

Microcontroladores

Semana 3

Semestre 2024-2

Profesor: Kalun José Lau Gan

1

Preguntas previas:

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

movlw 0x10
movlw A0H
movlw 0A0H ✓

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)

2

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.

3

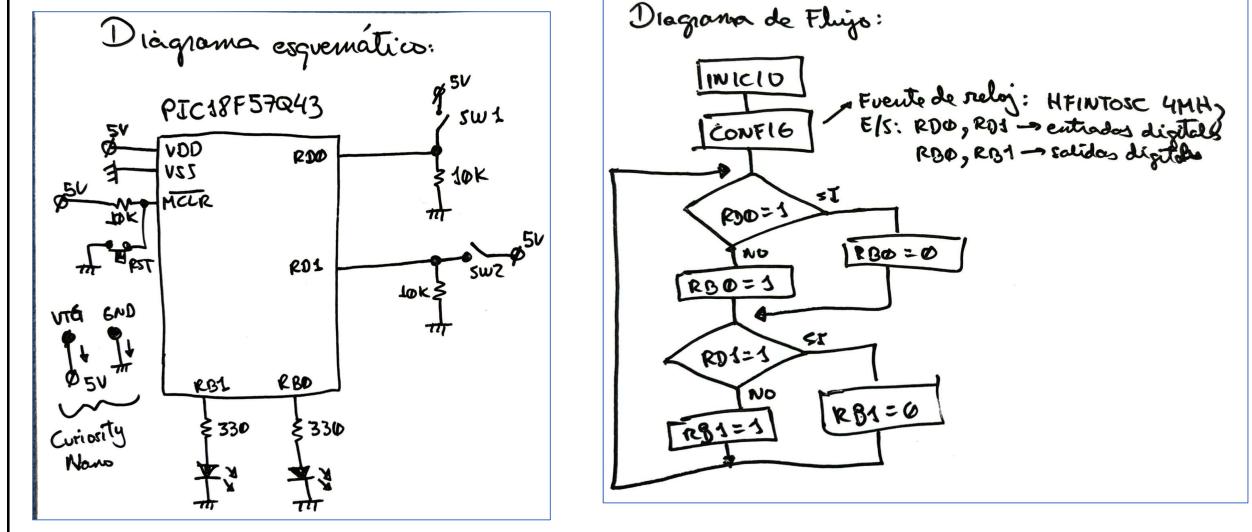
Preguntas previas:

- 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.

4

Preguntas previas:

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



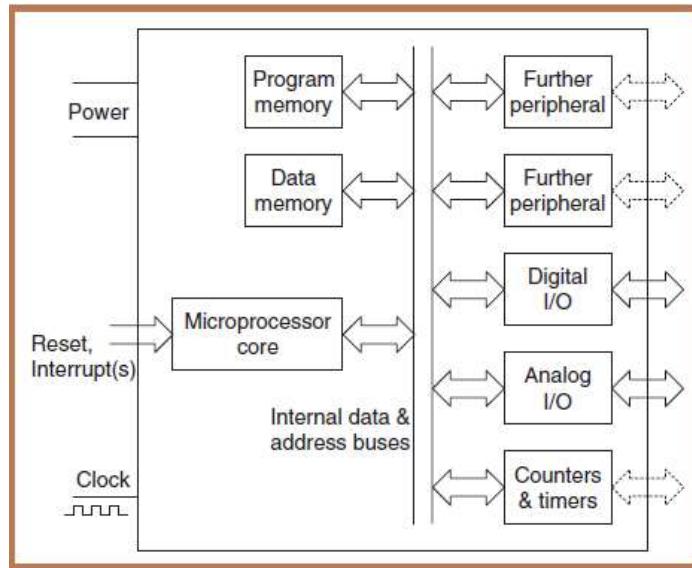
5

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

6

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

7

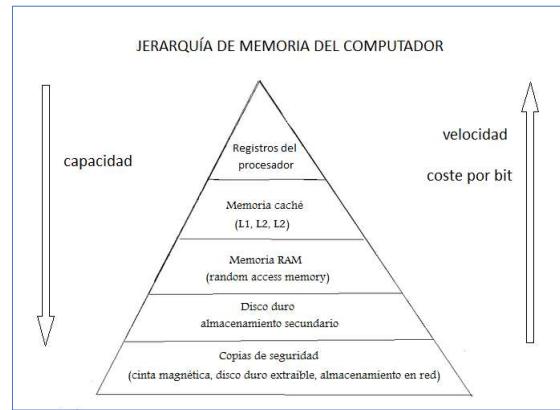
¿Cuál es la diferencia entre microprocesador y microcontrolador?

- **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.

8

¿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.



9

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-3FFFFH (22 bits – 4Mbyte de direcciones)

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 IDs (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 002Fh	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 FFFDh	Revision ID (1 Word) ^(5,A,B)
3F FFFF Eh to 3F FFFFh	Device ID (1 Word) ^(5,A,C)

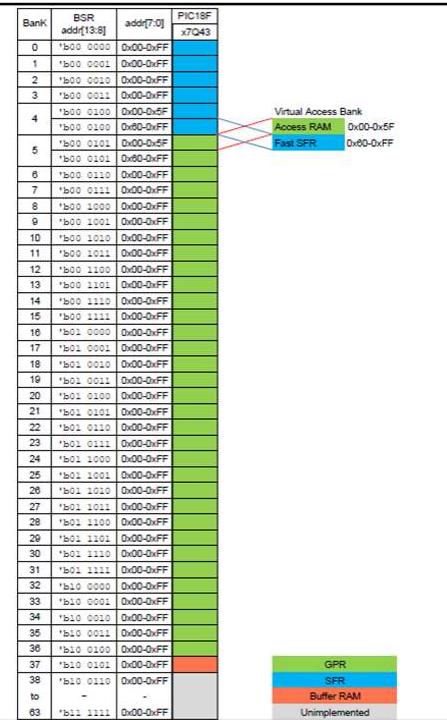
Notes:
 1. Storage Area Flash is implemented as the last 128 Words of User Flash, if enabled.
 2. The addresses do not roll over. The region is read as '0'.
 3. Not code-protected.
 4. Hard-coded in silicon.
 5. This region cannot be written by the user and it is not affected by a Bulk Erase.

10

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			



11

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 ANSELE, 1, 1 ;REO sea digital
```

12

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-var1=0, y-var1=0;
for (x-var1=0; x-var1<200; x-var1++) {
    for(y-var1=0; y-var1<200; y-var1++) {
        }                                } ¿Cuántos veces se repite? → 40000
    }
}

```

13

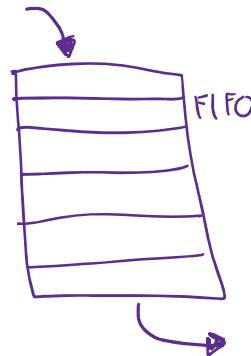
Recordando saltos a sub-rutinas

- ¿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.

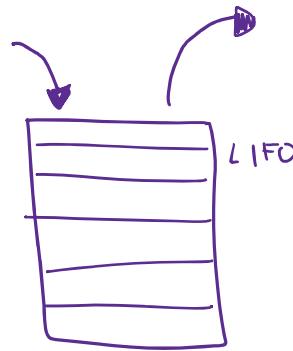
14

Memorias “Stack” o de pila (apilamiento)

- Estructura FIFO
(first in – first out)



- Estructura LIFO
(last in – first out)



15

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 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.

