

Microcontroladores

Semana 4

Semestre 2024-2

Por Kalun José Lau Gan

1

Preguntas previas

- En el ejercicio de generación de retardo de 40ms de la semana 3. ¿Cómo se declaran las etiquetas de los GPR?
 - En la parte superior del programa debajo de la declaración del PSECT se coloca:


```
x_var EQU 500H
y_var EQU 501H
```
- Hay una nueva versión del XC8: **v2.50 (actualizado al 02/09/2024)**
- ¿Cómo puedo saber en que banco se encuentra un registro?
 - Revisando en la hoja técnica, generalmente los registros de manipulación de E/S se encuentran en el BANK4, los registros que manipulan la configuración del módulo de oscilador en el BANK0, la memoria RAM esta mapeada a partir del BANK5 y así en sucesivo.
 - **En el detalle de algún registro, se visualiza la dirección por ejemplo 0x04C7, debido a que el BSR ocupa los bits del 8 al 13, entonces el bank de dicho registro será 04.**

04c7 H
BSR

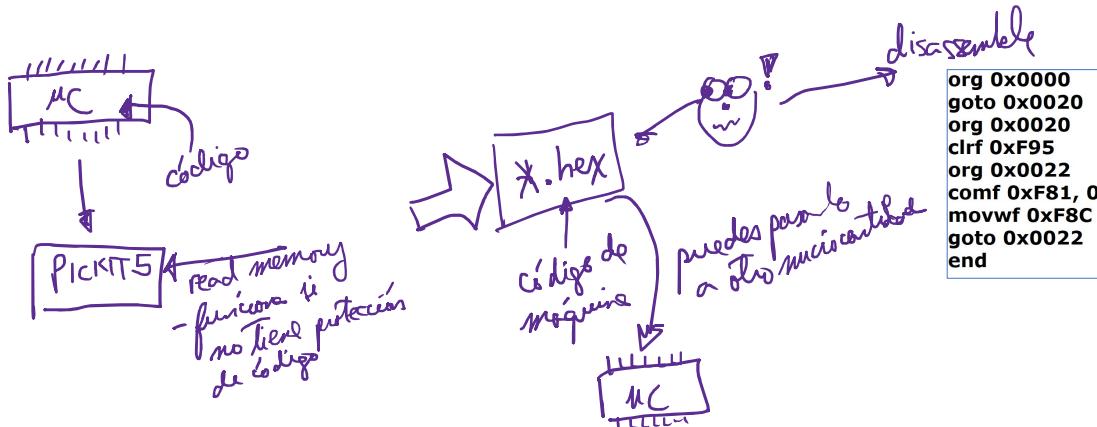
MOVLB	k	Move literal to BSR<5:0>
-------	---	--------------------------

0x04C7	TRISB	7:0
--------	-------	-----

2

Preguntas previas 2024:

- Profe, quiero sacar el programa de un micro para ver que hace y luego pasarlo a otro micro pero no se entiende nada.



3

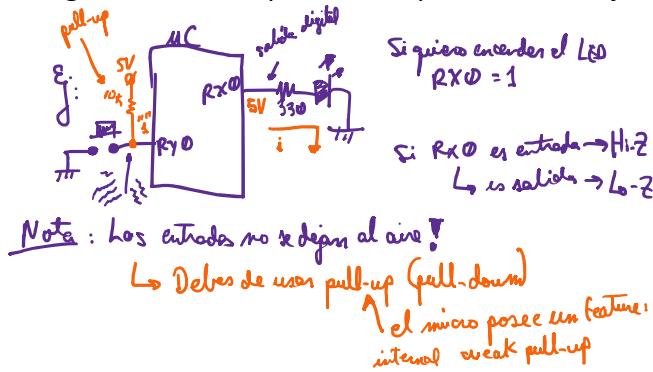
Preguntas previas 2024:

- Profe, he querido hacer los circuitos que hemos hecho en clase durante el fin de semana y no funciona, que puede ser?
 - Hay tres etapas importantes:
 - Durante el desarrollo del código en el MPLABX, si el código no compila revisar sintaxis
 - Cuando se está en el evento de programar la memoria del μC no gte sale el mensaje "Programming Successful", problema del hardware, posiblemente falla en la conexión entre la PC y el Curiosity Nano, o algún evento de corto circuito que no permite energizar y hacer la programación
 - Cuando se programó satisfactoriamente el μC y no funciona la aplicación, problemas en la implementación del hardware, revisar polaridad de los LEDs, valor de las resistencias, continuidad de los cables jumper, conexiones de alimentación, protoboard en mal estado, componentes quemados, etc.

4

Preguntas previas 2024:

- Profe, cuando pruebo los ejercicios no funciona la implementación, acerco mi mano y si funciona.
 - Tienes manos “mágicas” ... XD
 - El cuerpo humano tiene una capacitancia, esto permite que puedas almacenar carga eléctrica y que eso se ve reflejado cuando acerca la mano al circuito.
 - Revisar configuración de los puertos empleados en el ejemplo.



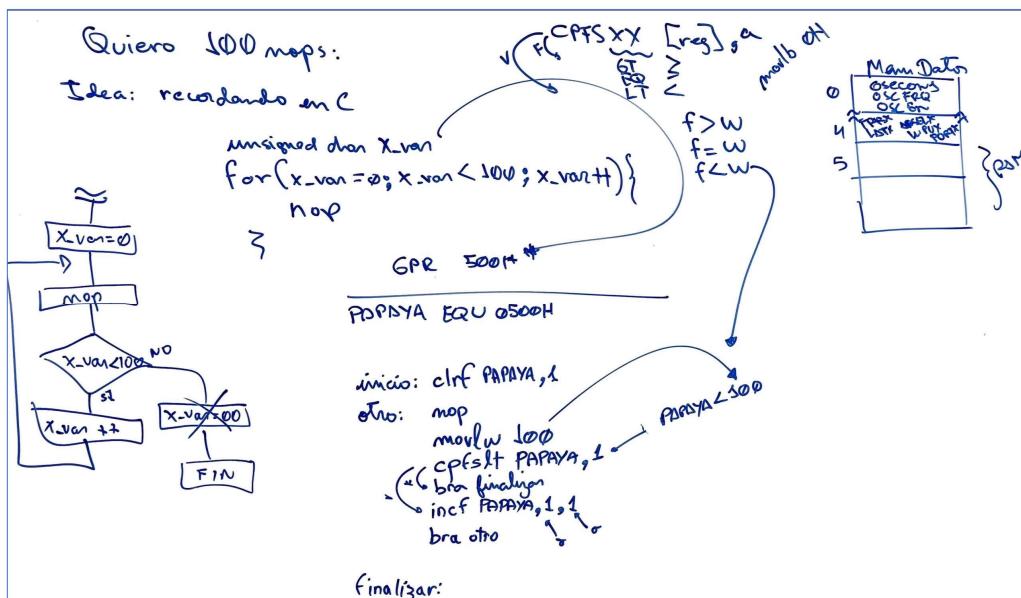
NOTA:

Recordar que los registros para manipular los E/S se encuentran en el BMWK4 (revisar Cap 19 del libro)

WPUB. 4 = 1

5

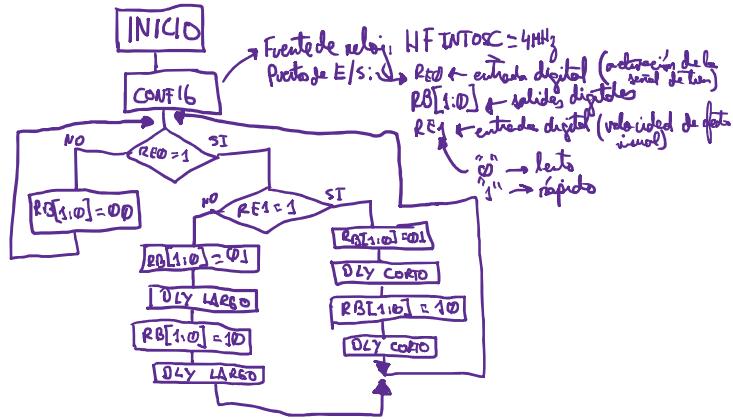
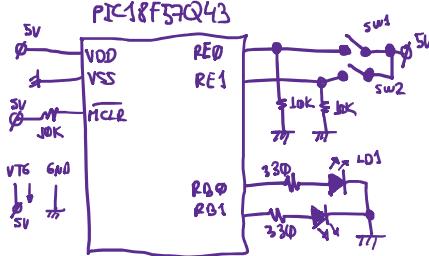
Preguntas previas 2024:



6

Preguntas previas 2024-2

- ¿Cómo sería la estructura funcional (diagrama de flujo) de la asignación que se dejó planteado en la semana 3?



