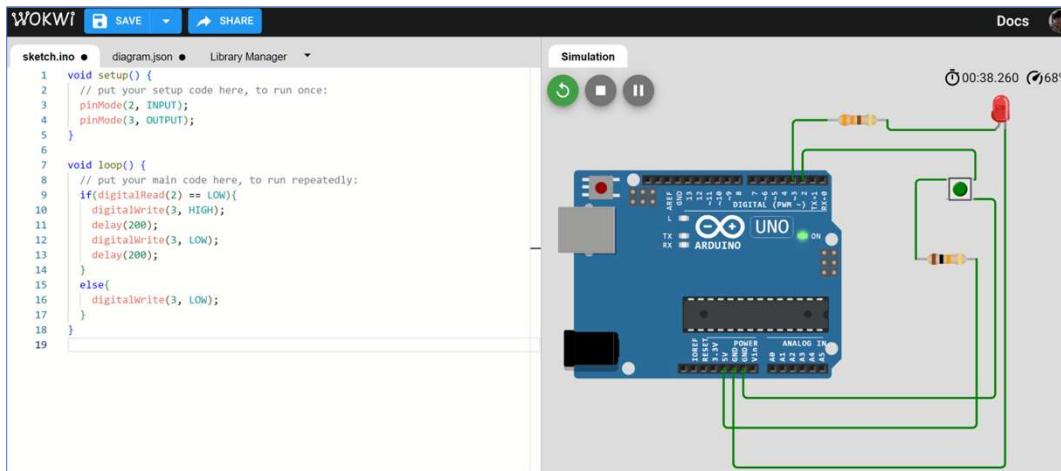
Ejemplo 2 – 2024-2

- 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.

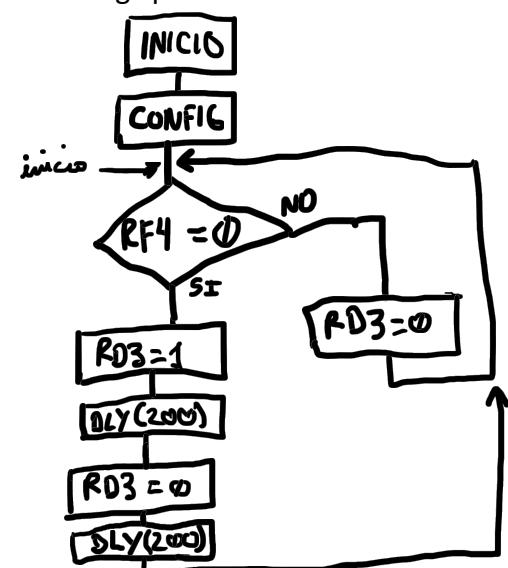
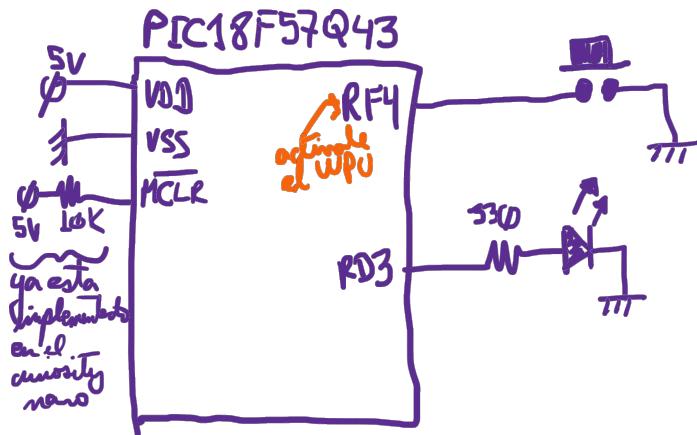
En Arduino:



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:

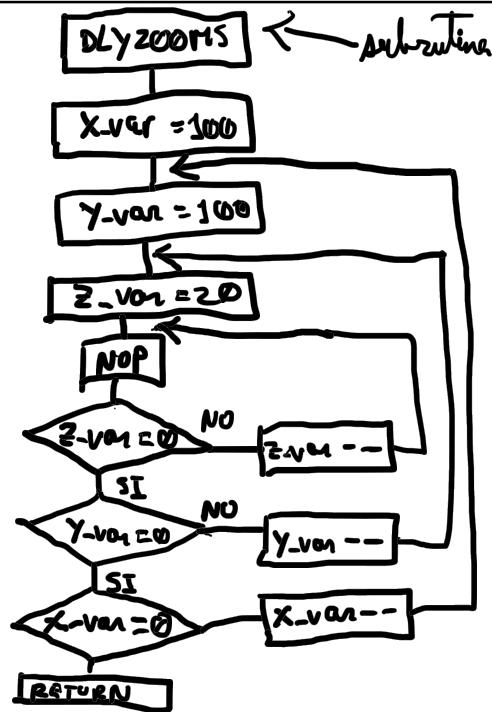


Ejemplo 2 (2025-1)

- 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.
Tú necesitas 200000us,
200000 nops?