16

MPASM vs XC8 PIC Assembler: Partes de un programa

MPASM: <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>	XC8 PIC ASM: <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> <p>Nota: Los bits de configuración se alojaron en un archivo header llamado "cabecera.inc"</p>

17

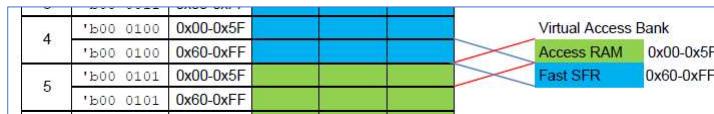
¿Por qué aprender Assembler (más difícil) si en el C también se puede hacer?

- En el assembler se prioriza en la eficiencia (espacio empleado en la memoria de programa y de datos, control detallado del consumo energético)
- 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)

18

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”

19

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	0001	11da	ffff	ffff	Z, N 1
DECf	f, d, a	Decrement f	1	0001	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
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
			0000	0000	01da	ffff	ffff	
MOVFFL	f _s , f _d	Move f _s (16-bit source) to f _d (destination)	3	1111	$\sum f_s$	$\sum f_d$	$\sum f_d$	None 1, 3
			1111	1111	$\sum f_s$	$\sum f_d$	$\sum f_d$	
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
RJRCF	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
RJRCF	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

20

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 ;muevas al Bank0
- movlw → mover un literal hacia el registro W
- movwf → mover el contenido de W hacia un registro (memoria de datos)
- bsf → setear un bit de un registro
- bcf → resetear un bit de un registro
- btfss → preguntar si un bit de un registro es igual a uno
- bra → salto

21

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

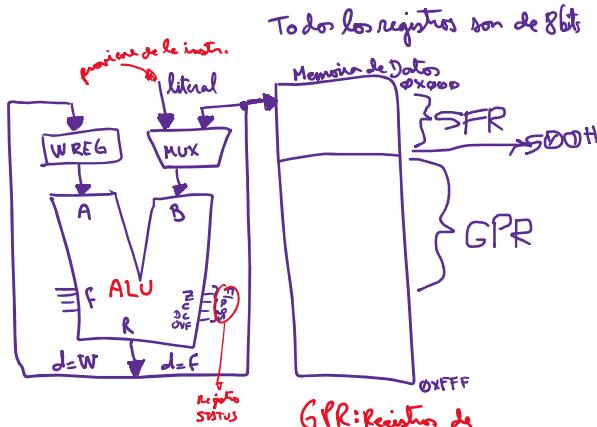
movwf TRISB, a *access bank*
movwf TRISB, b *BSR*

- 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	

22

El flujo de datos en el microcontrolador PIC18F57Q43



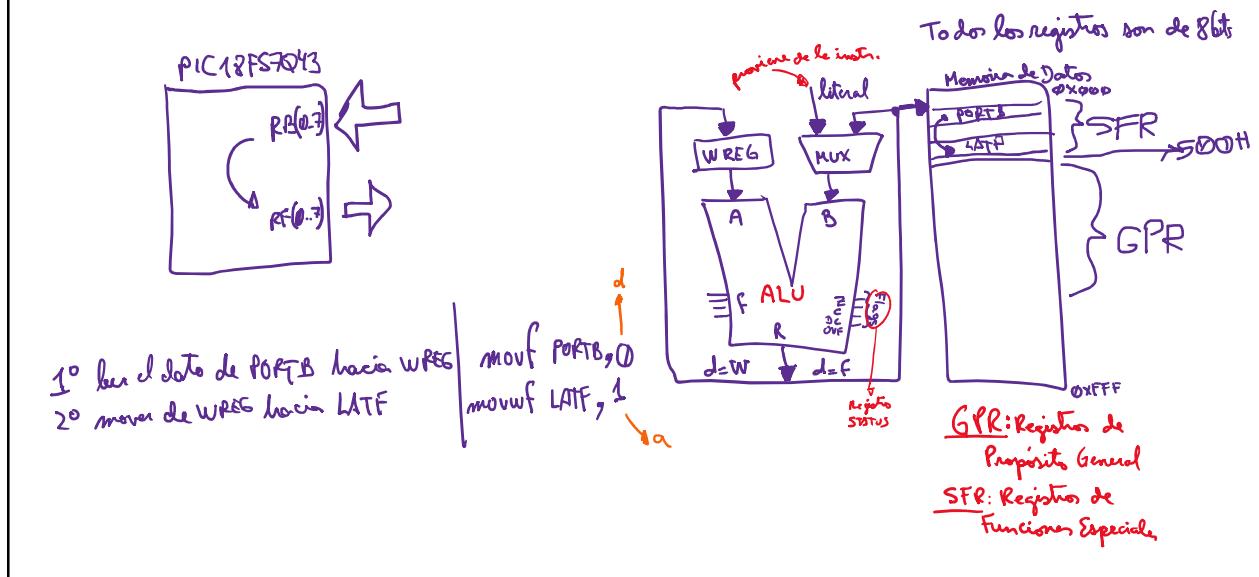
23

Comentarios y Notas:

- En los programas desarrollados en XC8 PIC Assembler, la mayor cantidad de instrucciones serán las de movimiento de datos.

24

Ejemplo: Pasar un dato del puerto B a al puerto F



25

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

26

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)

27

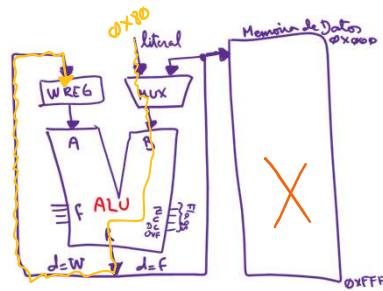
Instrucciones básicas en XC8 PIC Assembler

- Instrucciones de movimiento de datos

movlw [literal]

-mover un literal hacia WREG

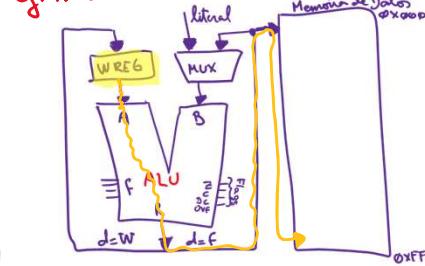
Ej: movlw 0x80



movwf [registro], a

-mover el contenido de WREG hacia [registro]

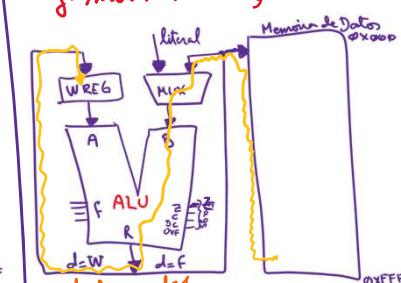
Ej: movwf TRISA



movf [registro], d, a

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

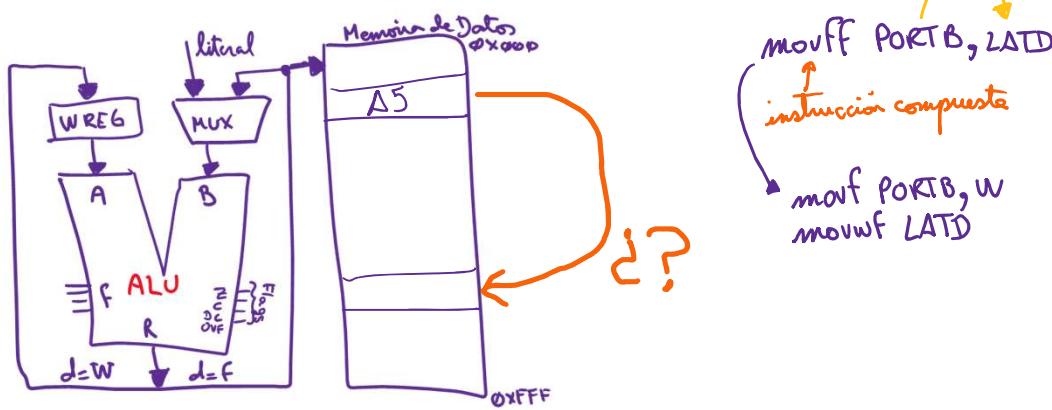
Ej: movf PORTC, W



28

Instrucción movff [registro1], [registro2]

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



29

¿Instrucción movfw?