7

No cometer estas acciones durante las evaluaciones

Everyone

```

Diego Luis Andres Estrella R... 404 PM
PROCESSOR 18F57Q43
#include "cabezera.inc"

PSECT upoino, class=CODE, reloc=2,
abs:
upoino:
ORG 000000H
bra configuro

ORG 001000H
table_7xDB3FH, 0DH, 0BH, 4FH, 6FH,
6DH, 7DH, 0TH, 7FH, 6TH, 40H, 63H,
5CH, 7CH, 5BH, 79H

ORG 000100H
configuro:
movlw 0H
movlw 0H
movwf OSCCON1, 1
movwf 03H
movwf OSCFRC1, 1
movlw 40H
movwf OSCBN, 1
movlw 4H
movwf 0H

movwf TRISB, 1
movwf ANSELB, 1
movlw 0FH
movwf ANSELD, 1
movwf WPUD, 1

movlw 0FH
clrf TBLPTRU, 1
movlw 10H
movwf TBLPTRH, 1
clrf TBLPTRL, 1

```

Say something

B I U G Send

*C byorm_soleado 16:31 10/04/2024

Everyone

```

clrf TBLPTRL, 1
inicio:
    clrf PORTD, 0, 1
    andlw 0FH
    addwf TBLPTRL, 1
    TBLRD
    movwf TABLAT,LATB
    bra inicio

INTO_ISR:
    bcf LATD,0,1
    bcf PIR1,0,1
    retfie
end upoino

Kalun José Lau Gan 405 PM
Diego, tu grupo va a recibir una fuerte penalización por haber colocado en publico el código

Diego Luis Andres Estrella R... 405 PM
pero no se si esta bien

Yuzetti Yadis Torres Altamirano... 405 PM
prof se le escapó a mi compañero

Kalun José Lau Gan 405 PM
estas transgrediendo el reglamento de estudiantes con posible reporte por plagio

```

PROBLEMAS CON EL LAB 1 GRUPO 6

u202013651 (Villanueva Gonzales, Sebastian Alonso) ... | ...
To: poeklau (Lau Gan, Kalun Jose); Kalun José Lau Gan Wed 4/10/2024 21:17

Estimado Profesor,

Habiendo acabado el laboratorio 1, mi grupo ha notado que accidentalmente subimos el archivo al grupo 6 pero de otra sección. Como puede visualizar en esta imagen (y en la carpeta compartida) el archivo fue subido a tiempo. Por la presión del tiempo finalmente erramos en la carpeta. Espero sea comprender nuestro error para la calificación. Los integrantes de mi grupo son Villanueva Gonzales, Sebastian Alonso - u202013651, Paredes Eche, Alessandro - u202112055, Espichan Quispe, Alci Mateo - u202111525, Hernandez De La Cruz, Diego Sebastian - u202123436 de la sección EL51 en el grupo 6.

Mis mejores deseos,

Villanueva Gonzales, Sebastian Alonso

8

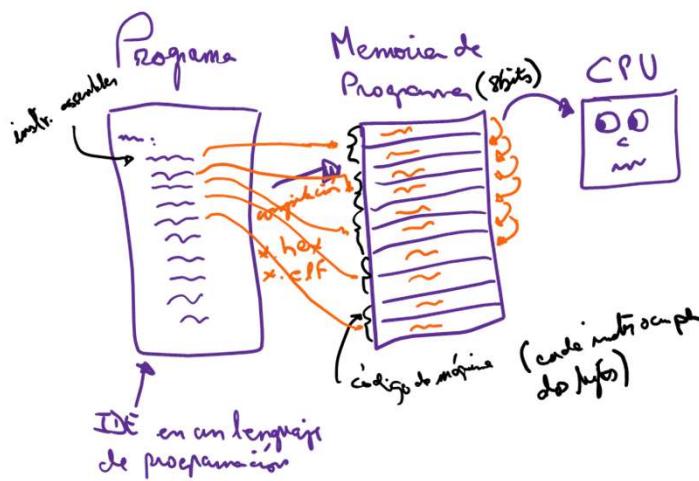
Agenda

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

9

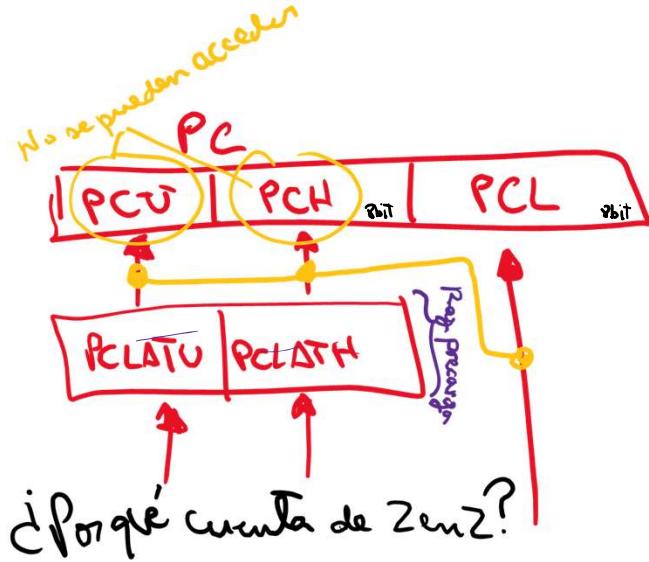
Recordando:

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



10

El Contador de Programa (PC)



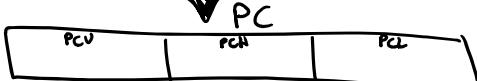
- Indispensable para la ejecución secuencial de las instrucciones.
- Su función es de alojar la dirección de la siguiente instrucción que el CPU va a ejecutar.
- Según hoja técnica, consta de 21 bits separados en tres registros: PCU, PCH y PCL.
- PCU y PCH no son accesibles, para que se pueda escribir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

11

Ejemplo

- Cargar dirección 00288AH en PC:

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



Nota: El banco extra debe de ser el 4

9.8 Register Summary - Memory Organization

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x04D9	FSR2	7:0								FSRL[7:0]
		15:8								FSRH[5:0]
0x04DB	PLUSW2	7:0								PLUSW[7:0]
0x04DC	PREINC2	7:0								PREINC[7:0]
0x04DD	POSTDEC2	7:0								POSTDEC[7:0]
0x04DE	POSTINC2	7:0								POSTINC[7:0]
0x04DF	INDF2	7:0								INDF[7:0]
0x04E0	BSR	7:0								BSR[5:0]
0x04E1	FSR1	7:0								FSRL[7:0]
		15:8								FSRH[5:0]
0x04E3	PLUSW1	7:0								PLUSW[7:0]
0x04E4	PREINC1	7:0								PREINC[7:0]
0x04E5	POSTDEC1	7:0								POSTDEC[7:0]
0x04E6	POSTINC1	7:0								POSTINC[7:0]
0x04E7	INDF1	7:0								INDF[7:0]
0x04E8	WREG	7:0								WREG[7:0]
0x04E9	FSR0	7:0								FSRL[7:0]
		15:8								FSRH[5:0]
0x04EB	PLUSW0	7:0								PLUSW[7:0]
0x04EC	PREINC0	7:0								PREINC[7:0]
0x04ED	POSTDEC0	7:0								POSTDEC[7:0]
0x04EE	POSTINC0	7:0								POSTINC[7:0]
0x04EF	INDF0	7:0								INDF[7:0]
0x04F0	...	Reserved								
0x04F8	PCL	7:0								PCL[7:0]
0x04F9	PCLAT	15:8								PCLATH[7:0]
0x04FA	STKPTR	7:0								STKPTR[8:0]
0x04FC	TOS	7:0								TOS[7:0]
0x04FD	TOS	15:8	23:16							TOS[15:8] TOS[20:16]

12

Ejemplo

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

```

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

```

↓
Direcciones de la memoria de programme

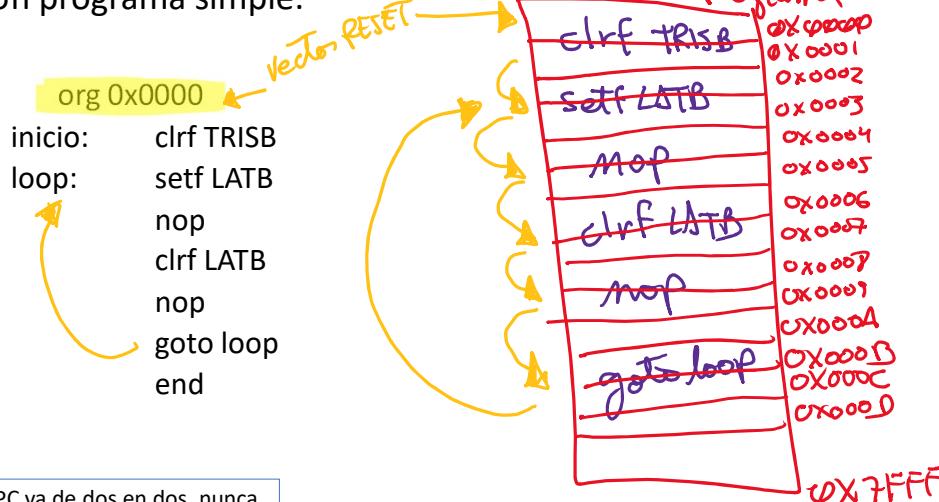
0x0000
 0x0020
 0x0022
 0x0024
 0x0026

hace un salto a 0x0020
 haz un salto a 0x0022
 modifica el PC

13

¿Cómo funciona el PC?