Tenemos que construir una
estructura de repetición
anidada para ese nop

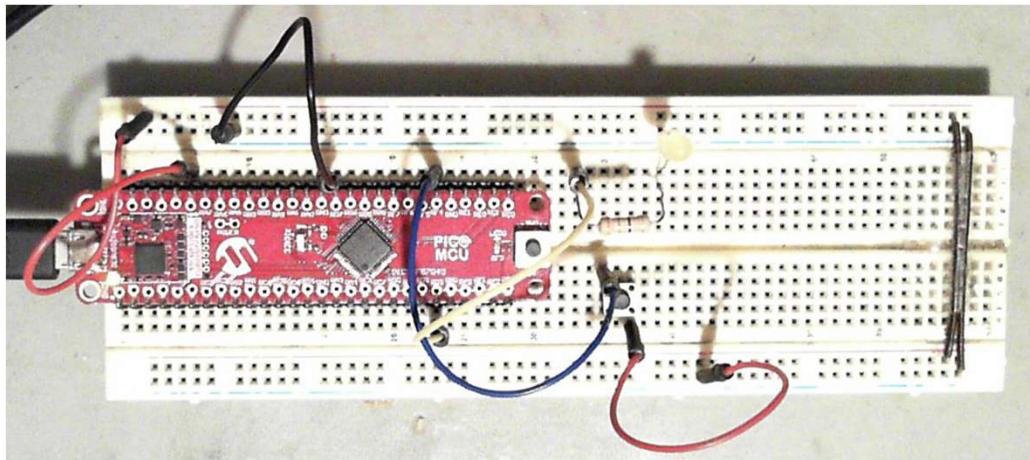


Ejemplo 2 (2025-1)

- 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 18F57043 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 ;Configuración del modulo de oscilador 18 movlb 0H 19 movlw 60H 20 movwf OSCCON1, 1 ;NOSC=HFINTOSC, NDIV 21 movlw 02H 22 movwf OSCFRC, 1 ;HFINTOSC a 4MHz 23 bsf OSCEN, 6, 1 ;HFINTOSC enabled 24 25 ;Configuración 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 bsf WPUF, 4, 1 ;RF4 pullup enabled 31 ;RD3 como salida digital 32 bcf TRISD, 3, 1 ;RD3 salida 33 bcf ANSELD, 3, 1 ;RD3 digital </pre>	<pre> 35 inicio: 36 btfc PORTF, 4, 1 ;pregunto si presione el botón 37 brcf Z_VAR, 1, 1 ;falso, salta a noperación 38 bra noperación 39 40 noperación: 41 bcf LATD, 3, 1 ;apago el LED 42 bra inicio 43 44 siperación: 45 bsf LATD, 3, 1 ;enciendiendo el LED 46 call dly200ms 47 bcf LATD, 3, 1 ;apago el LED 48 call dly200ms 49 bra inicio 50 51 ;subrutina de retardo 52 dly200ms: 53 movlb 5H 54 movlw 50 55 movwf Z_VAR, 1 56 57 llegada_uno: 58 movlw 50 59 movwf Y_VAR, 1 60 61 llegada_dos: 62 movlw 20 63 movwf Z_VAR, 1 64 65 llegada_tres: 66 nop 67 ;pregunta si z_var llegó a cero 68 movf Z_VAR, 1, 1 69 movlb 4H 70 btfss STATUS, 2, 1 ;pregunta si se levantó bandera Z 71 bra noselevantol 72 movlb 5H 73 ;pregunta si y_var llegó a cero 74 movf Y_VAR, 1, 1 75 movlb 4H 76 btfss STATUS, 2, 1 ;pregunta si se levantó bandera Z 77 bra noselevantol2 78 movlb 5H 79 ;pregunta si x_var llegó a cero 80 movf X_VAR, 1, 1 81 movlb 4H 82 btfss STATUS, 2, 1 ;pregunta si se levantó bandera Z 83 bra noselevantol3 84 return 85 86 noselevantol: 87 movlb 5H 88 decf Z_VAR, 1, 1 ;decremento de z_var 89 bra llegada_tres 90 91 noselevantol2: 92 movlb 5H 93 decf Y_VAR, 1, 1 ;decremento de y_var 94 bra llegada_dos 95 96 noselevantol3: 97 movlb 5H 98 decf X_VAR, 1, 1 ;decremento de x_var 99 bra llegada_uno 100 101 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:

- 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
- Verificar que el proyecto creado en el MPLABX se haya seleccionado el Curiosity Nano PIC18F57Q43 (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 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

Ejercicios adicionales:

- Desarrollar un titilador de un LED conectado en RE0 en el cual su periodo de parpadeo dependerá del estado de un switch conectado en RB0, si RB0=1 el periodo será de 500ms, si RB0=0 el periodo será de 100ms.
- Desarrollar una señal de cruce de tren con entrada de activación mediante el uso de un switch.
- Desarrollar una “vela electrónica” con entrada de activación, dicha entrada tendrá como sensor de luz a un L.D.R.

Fin de la sesión