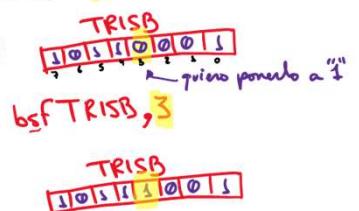
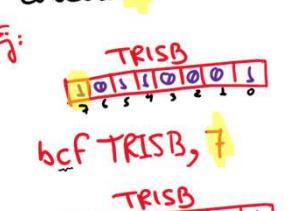
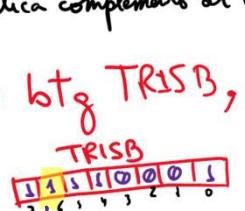
- ¡Instrucción no documentada en la hoja técnica!
- Instrucción “legacy”
- Esta no es una instrucción en sí, sino una “pseudo-instrucción” del lenguaje assembler.
- Lo que hace es mover el contenido de un registro y lo coloca en el Wreg.

Ej: movfw TRISA simila
 \rightarrow movf TRISA, w

30

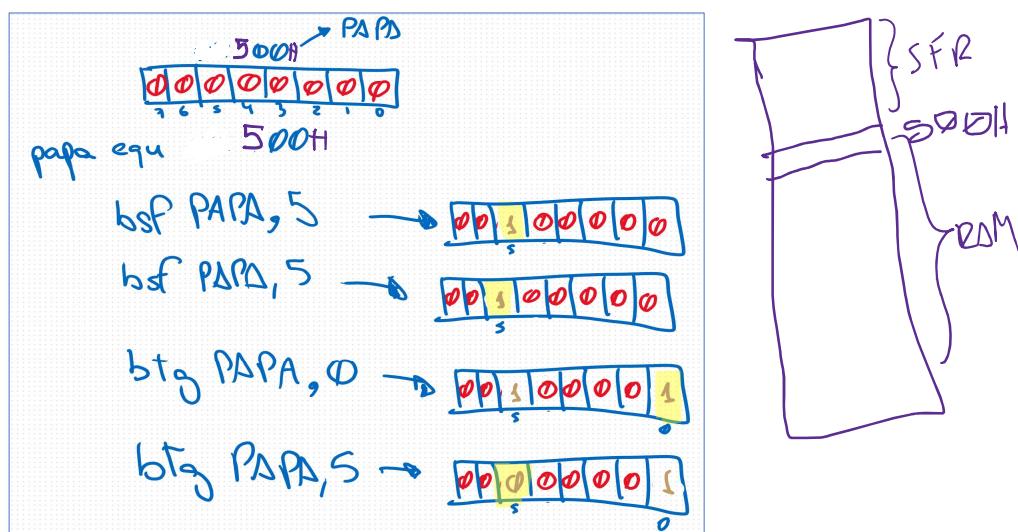
Instrucciones básicas en XC8 PIC Assembler

- 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:  </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>
--	--

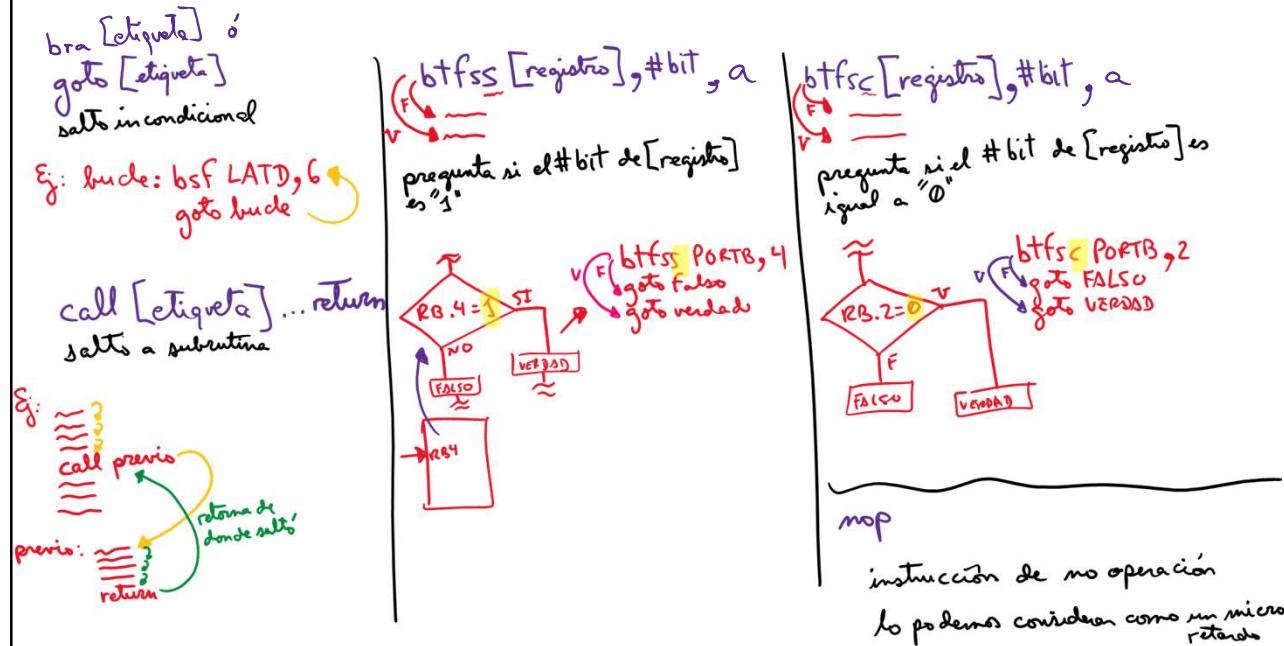
31

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



32

Instrucciones básicas en XC8 PIC Assembler

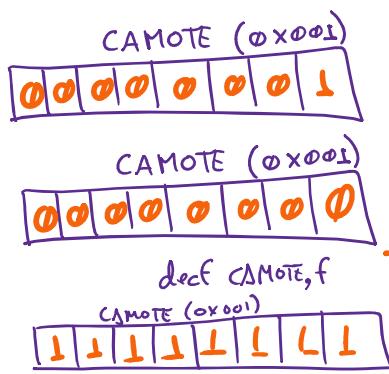


33

Instrucciones básicas en XC8 PIC Assembler

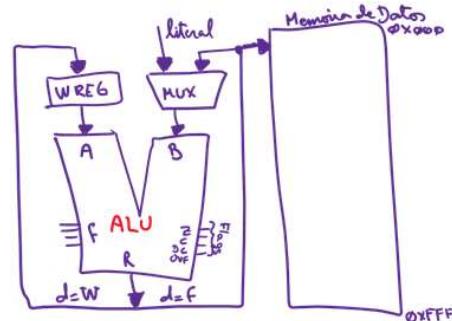
- Instrucción decf / incf
 - Decremento (decf) o incremento (incf) de registro, ambos de **uno en uno**