- Un programa simple:



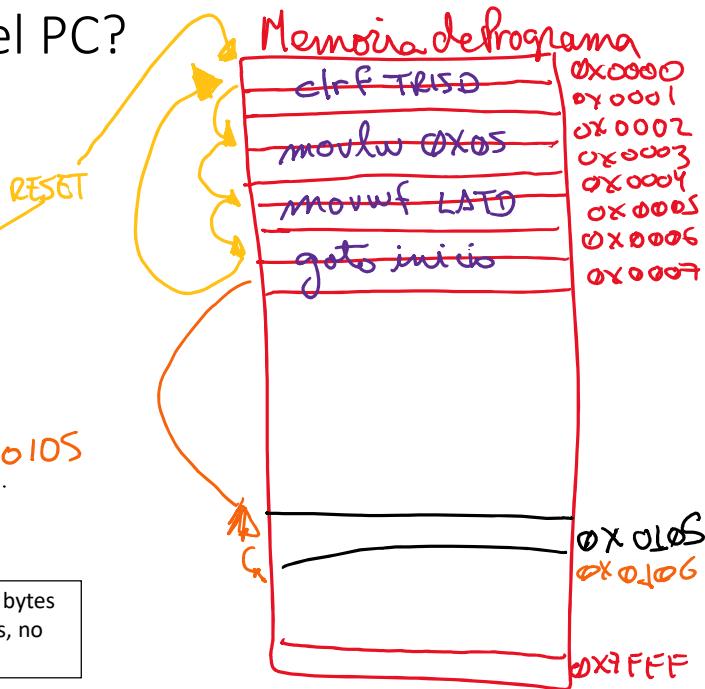
14

¿Cómo funciona el PC?

- Se tiene el siguiente programa

```
org 0x0000
inicio: clrf TRISD
          movlw 0x05
          movwf LATD
          goto inicio
end
```

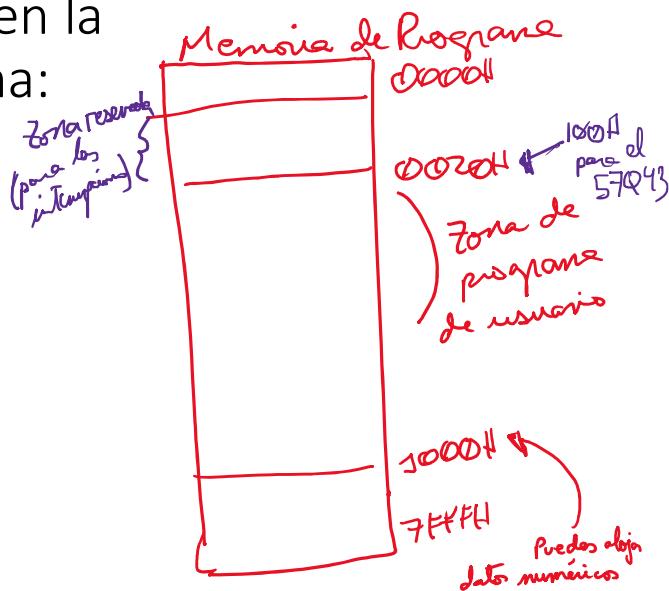
Nota: Las instrucciones en MPASM ocupan 2 bytes
 Nota: Se comprueba que PC va de dos en dos, no puede contener una dirección impar



15

Información alojada en la memoria de programa:

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



16

Otro ejemplo de PC:

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

bra 86H

```

ORG 000000H           ;00H
bra 000080H

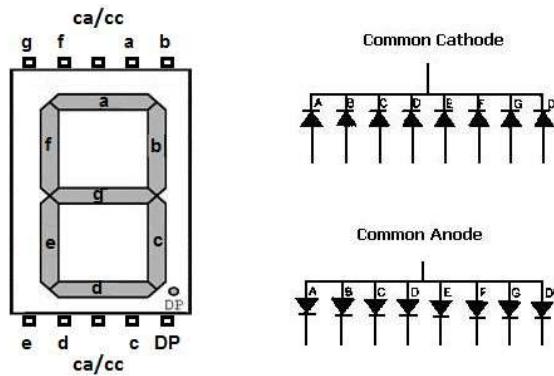
ORG 000080H           ;80H
movlb 4H
bcf TRISA, 0, 1        ;82H
bcf ANSELA, 0, 1        ;84H
btg LATA, 0, 1          ;86H
nop
clrf PCLATU
clrf PCLATH
movlw 86H
movwf PCL               ; hace un salto a 86H

```

17

El display de siete segmentos

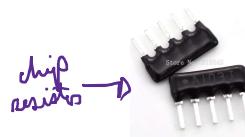
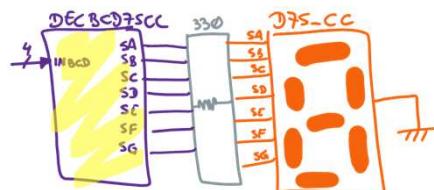
- Dos tipos: ánodo común y cátodo común
- Tablas de decodificación diferentes entre ellos.
- Es necesario colocarle resistencias limitadoras de corriente para cada segmento que compone el display (100ohm a 1kohm).
- Evitar usar solo una resistencia en el pin común del display ya que al cambiar de carácter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



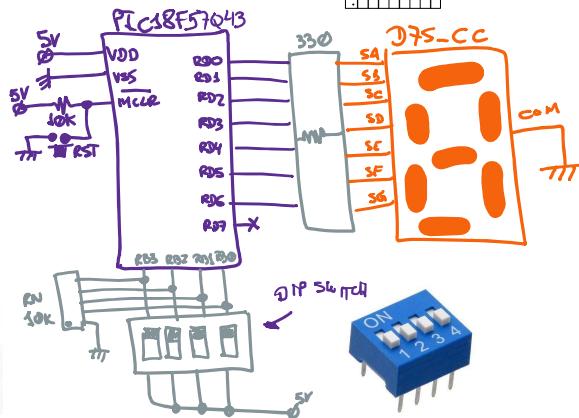
18

Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:



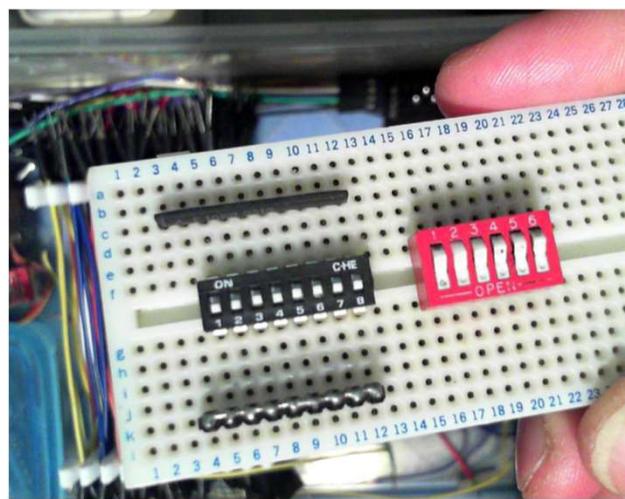
Hardware propuesto:



19

Uso de DIP Switches y Chip Resistors

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



20

Ejemplo 2024-1 Sem4

```

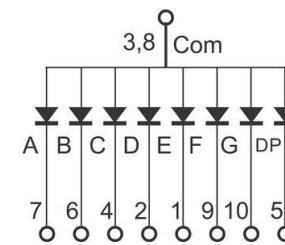
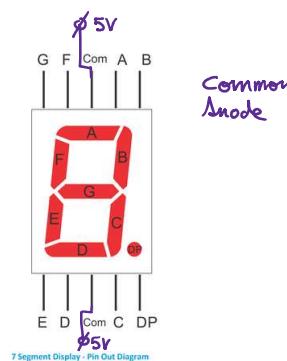
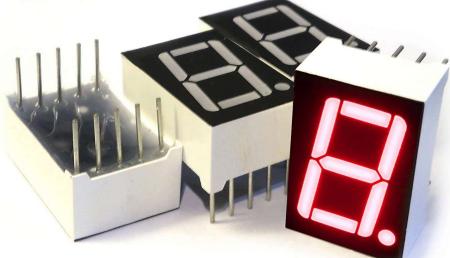
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6    ORG 000000H
7    bra configuro
8
9    ORG 001000H
10   tabla_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12   ORG 000100H
13   configuro:
14     movlb 0H
15     movlw 60H
16     movwf OSCCON1, 1
17     movlw 03H
18     movwf OSCFRC, 1
19     movlw 40H
20     movwf OSCEN, 1
21     movlb 4H
22     movlw 0FFH
23     movwf TRISB, 1
24     movlw 0F0H
25     movwf ANSELB, 1
26     movlw 0FH
27     movwf WPUB, 1
28     movlw 80H
29     movwf TRISD, 1
30     movwf ANSELD, 1
31     movlw 00H
32     movwf TBLPTRU, 1
33     movlw 10H
34     movwf TBLPTRH, 1
35     movlw 00H
36     movwf TBLPTRL, 1
37
38 inicio:
39     comf PORTB, 0, 1      ;complemento PORTB y lo mando a Wreg
40     andlw 0FH              ;enmascaramiento para que pase solo los 4 menos sign
41     movwf TBLPTRL, 1        ;cambio de dirección de apunte de TBLPTR segun entra
42     TBLRD*                 ;acción de lectura del TBLPTR
43     movff TABLAT, LATD     ;mando el contenido leido al puerto del display
44     bra inicio
45
46 end upcino

```

21

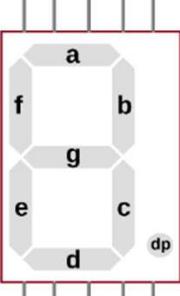
Observación:

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



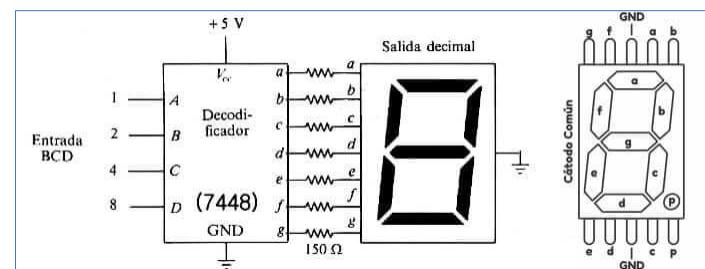
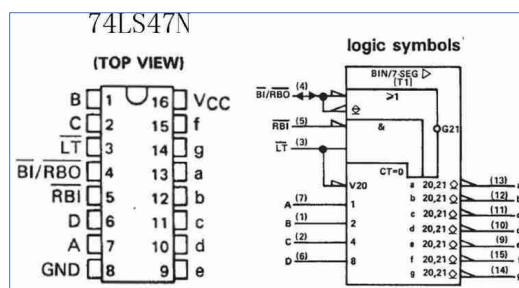
22

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

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

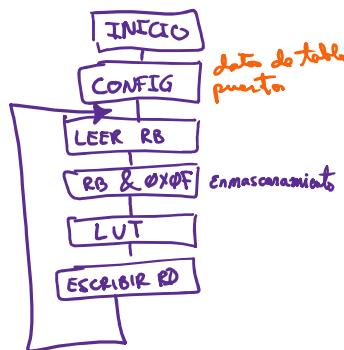
23

Circuitos TTL decodificadores de siete segmentos

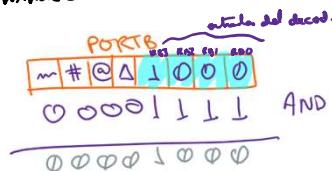


24

Diagrama de flujo



Enmascaramiento



Código previo (empleando PC como LUT)

```

31      movf PORTB, w
32      andlw 0x0F
33      movwf valor_entrada
34      call tabla_pc
35      movwf LATD
36      goto loop
37
38
39
40      movf valor_entrada, w
41      addwf PCL, f
42      (0)retlw 0x3F
43      (1)retlw 0x06
44      (2)retlw 0x5B
45      (3)retlw 0x4F
46      retlw 0x66
47      retlw 0x6D
48      retlw 0x7D
49      retlw 0x07
50      retlw 0x7F
51      retlw 0x67
52
53      end
      
```

Diagrama de memoria programada (Mem_Prog):

Movf val... ,W	x+0
addwf PCL,f	x+1
retlw 0X3F	x+2
retlw 0X06	x+3
retlw 0X5B	x+4
retlw 0X4F	x+5
retlw 0X66	x+6
retlw 0X6D	x+7
retlw 0X7D	x+8
retlw 0X07	x+9
retlw 0X7F	:
retlw 0X67	

Problema: Debido a que se esta empleando el PC como LUT, al interactuar los datos de entrada (PORTB) con PC se obtendrán saltos a direcciones **impares!**

25

Propuesta de arreglo de problema

```

39      tabla_pc:
40          movf valor_entrada, w
41          addwf valor_entrada, f
42          movf valor_entrada, w
43          addwf PCL, f
      
```

Diagrama de flujo propuesto:

- Entrada: PORTB (3)
- ↓
- enmascaramiento
- ↓
- valor_entrada (3)
- ↓
- W ← 3
- ↓
- W + valor_entrada
- 3 + 3 → valor_entrada
- ↓
- W ← 6

Nota: Para que se realicen los saltos correctos empleando el PC (como LUT) se propone que luego de obtener el valor del dato de entrada de PORTB, éste dato se sumará a si mismo de tal modo que sea el doble, se obtenga las direcciones de salto en número par para el PC y funcione correctamente la LUT

26

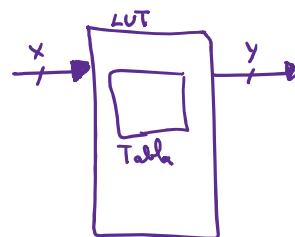
Acerca de la memoria de programa

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

27

Tablas de búsqueda (lookup tables - LUT)

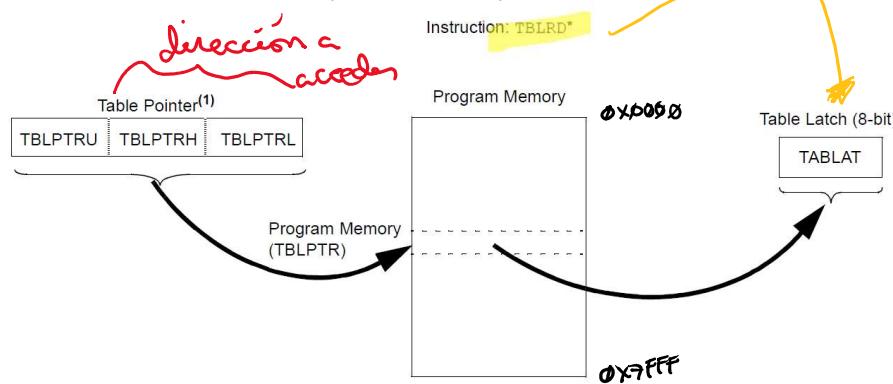
- Decodificadores implementados en software



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

28

El puntero de tabla (TBLPTR)

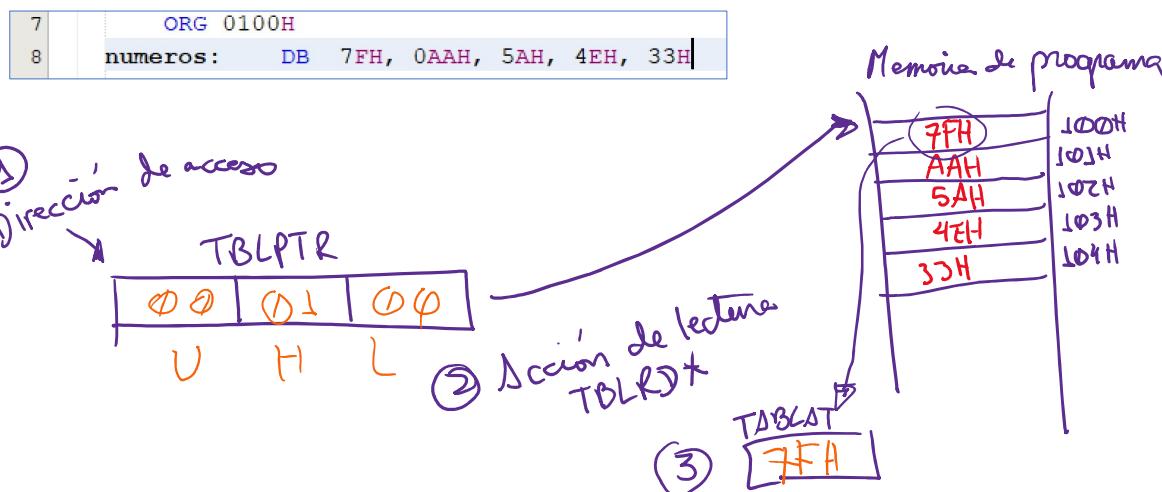


Note 1: Table Pointer register points to a byte in program memory.