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



decf CAMOTE, f

Registro STATUS: Z=1



34

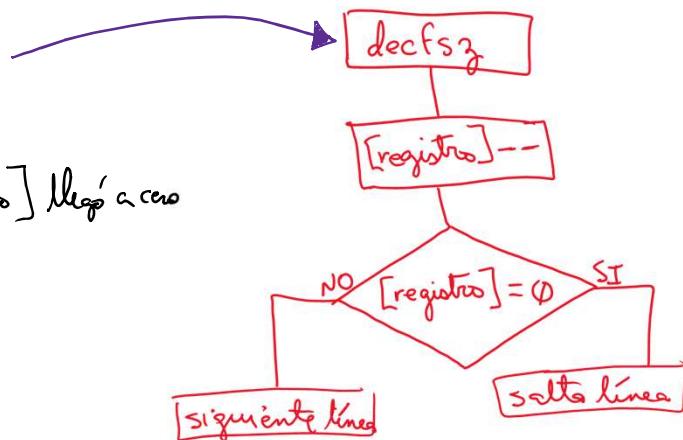
Instrucciones básicas en XC8 PIC Assembler

decfsz [registro], d

decrementa y pregunta si [registro] llegó a cero

incfsz [registro], d

incrementa y pregunta si [registro] llegó a cero



35

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

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

ejemplo: TRISB
10110001

setf TRISB

\Rightarrow TRISB
11111111

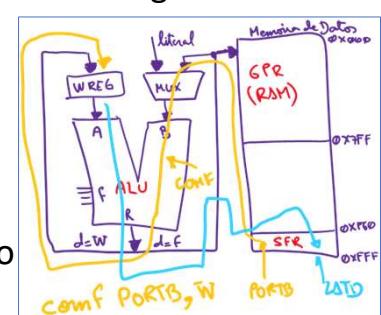
- **clrf [registro]** : Coloca todos los bits del registro indicado a cero lógico

ejemplo: TRISB
10110001

clrf TRISB

\Rightarrow TRISB
00000000

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



36

Tiempo de ejecución de las instrucciones en XC8 PIC Assembler

- Se utiliza la siguiente fórmula:

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

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

$$f = \frac{1}{T}$$

37

Ejemplo de tiempo de ejecución

inicio: setf LATE, 1 → Todo a uno

1 cycle cada uno

Mop
Mop
Mop

clrf LATE, 1 → todo a cero

Mop
Mop
Mop
Mop
Mop

2 cycles cada uno { para inicio

→ Total: 10 cycles x

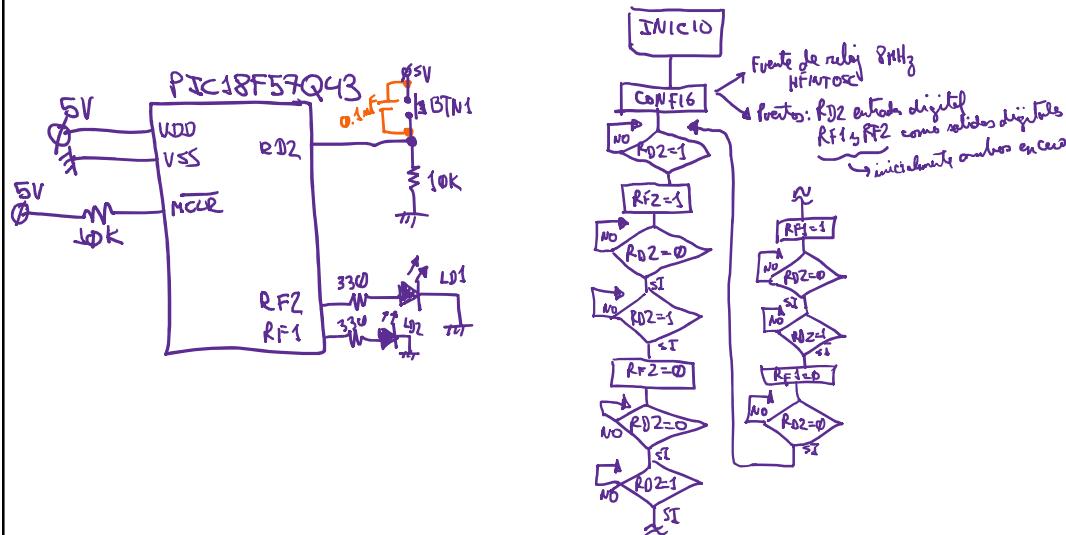
$\frac{1\text{us}}{10\text{us}} \rightarrow$ tiempo de ejecución del programa.

Si mi fuente de reloj = 4 MHz
⇒ cada instrucción simple se estará ejecutando en 1us.

38

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

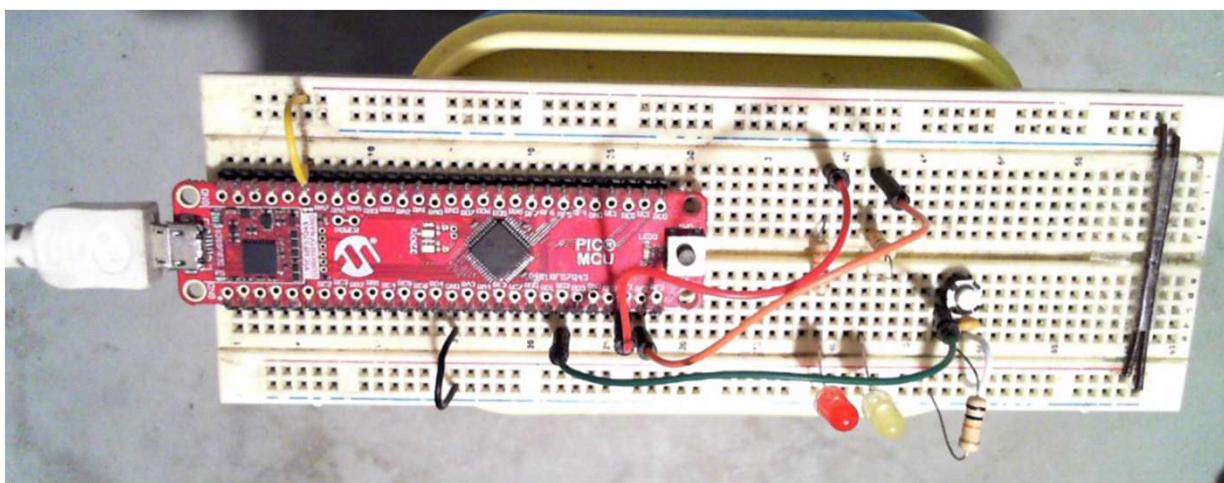
- Latch: Dispositivo enclavador – Mantiene el valor de salida hasta que pulses el botón