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

29

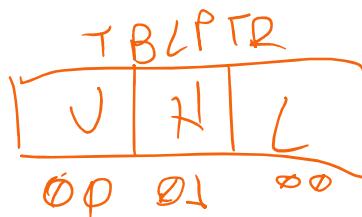
Operación con el TBLPTR



30

Operación con el TBLPTR

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



```

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

```

31

Modificación del ejemplo del decodificador anterior empleando TBLPTR

```

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

```

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

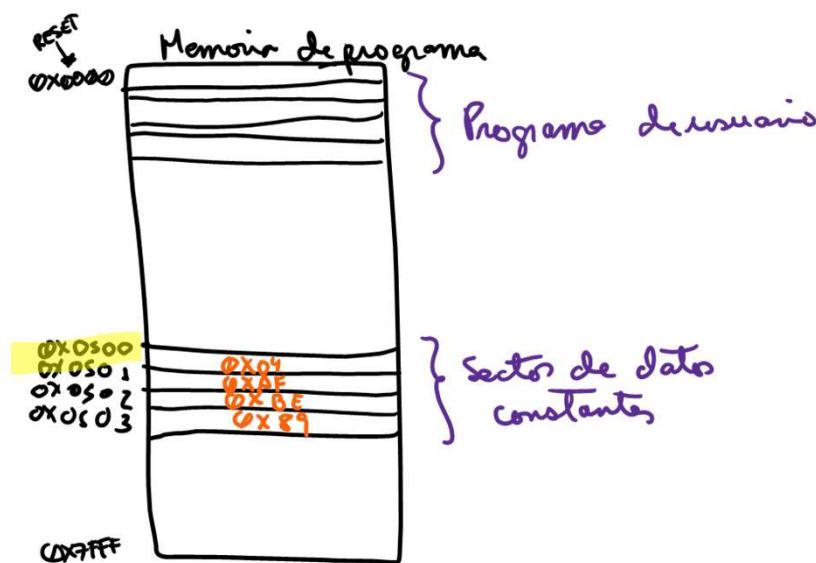
32

Ejemplo:

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

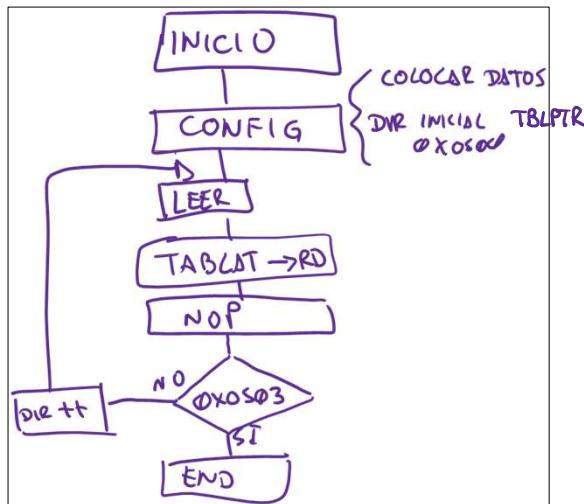
33

Desarrollo del ejemplo:



34

Diagrama de flujo:



35

Resumen de instrucciones con toma de decisión:

Formato:

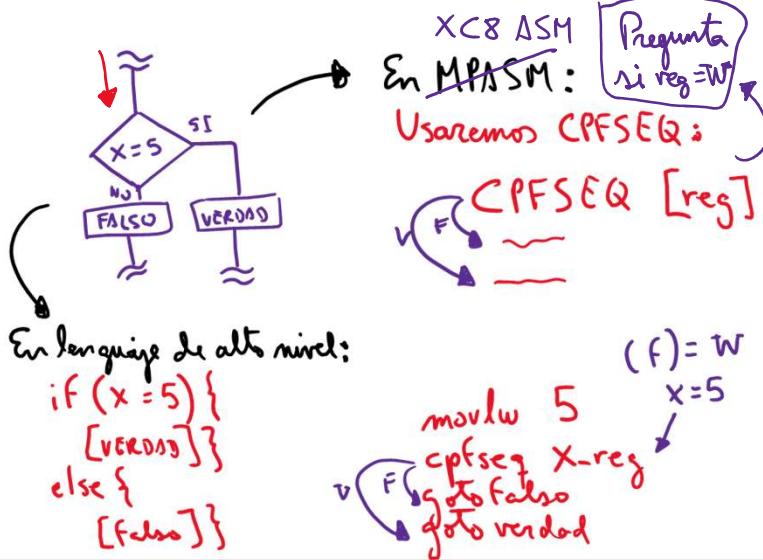
- btfss, btfsc
 - Prueba el #bit de [registro] si es 0 ó 1
 - decfsz, incfsz
 - Decrementa o incrementa [registro] y pregunta si es cero.
 - dcfnsz, icfnsz
 - Decrementa o incrementa [registro] y pregunta si **no** es cero
 - cpfsgt, cpfseq, cpfslt
 - Comparaciones numéricas entre [registro] y W
 - tstfsz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

36

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

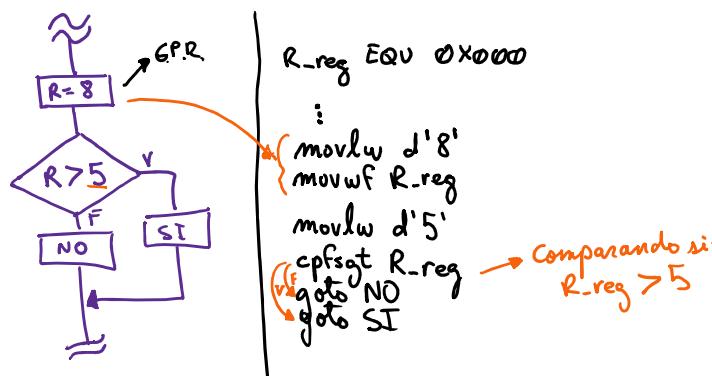
$CPFSEQ \rightarrow f = W_{reg}$
 $CPFS LT \rightarrow f < W_{reg}$
 $CPFS GT \rightarrow f > W_{reg}$

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



37

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

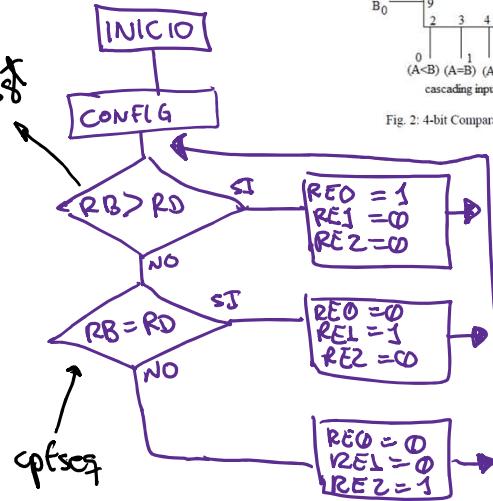
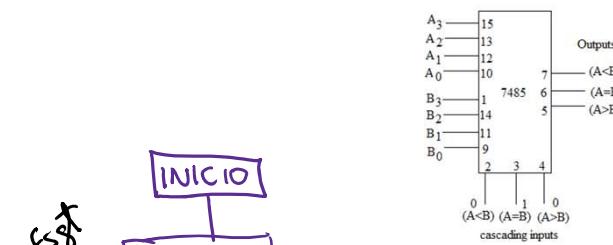
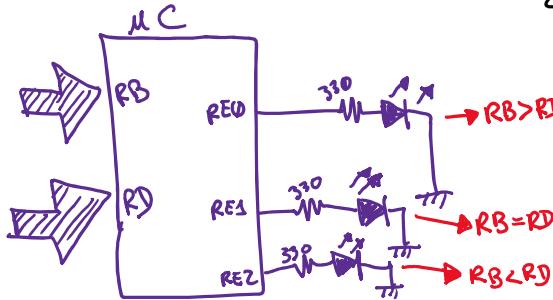