39

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

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



40

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

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

```

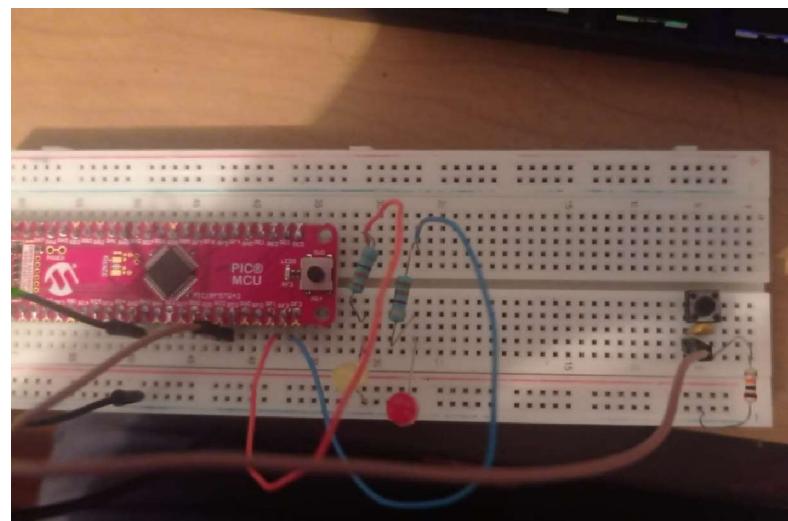
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6    ORG 000000H          ;vector de reset
7    bra configuro        ;salto a label configuro
8
9  configuro:             ;label configuro
10   movlb 0H              ;al BANK0
11   movlw 60H
12   movwf 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   bsf TRISD, 2, 1       ;RD2 como entrada
19   bcf ANSELB, 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
35 otroal:
36   btfss PORTD, 2, 1     ;pregunto si presione el boton
37   bra otroal             ;verdad, apago el LED
38
39 otro2:
40   btfsc PORTD, 2, 1     ;pregunto si solte el boton
41   bra otro2             ;falso, no presione el boton y vuelvo a preguntar
42
43 otro3:
44   btfsc PORTD, 2, 1     ;pregunto si deje de presionar el boton
45   bra otro3             ;falso y vuelvo a preguntar
46
47 otro4:
48   btfss PORTD, 2, 1     ;pregunto si pulse el boton
49   bra otro4             ;verdad, apago el LED
50
51 otro3:
52   btfsc PORTD, 2, 1     ;pregunto si deje de presionar el boton
53   bra otro3             ;falso y vuelvo a preguntar
54
55   bra inicio            ;regreso a label inicio
56
56 end upcino

```

41

Error en la implementación

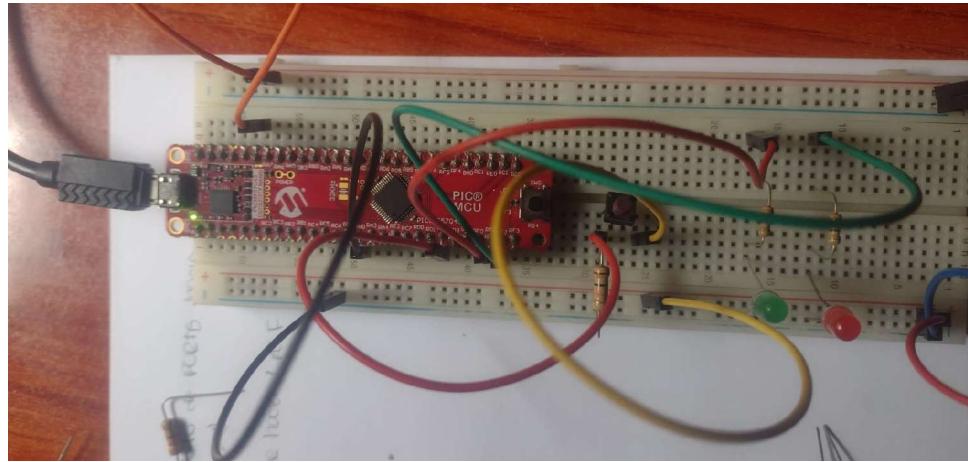
- “Profesor compila pero no funciona”
- Ubica el error:



42

Error en la implementación

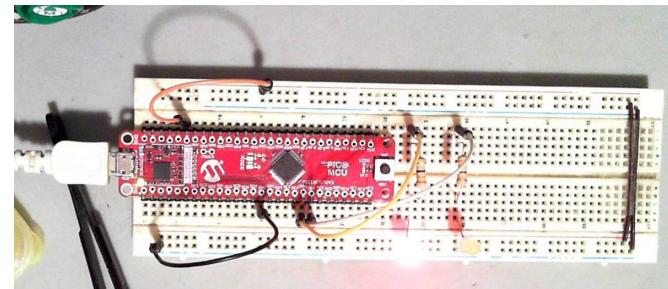
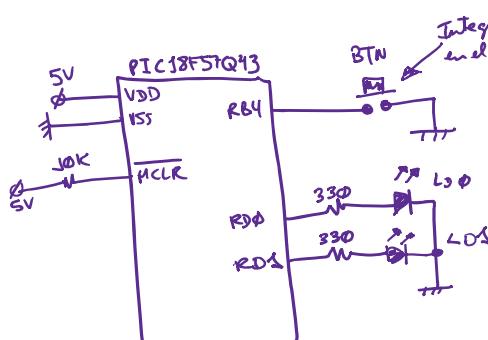
- “Profesor compila pero no funciona”
- Ubica el error:



43

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

- Diseño del hardware



44

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

- Código en XC8 PIC Assembler

```

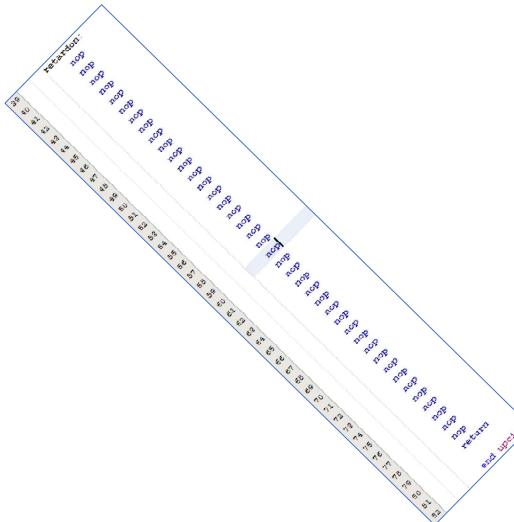
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6      ORG 000000H
7      bra configuro
8
9      ORG 000020H
10 configuro:
11     movlb 0H
12     movlw 60H
13     movwf OSCCON1, 1
14     movlw 02H
15     movwf OSCFRQ, 1
16     movlw 40H
17     movwf 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
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         bra inicio
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

```

45

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

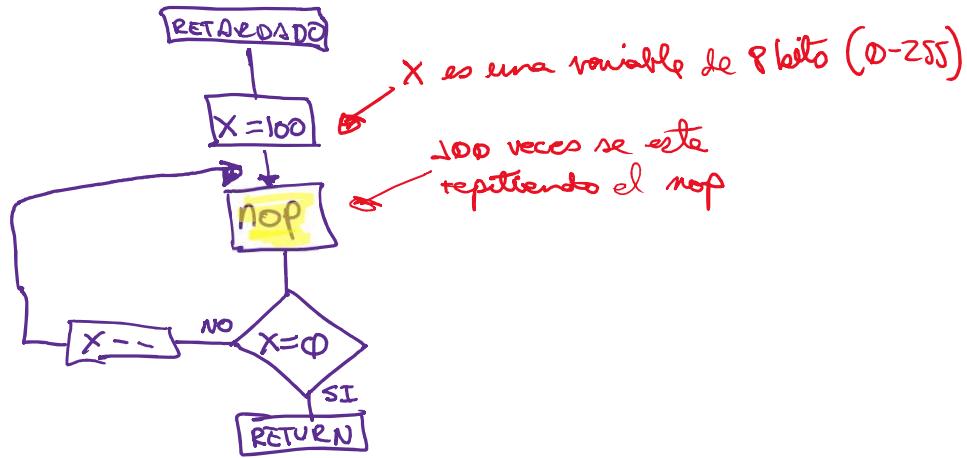
- Escribir 40000 nops no es práctico ni eficiente!