Nota: En las instrucciones cpfs-- el segundo parámetro de la comparación debe de estar en Wreg

38

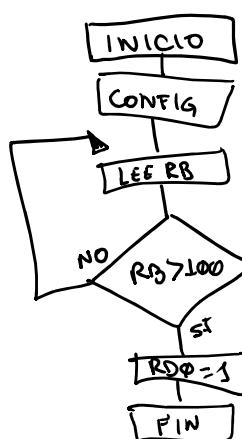
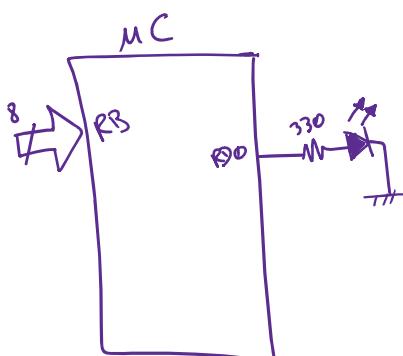
Ejercicio:

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



39

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



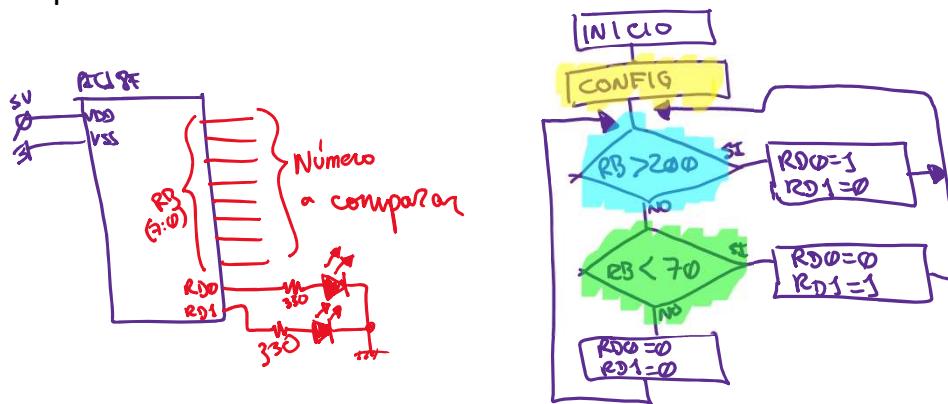
$f > w_{reg}$

bucle:
`movlw .100`
`cpfsat PORTB`
`goto bucle`
`bsf LATD, 0`

40

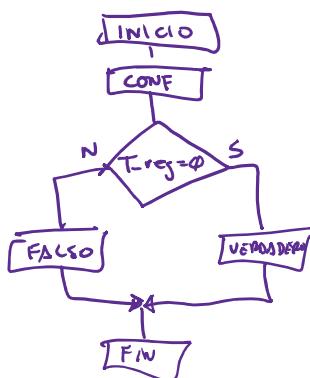
Ejercicio:

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



41

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



Opción 1:

inicio: movlw .0
cpfseq T-reg
v → goto FALSO
v → goto VERDADERO

Opción 2:

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

Opción 3:

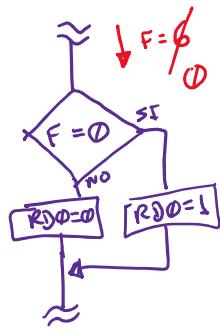
inicio: movf T-reg, W
sublw .0
btfs STATUS, Z
v → goto FALSO
v → goto VERDADERO

Opción 4:

inicio: movf T-reg, W
sublw .0
b7 VERDADERO
FALSO: ≡
VERDADERO: ≡

42

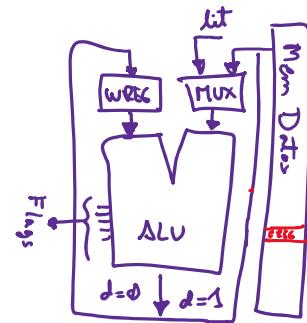
Aplicación de sublw para tomas de decisión:



movf F-reg, 0
sublw 00H
btfls STATUS, Z

bit 2
(Z)

F-reg
-F-reg
0-
6-
¿Se levantaría
algun flag?
Z=0 N=1



43

Los saltos BRANCH

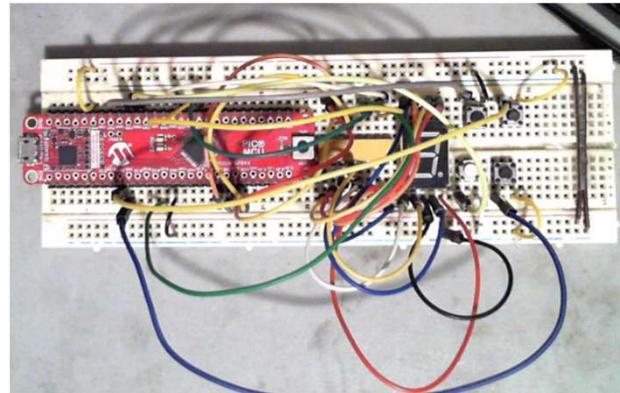
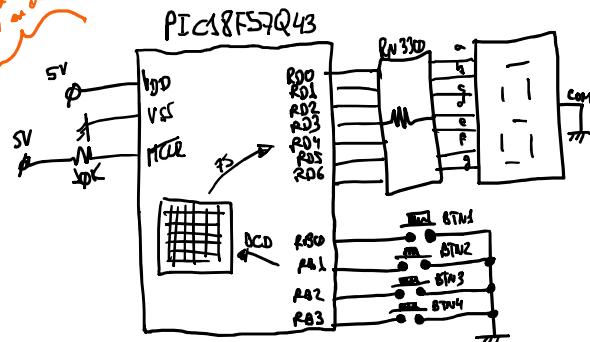
- Son saltos cortos condicionales en base a los flags del registro STATUS

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

44

Ejemplo 2024-1 Sem4

No Tengo
que
de
Cabeza
Pero



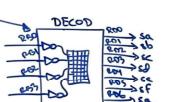
45

Ejemplo 2024-1 Sem4

Ejemplo de aplicación: Decodificador BCD → 7SEG

Observaciones del hardware:

- 1: Entradas del decodificador: RB0 ~ RB3
son activas en bajo
- 2: Salidas del decodificador: RD0 ~ RD6
salidas digitales



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

- complemento
- enmascaramiento

Procediendo "LOOK UP TABLES" o tablas de respuesta

$$\text{uint}_8 \text{ tabla}[] = \{0x3f, 0x06, 0x5b, 0x4f, \dots\}$$

uint_8 salida;

$$\text{salida} = \text{tabla}[\text{numero} \& 0x0f];$$

vector de RST → out Memoria de Páginas

Dirección página

Zona de programación 0x0ff

Memoria de Páginas

TBLPTR (table pointer)

clr TBLPTRU movlw 0EH

movwf TBLPTRH

clr TBLPTRL

tblrd* lec de memoria de páginas

el resultado se guarda en TBLPTR

Bank 0 Bank 1 Bank 2 Bank 3

Memoria Datos

Bank 4

Memoria Datos

<p

Ejemplo 2024-1 Sem4

```

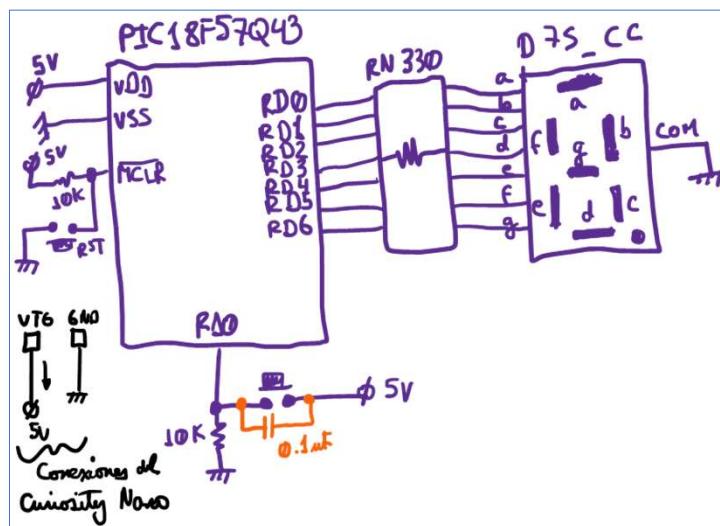
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6    ORG 000000H
7    bra configuro
8
9    ORG 001000H
10   tabla_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12   ORG 000100H
13   configuro:
14     movlb 0H
15     movlw 60H
16     movwf OSCCON1, 1
17     movlw 03H
18     movwf OSCFRQ, 1
19     movlw 40H
20     movwf OSCEN, 1
21     movlb 4H
22     movlw 0FFH
23     movwf TRISB, 1
24     movlw 0F0H
25     movwf ANSELB, 1
26     movlw 0FH
27     movwf WPUB, 1
28     movlw 80H
29     movwf TRISD, 1
30     movwf ANSELD, 1
31     movlw 00H
32     movwf TBLPTRU, 1
33     movlw 10H
34     movwf TBLPTRH, 1
35     movlw 00H
36     movwf TBLPTRL, 1
37
38 inicio:
39     comf PORTB, 0, 1      ;complemento PORTB y lo mando a Wreg
40     andlw 0FH              ;enmascaramiento para que pase solo los 4 menos sign
41     movwf TBLPTRL, 1        ;cambio de dirección de apunte de TBLPTR según entra
42     TBLRD*                 ;acción de lectura del TBLPTR
43     movff TABLAT, LATD     ;mando el contenido leído al puerto del display
44     bra inicio
45
46 end upcino

```

47

Ejemplo 2024-2

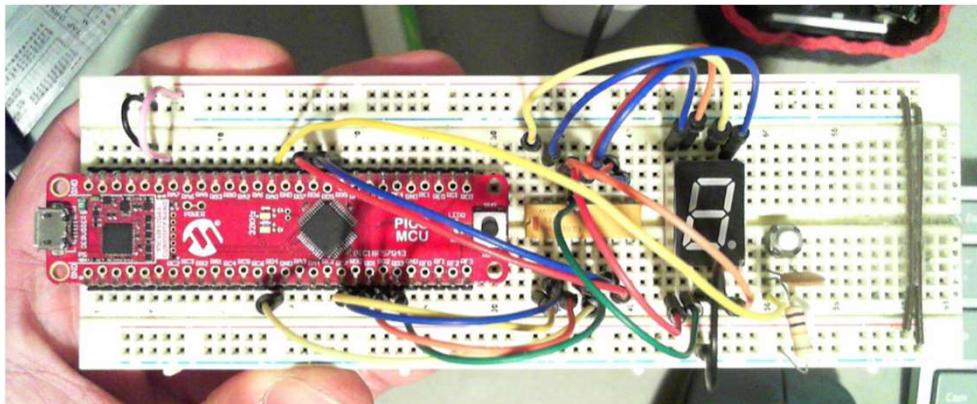
- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.



48

Ejemplo 2024-2

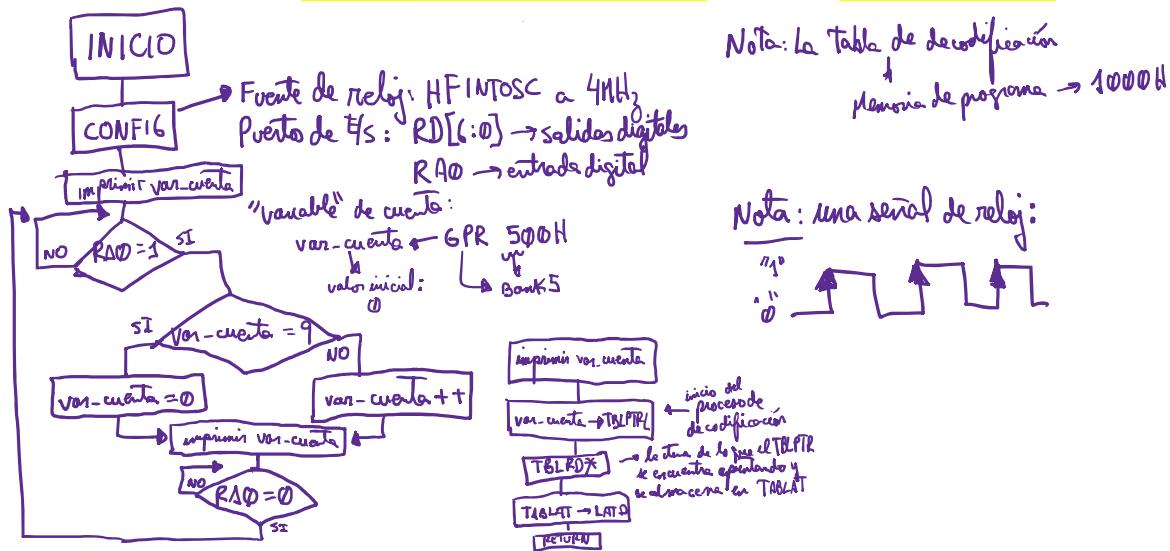
- Desarrollar un contador 0-9 con una entrada de cuenta activa en alto y visualización en display de siete segmentos del tipo cátodo común.



49

Ejemplo 2024-2

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



50

Ejemplo 2024-2

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

```

1      PROCESSOR 18F57Q43
2      #include "cabecera.inc"
3
4      var_cuenta EQU 500H      ;un label al GPR 500H
5
6      PSECT upcino, class=CODE, reloc=2, abs
7      upcino:
8          ORG 000000H
9          bra configuro
10         ORG 000800H
11         decod_bcd: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
12             ; 0   1   2   3   4   5   6   7   8   9
13
14         ORG 000100H
15         configuro:
16             ;configuracion de la fuente de reloj
17             movlb 0H
18             movlw 60H
19             movwf OSCCON1, 1
20             movlw 02H
21             movwf OSCFRQ, 1
22             movlw 40H
23             movwf OSCEN, 1
24             ;configuracion de las E/S
25             movlb 4H
26             movlw 80H
27             movwf TRISD, 1      ;RD[6:0] como salidas
28             movwf ANSEL0, 1      ;RD[6:0] como digital
29             bsf TRISA, 0, 1     ;RA0 como entrada
30             bcf ANSELA, 0, 1     ;RA0 como digital
31
32         ;configuraciones adicionales
33         clrf TBLPTRU, 1
34         movlw 08H
35         movwf TBLPTRH, 1
36         clrf TBLPTRL, 1      ;TBLPTR direccinado a 0800H en la mem de p
37         movlb 5H
38         clrf var_cuenta, 1   ;var_cuenta con valor inicial cero
39         movlb 4H
40         call imprimir_var_cuenta
41
42         inicio_contador:
43             btfss PORTA, 0, 1  ;pregunto si presione boton
44             bra inicio_contador ;falso, no presione boton
45             movlb 5H             ;verdad
46             movlw 9
47             cpfseq var_cuenta, 1 ;pregunto si var_cuenta=9
48             bra no_es_nueva      ;falso
49             bra si_es_nueva       ;verdad
50             no_es_nueva:
51                 incf var_cuenta, 1, 1  ;incremento var_cuenta
52                 bra siguiente
53             si_es_nueva:
54                 clrf var_cuenta, 1   ;mando a cero var_cuenta
55             siguiente:
56                 call imprimir_var_cuenta
57             aun_no:
58                 btfsc PORTA, 0, 1    ;pregunto si dejaste de presionar boton
59                 bra aun_no
60                 bra inicio_contador
61
62         imprimir_var_cuenta:
63             movlb 5H
64             movwf var_cuenta, 0, 1 ;muevo var_cuenta a Wreg
65             movlb 4H
66             movwf TBLPTRL, 1      ;dirección del TBLPTR
67             TBLRD*
68             movwf TABLAT, LATD    ;acción de lectura
69             return
70
71         end upcino

```

51

Códigos de ejemplos 2024-2

• Decodificador BCD-7Segmentos

```

1      PROCESSOR 18F57Q43
2      #include "cabecera.inc"
3
4      PSECT upcino, class=CODE, reloc=2, abs
5      upcino:
6          ORG 000000H
7          bra configuro
8
9          ORG 001000H
10         decod7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11             ; 0   1   2   3   4   5   6   7   8   9
12
13         ORG 000100H
14         configuro:
15             ;conf fuente de reloj
16             movlb 0H           ;voy al Bank0
17             movlw 60H
18             movwf OSCCON1, 1
19             movlw 02H
20             movwf OSCFRQ, 1
21             movlw 40H
22             movwf OSCEN, 1
23             ;conf puertos E/S
24             movlb 4H           ;voy al Bank4
25             setf TRISB, 1      ;RB[3:0] como entradas
26             movlw 0FOH
27             movwf ANSEL0, 1      ;RB[3:0] como digitales
28             bsf TRISE, 0, 1     ;RE0 como entrada
29             bcf ANSELA, 0, 1     ;RE0 como digital
30             movlw 80H
31             movwf TRISD, 1      ;RD[6:0] como salida
32             movwf ANSELD, 1      ;RD[6:0] como digital
33
34         ;conf adicional
35         clrf TBLPTRU, 1
36         movlw 10H
37         movwf TBLPTRH, 1      ;          U H L
38         clrf TBLPTRL, 1      ;TBLPTR apuntando a 001000H
39
40         inicio_deco:
41             movf PORTB, 0, 1    ;Leo el puerto RB[3:0] y lo almaceno en Wreg
42             andlw 0FH            ;Enmascaramiento de los cuatro LSB de Wreg
43             movwf TBLPTRL, 1      ;asignamos el valor de Wreg hacia TBLPTRL
44             TBLRD*                ;acción de lectura de lo apuntado por TBLPTR
45             movff TABLAT, LATD    ;muevo contenido leido y alojado en TABLAT hacia RD
46             bra inicio_deco
47
48         end upcino