46

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

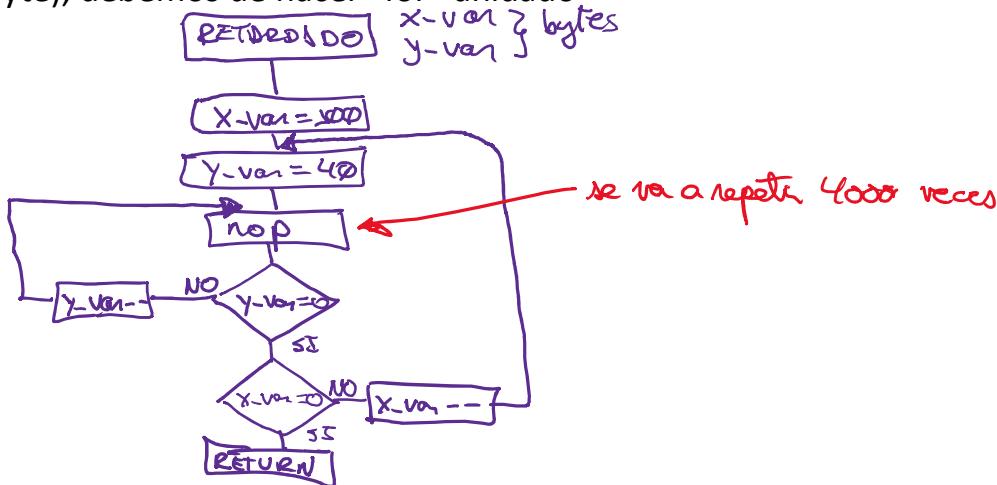
- Usar muchos nops ocupa demasiado espacio
- Se puede armar una rutina de bucle de repetición (for)



47

¿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



48

¿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

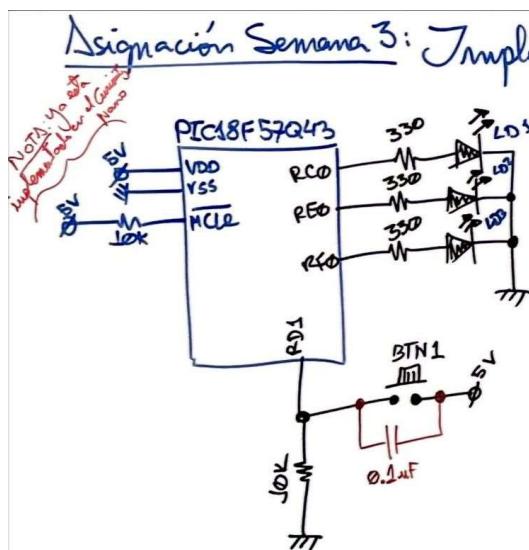
```

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

```

49

Ejemplo 2024-1



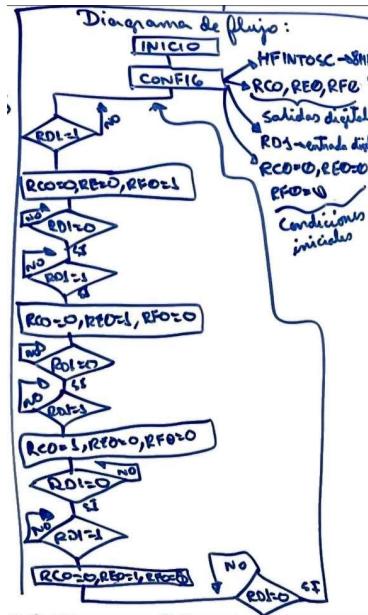
Funcionamiento: Al pulsar BTN1 se hará el cambio de visualización en los LED's según tabla siguiente:

	LD1	LD2	LD3
S1	OFF	OFF	ON
S2	OFF	ON	OFF
S3	ON	OFF	OFF
S4	OFF	ON	OFF

50

Ejemplo 2024-1

- Diagrama de flujo:



51

```

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:
6    ORG 000000H ;etiqueta upcino
7    bra configuro ;vector de reset
8
9    ORG 000100H ;salto a etiqueta configuro
10 configuro:
11   movlb 0H ;zona de programa de usuario
12   movlw 60H ;etiqueta configuro
13   movwf OSCCON1, 1 ;al BANK0
14   movlw 03H
15   movwf OSCFRC, 1 ;NOSC=110 (HFINTOSC), NDIV=0000 (1:1)
16   movlw 40H ;42
17   movwf OSCEN, 1 ;43
18   movlb 4H ;44
19   bcf TRISD, 1, 1 ;HFINTOSC a 8MHz
20   bcf ANSEL0, 1, 1 ;45
21   bcf TRISC, 0, 1 ;46
22   bcf ANSEL0, 0, 1 ;47
23   bcf TRISE, 0, 1 ;48
24   bcf ANSELE, 0, 1 ;49
25   bcf TRISE, 0, 1 ;50
26   bcf ANSELF, 0, 1 ;51
27   bcf LATC, 0, 1 ;52
28   bcf LATE, 0, 1 ;53
29   bcf LATF, 0, 1 ;54
30
31 inicio:
32   btfs PORTD, 1, 1 ;pregunto si pulse el BTN1
33   bra inicio ;no pulse BTN1 y me regreso
34   bcf LATC, 0, 1 ;60
35   bcf LATE, 0, 1 ;61
36   bsf LATF, 0, 1 ;62
37 otrol:
38   btfc PORTD, 1, 1 ;pregunto si deje de pulsar BTN1
39   bra otrol ;sigue BTN1 pulsado y regreso
40 otro2:
41   btfs PORTD, 1, 1 ;66
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

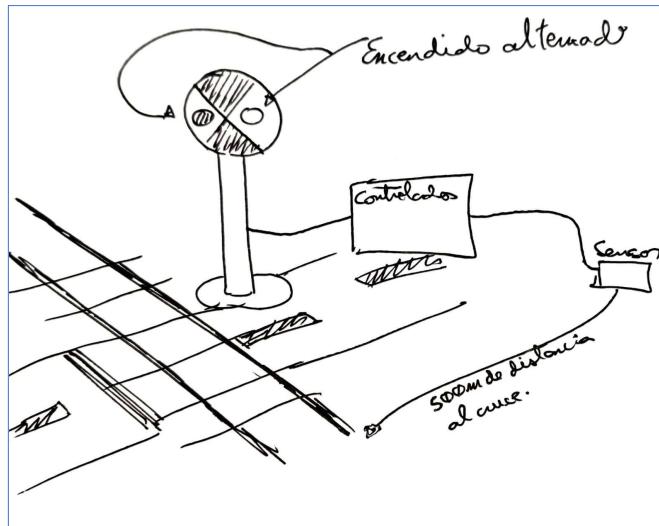
```

Ejemplo 2024-1

- Código:

Ejemplo 2 – 2024-2

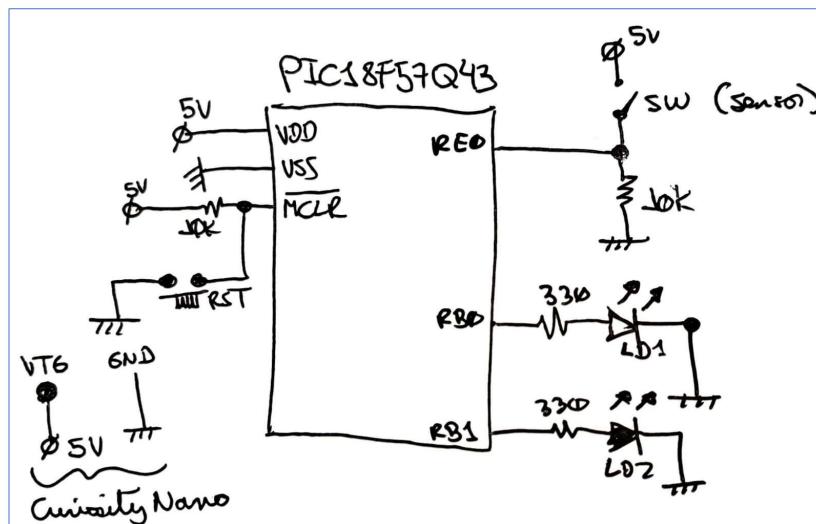
- Desarrollar una aplicación de una señal de cruce de tren con control de encendido a través de un pulsador.