```

52

```

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

```

Códigos de ejemplos 2024-2

ecodificador BCD-7Segmentos

53

```

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

```

Códigos de ejemplos 2024-2

• Decodificador BCD-7Segmentos

54

<pre> 1 PROCESSOR 18F57Q43 2 #include "cabecera.inc" 3 4 PSECT upcino, class=CODE, reloc=2, abs 5 upcino: 6 ORG 000000H 7 bra configuro 8 9 ORG 001000H 10 deco_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H 11 12 ORG 000100H 13 configuro: 14 ;conf fuente de reloj: 15 movlb 0H 16 movlw 60H 17 movwf OSCCON1, 1 18 movlw 02H 19 movwf OSCFRQ, 1 20 movlw 40H 21 movwf OSCEN, 1 22 ;conf puertos E/S: 23 movlb 4H 24 movlw 80H 25 movwf TRISD, 1 ;RD[6:0] como salidas 26 movwf ANSEL0, 1 ;RD[6:0] como digitales 27 setf TRISB, 1 ;RB[3:0] como entradas 28 movlw OFOH 29 movwf ANSELB, 1 ;RB[3:0] como digitales 30 bsf TRISE, 0, 1 ;RE0 como entrada 31 bcf ANSELE, 0, 1 ;RE0 como digital </pre>	<pre> 29 ;conf adicional 30 clrf TBLPTRU, 1 31 movlw 10H 32 movwf TBLPTRH, 1 33 clrf TBLPTRL, 1 ;TBLPTR esta apuntando a 001000H 34 35 inicio_deco: 36 movf PORTB, 0, 1 ;leo RB y lo mando a Wreg 37 andlw 0FH ;enmascaramiento para que pase solo RB[3:0] 38 movwf TBLPTRL, 1 ;movemos Wreg a TBLPTRL para iniciar decod 39 TBLRD* ;accion de lectura de lo apuntado por TBLPTR 40 movff TABLAT, LATD ;mueve contenido de TABLAT a LATD 41 bra inicio_deco 42 43 end upcino </pre>
--	---

55

<pre> 1 PROCESSOR 18F57Q43 2 #include "cabecera.inc" 3 4 cuenta EQU 500H ;asignacion de label a GPR 500H 5 6 PSECT upcino, class=CODE, reloc=2, abs 7 upcino: 8 ORG 000000H 9 bra configuro 10 11 ORG 001000H 12 decod: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H 13 ; 0 1 2 3 4 5 6 7 8 9 14 15 ORG 000100H 16 configuro: 17 ;configuracion de la fuente de reloj 18 movlb 0H 19 movlw 60H 20 movwf OSCCON1, 1 21 movlw 02H 22 movwf OSCFRQ, 1 23 movlw 40H 24 movwf OSCEN, 1 25 ;configuracion de las E/S 26 movlb 4H 27 movlw 80H 28 movwf TRISB, 1 ;RD[6:0] como salidas 29 movwf ANSELB, 1 ;RD[6:0] como digitales 30 setf TRISD, 1 ;RD como entradas 31 movlw OFOH 32 movwf ANSEL0, 1 ;RD[3:0] como digitales 33 bsf TRISE, 0, 1 ;RE0 como entrada </pre>	<pre> 34 bcf ANSELE, 0, 1 ;RE0 como digital 35 ;configuraciones adicionales 36 clrf TBLPTRU, 1 37 movlw 10H 38 movwf TBLPTRH, 1 39 clrf TBLPTRL, 1 ;direccion asignada a TBLPTR=001000H 40 movlb 5H ;al bank 5 41 clrf cuenta, 1 ;valor inicial de cuenta=0 42 movlb 4H ;al bank 4 43 call imp_7s 44 45 inicio_contador: 46 btfs PORTE, 0, 1 ;pregunto si preisone el boton 47 bra inicio_contador ;falso, no presione el boton 48 movlb 5H 49 movlw 9 50 cpifseq cuenta, 1 ;pregunto si cuenta=9 51 bra no_es_nueve 52 bra si_es_nueve 53 no_es_nueve: 54 incf cuenta, 1, 1 ;incremento cuenta 55 bra juntada 56 si_es_nueve: 57 clrf cuenta, 1 ;cuenta igual a cero 58 juntada: 59 call imp_7s 60 movlb 4H 61 btfs PORTE, 0, 1 ;pregunto si solte el boton 62 bra juntada ;falso, sigue pulsado 63 bra inicio_contador 64 65 imp_7s: 66 movlb 5H 67 movf cuenta, 0, 1 ;mando cuenta a Wreg 68 movlb 4H 69 movwf TBLPTRL, 1 ;mueve Wreg a TBLPTRL 70 TBLRD* 71 movff TABLAT, LATB 72 return 73 74 end upcino </pre>
---	--

56

- Contador BCD-7Segmentos

<pre> 1 PROCESSOR 18F57Q43 2 #include "cabecera.inc" 3 4 var_cuenta EQU 500H ;asignando etiqueta a GPR 500H 5 6 PSECT upcino, class=CODE, reloc=2, abs 7 upcino: 8 ORG 000000H 9 bra configuro 10 11 ORG 001000H 12 deco_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H 13 14 ORG 000100H 15 configuro: 16 ;conf fuente de reloj: 17 movlb 0H 18 movlw 60H 19 movwf OSCCON1, 1 20 movlw 02H 21 movwf OSCFRCQ, 1 22 movlw 40H 23 movwf OSCEN, 1 24 ;conf puertos E/S: 25 movlb 4H 26 movlw 80H 27 movwf TRISD, 1 ;RD[6:0] como salidas 28 movwf ANSELD, 1 ;RD[6:0] como digitales 29 setf TRISE, 1 ;RB[3:0] como entradas 30 movlw 0FOH 31 movwf ANSELB, 1 ;RB[3:0] como digitales 32 bsf TRISE, 0, 1 ;RE0 como entrada 33 bcf ANSELE, 0, 1 ;RE0 como digital </pre>	<pre> 34 ;conf adicional 35 clrf TBLPTRU, 1 36 movlw 10H 37 movwf TBLPTRH, 1 38 clrf TBLPTRL, 1 ;TBLPTR esta apuntando a 001000H 39 movlb 5H 40 clrf var_cuenta, 1 ;valor inicial de var_cuenta=0 41 movlb 4H 42 call imp_7s ;nos vamos a Bank4 43 inicio_conta: 44 btfs PORTE, 0, 1 ;nos vamos a Bank5 45 bra \$-2 46 movlb 5H 47 movlw 9 48 cpfseq var_cuenta, 1 ;pregunto si var_cuenta=9 49 bra no_es_nueve 50 clrf var_cuenta, 1 ;verdad, es nueve y lo mandamos a cero 51 bra siguiente 52 no_es_nueve: 53 incf var_cuenta, 1, 1 ;salto a label siguiente 54 siguiente: 55 call imp_7s 56 btfs PORTE, 0, 1 ;pregunto si dejé de pulsar boton 57 bra \$-2 58 bra inicio_conta ;falso, boton sigue presionado 59 60 imp_7s: 61 movlb 5H 62 movf var_cuenta, 0, 1 ;retorno al inicio 63 movlb 4H 64 movwf TBLPTRL, 1 65 TBLRD* 66 movff TABLAT, LATD ;al Bank4 67 return 68 69 end upcino </pre>
--	---

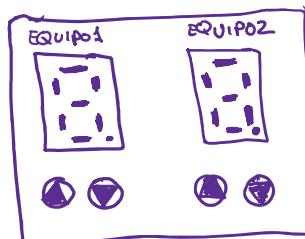
57

Códigos de ejemplos 2024-2

- Contador BCD-7Segmentos

Ejercicios adicionales:

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



58

Fin de la sesión