53

Ejemplo 2 – 2024-2

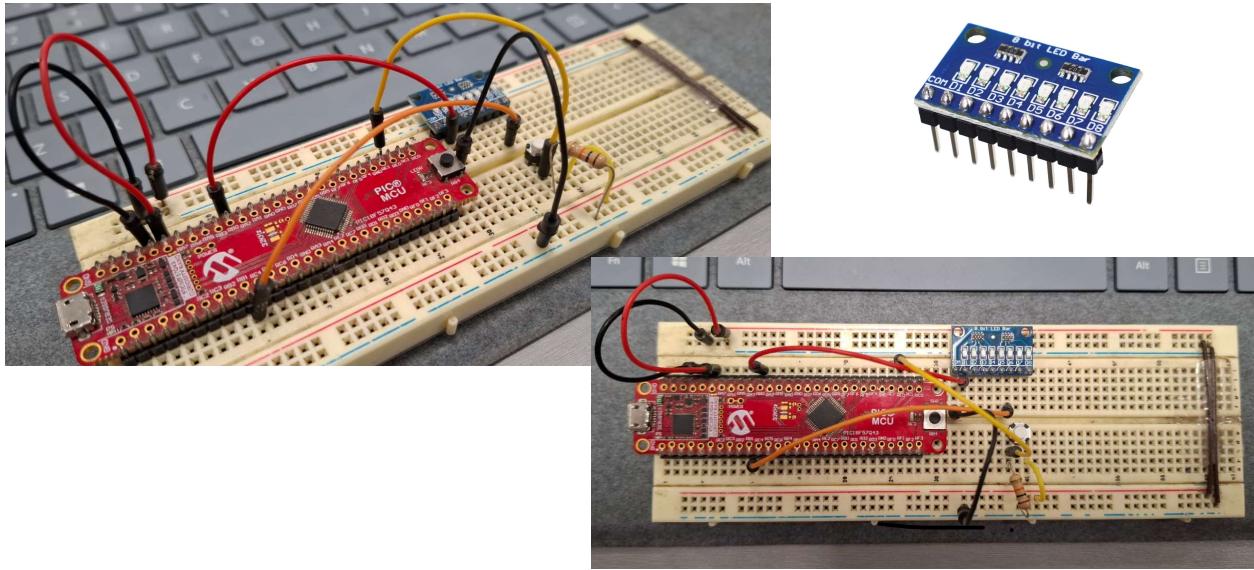
- Diseño del hardware – Diagrama esquemático



54

Ejemplo 2 – 2024-2

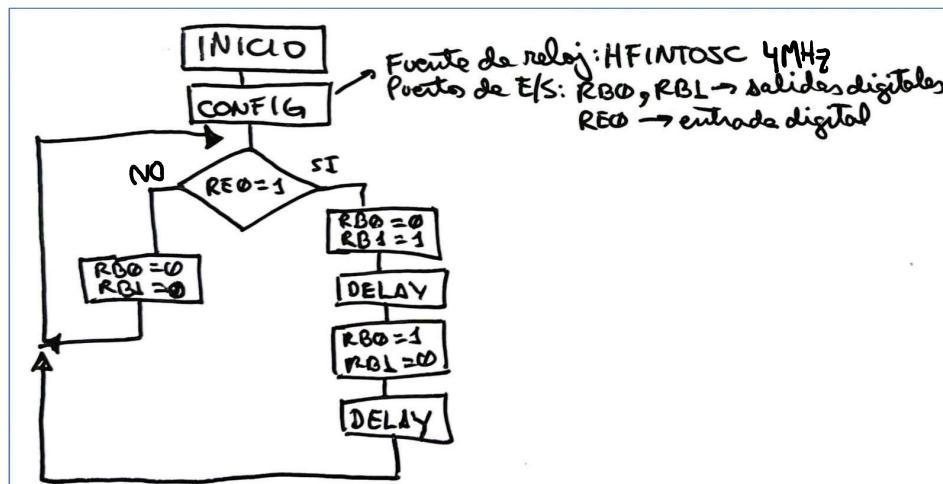
- Diseño del hardware – Implementación



55

Ejemplo 2 – 2024-2

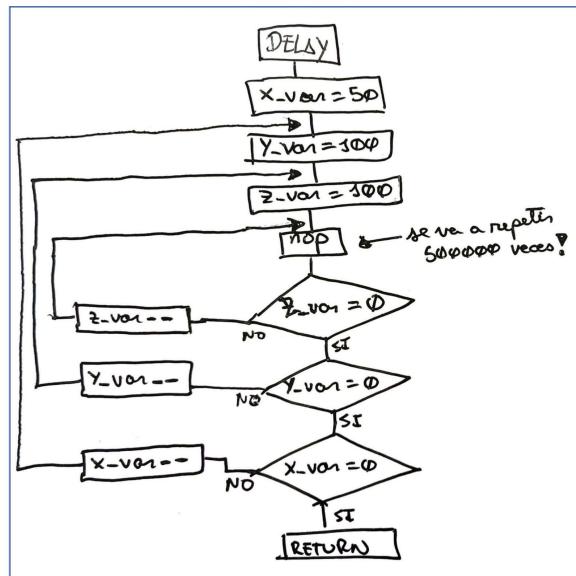
- Diseño de software: Diagrama de flujo



56

Ejemplo 2 – 2024-2

- Diseño de software: Diagrama de flujo



57

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 TRISE, 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                 cpfsq x_var, 1
66
67         falso_z:
68             movlw 0
69             cpfsq y_var, 1
70             bra falso_y
71             movlw 0
72             cpfsq x_var, 1
73             bra falso_x
74             movlb 4H               ;regresamos al Bank0
75             return
76
77         falso_x:
78             deef z_var, 1, 1
79             bra punto_0
80             falso_y:
81             deef y_var, 1, 1
82             bra punto_1
83             falso_z:
84             deef x_var, 1, 1
85             bra punto_2
86
87         end upcino.

```

58

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   es_verdad:
7  z_var EQU 502H           39     bsf LATB, 0, 1 ;enciendiendo RB0
8
9    PSECT upcino, class=CODE, reloc=2, abs 40     bcf LATB, 1, 1 ;apago RB1
10 upcino:                 41     call retardo ;llamada a subrutina retardo
11   ORG 000000H ;vector de RESET 42     bcf LATB, 0, 1 ;apago RB0
12   bra configuro ;salto a label configuro 43     bcf LATB, 1, 1 ;enciendiendo RB1
13
14   ORG 000100H ;zona de programa de usu 44     call retardo ;llamada a subrutina retardo
15 configuro:               45     bra inicio_tren ;retorno al inicio_tren
16   ;configuracion la fuente de reloj 46   es_falsoz:
17   movlb 0H ;nos vamos al Bank0 47     bcf LATB, 0, 1 ;apago RB0
18   movlw 60H                   48     bcf LATB, 1, 1 ;apago RB1
19   movwf OSCCON1, 1 ;NOSC=HFINTOSC, NDIV 49     bra inicio_tren ;retorno al inicio_tren
20   02H                         50   ;Subrutina de retardo
21   movwf OSCFRCQ, 1 ;HFINTOSC a 4MHz 51     retardo:
22   movlw 40H                   52       movlw 50
23   movwf OSCEN, 1 ;HFINTOSC habilitado 53       movwf x_var, 1 ;x_var valor
24   ;configuracion las E/S 54   punto_tres:
25   movlb 4H ;saltamos al Bank1 55       movlw 50
26   bcf TRISE, 0, 1 ;RE0 es entrada 56       movwf y_var, 1 ;y_var valor
27   bcf ANSELE, 0, 1 ;RE0 es digital 57   punto_dos:
28   bcf TRISB, 0, 1 ;RB0 es salida 58       movlw 50
29   bcf TRISB, 1, 1 ;RB1 es salida 59       movwf z_var, 1 ;z_var valor
30   ;Recordando bcf [registro], #bit, access 60   punto_uno:
31   bcf ANSELB, 0, 1 ;RB0 es digital 61       nop ;instruccion no operacion
32   bcf ANSELB, 1, 1 ;RB1 es digital 62       movwf z_var, 1, 1 ;evaluar z_var
33
34   inicio_tren:               63       btfs STATUS, 2, 1 ;pregunto si Z=1
35   btfs PORTE, 0, 1 ;pregunto si RE0 es 1 64       bra z_var_noescero ;Falso salta a label z_var_noescero
36   bra es_falsoz ;viene aqui cuando es Falso 65       movf y_var, 1, 1 ;Verdad, evaluar y_var
37   bra es_verdad ;viene aqui cuando es Verdadero 66       btfs STATUS, 2, 1 ;pregunta si Z=1
38   es_verdad:                67       bra y_var_noescero ;Falso salta a label y_var_noescero
39     bsf LATB, 0, 1 ;enciendiendo RB0 68       movf x_var, 1, 1 ;evaluo x_var
40     bcf LATB, 1, 1 ;apago RB1 69       btfss STATUS, 2, 1 ;preguntar si Z=1
41     call retardo ;llamada a subrutina retardo 70       bra x_var_noescero ;Falso salta label x_var_noescero
42     bcf LATB, 0, 1 ;apago RB0 71       return
43     bcf LATB, 1, 1 ;enciendiendo RB1 72       z_var_noescero:
44     call retardo ;llamada a subrutina retardo 73       decf z_var, 1, 1 ;decrementamos z_var
45     bra inicio_tren ;retorno al inicio_tren 74       bra punto_uno ;salto a label punto_uno
46   es_falsoz:                75       z_var_noescero:
47     bcf LATB, 0, 1 ;apago RB0 76       decf y_var, 1, 1 ;decrementamos y_var
48     bcf LATB, 1, 1 ;apago RB1 77       bra punto_dos ;salto a label punto_dos
49     bra inicio_tren ;retorno al inicio_tren 78       x_var_noescero:
50   ;Subrutina de retardo 79       decf x_var, 1, 1 ;decremente x_var
51   retardo:                  80       bra punto_tres
52     movlw 50 81       end_upcino
53     movwf x_var, 1 ;x_var valor
54   punto_tres:               82
55     movlw 50
56     movwf y_var, 1 ;y_var valor
57   punto_dos:                83
58     movlw 50
59     movwf z_var, 1 ;z_var valor
60   punto_uno:                84
61     nop ;instruccion no operacion
62     movwf z_var, 1, 1 ;evaluar z_var
63     btfs STATUS, 2, 1 ;pregunto si Z=1
64     bra z_var_noescero ;Falso salta a label z_var_noescero
65     movf y_var, 1, 1 ;Verdad, evaluar y_var
66     btfs STATUS, 2, 1 ;pregunta si Z=1

```

59

Ejemplo 2 – 2024-2

- Diseño de software: Codificación en lenguaje XC8 PIC Assembler

```

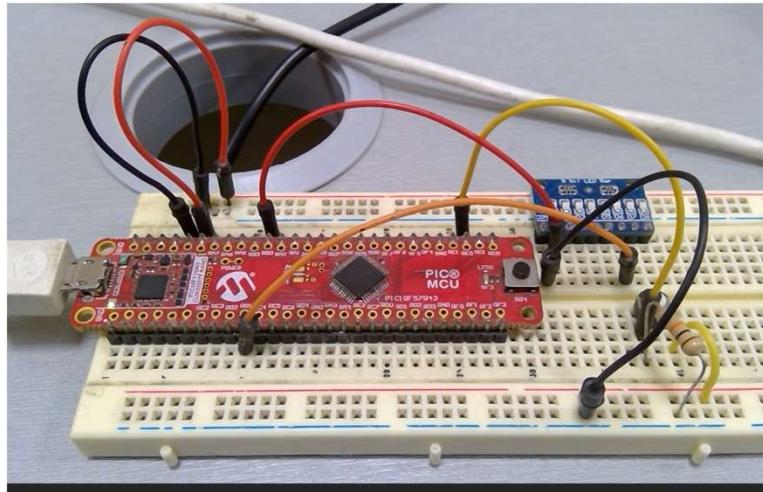
1      PROCESSOR 18F57Q43
2      #include "cabecera.inc"
3
4      x_var EQU 500H
5      y_var EQU 501H
6      z_var EQU 502H
7
8      PSECT upcino, class=CODE, reloc=2, abs
9      upcino:
10     ORG 000000H      ;vector de reset
11     bra configuro
12
13     ORG 000100H      ;zona de usuario
14 configuro:
15     ;configuracion de la fuente de reloj
16     movlb 0H           ;al Bank0
17     movlw 60H
18     movwf OSCCON1, 1   ;NOSC=HFINTOSC NDIV
19     movlw 02H
20     movwf OSCFRC, 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 0xFC          ;en binario 1111110
26     movwf TRISE, 1      ;RB0 y RB1 como sal
27     movwf ANSELB, 1      ;RB0 y RB1 como dig
28     bcf TRISE, 0        ;REO como entrada
29
30
31 inicio_tren:
32     btffs PORTE, 0, 1  ;pregunto
33     bra es_falso       ;viene a
34     bra es_verdad      ;viene a
35 es_falso:
36     clrf LATB, 1        ;apagamo
37     bra inicio_tren
38 es_verdad:
39     movlw 01H
40     movwf LATB, 1        ;RBL=0,
41     call retardado      ;llamada
42     movlw 02H
43     movwf LATB, 1        ;RBL=1,
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 nop
60 movf z_var, 1, 1
61 movlb 4H               ;Al Bank4
62 btffs STATUS, 2, 1     ;pregunto si z_var llego a c
63 bra zvar_noescero      ;falso zvar no llego a cero
64 movlb 5H               ;Al bank5
65 movf y_var, 1, 1        ;verdad y prosigo a ls sigui
66 movlb 4H               ;Al Bank4
67 btffs STATUS, 2, 1     ;pregunto si y_var llego a c
68 bra yvar_noescero      ;falso yvar no llego a cero
69 movlb 5H               ;Al bank5
70 movf x_var, 1, 1
71 movlb 4H               ;Al Bank4
72 btffs STATUS, 2, 1     ;pregunto si x_var llego a c
73 bra xvar_noescero      ;falso xvar no llego a cero
74 return
75 zvar_noescero:
76     movlb 5H             ;Al bank5
77     decf z_var, 1, 1     ;decremento z_var
78     bra punto_uno
79 yvar_noescero:
80     movlb 5H             ;Al bank5
81     decf y_var, 1, 1     ;decremento y_var
82     bra punto_dos
83 xvar_noescero:
84     movlb 5H             ;Al bank5
85     decf x_var, 1, 1     ;decremtno x_var
86     bra punto_tres
87
88 end upcino

```

60

Ejemplo 2 – 2024-2

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



61

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

62

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.

63

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

64

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.

65

Fin de la sesión

66

33