

Microcontroladores

Semana 4

Semestre 2024-1

Por Kalun José Lau Gan

1

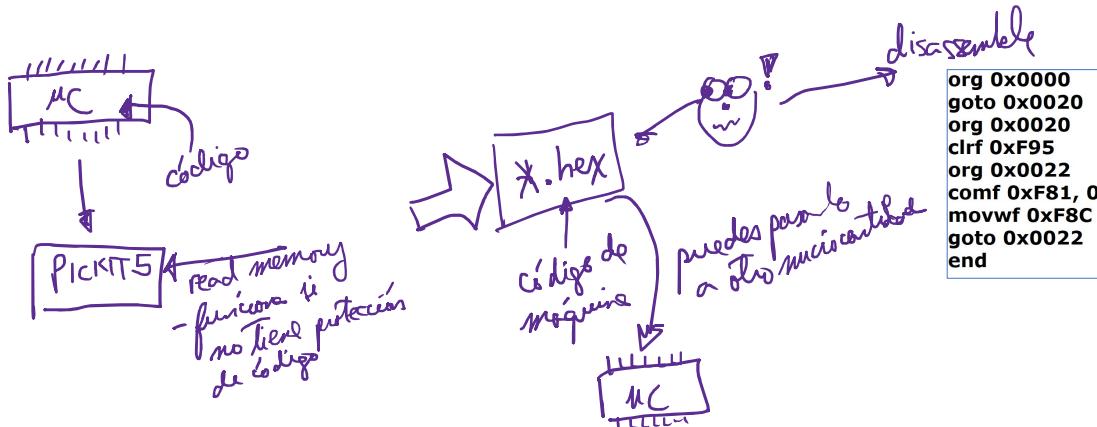
Preguntas previas

- 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: v2.45 (lanzado el 31/08/2023)
- ¿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.

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

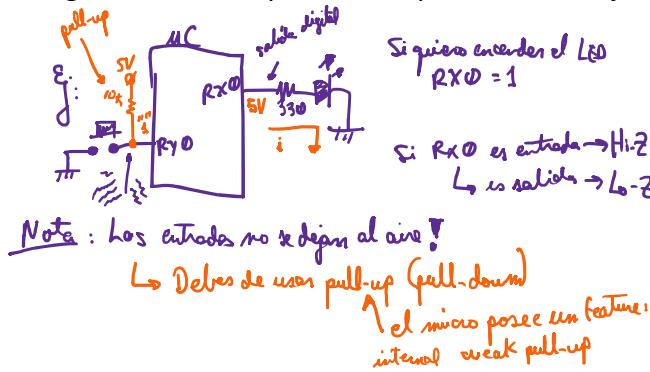
Preguntas previas 2024:

- 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 esta 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 aceras la mano al circuito.
 - Revisar configuración de los puertos empleados en el ejemplo.



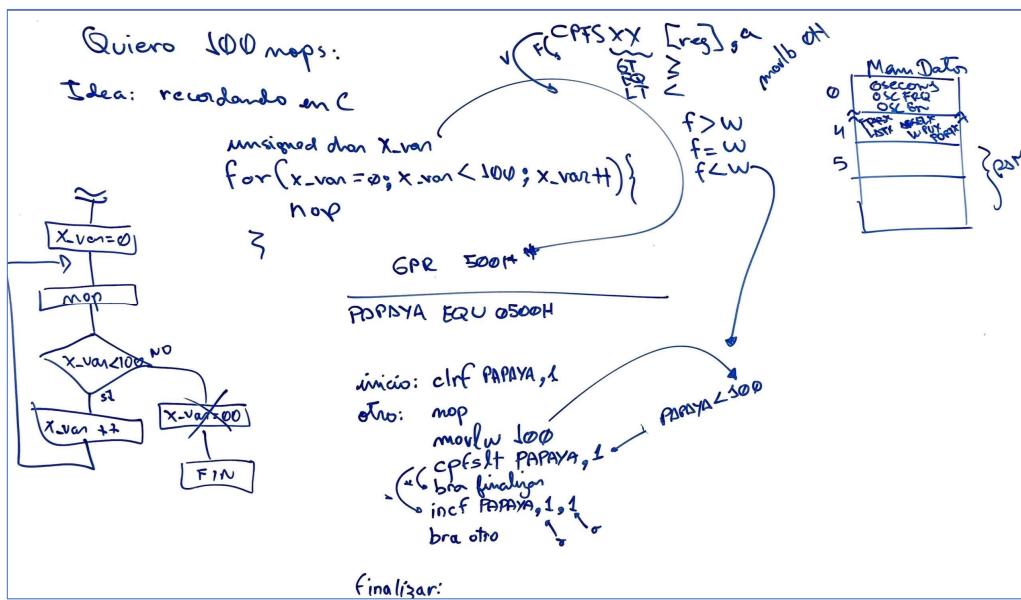
NOTA:
Recordar que los registros para manejar las E/S se encuentran en el BANK4 (revisar Cap 19 del libro)

WPUB.4 = 1

BANK4

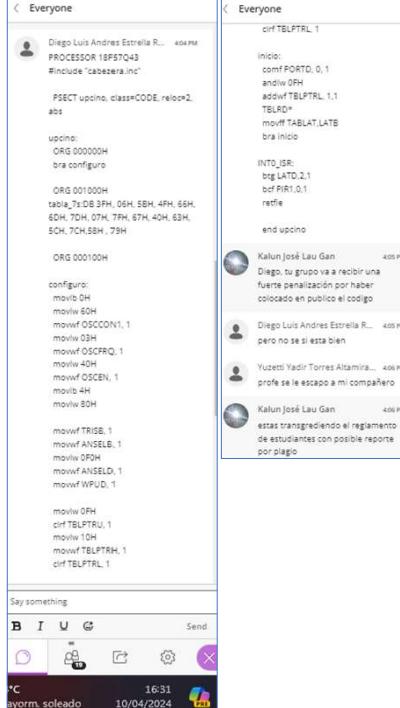
5

Preguntas previas 2024:



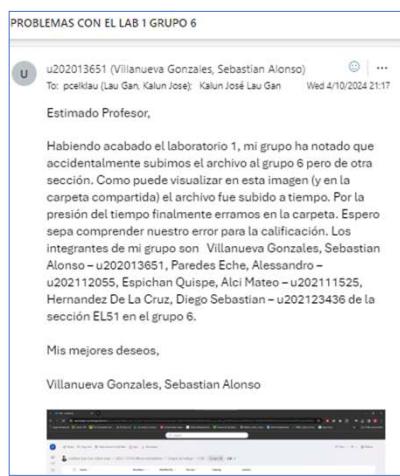
6

No cometer estas acciones durante las evaluaciones



The image shows two Microsoft Word documents side-by-side. Both documents contain assembly language code. The left document has a title bar 'Everyone' and a status bar at the bottom showing '10/04/2024 16:31'. The right document also has a title bar 'Everyone' and a status bar at the bottom showing '10/04/2024 16:31'. The code snippets include instructions like PSECT, upoin, and movlw.

PROBLEMAS CON EL LAB 1 GRUPO 6



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

Estimado Profesor,

Haciendo 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 erranmos en la carpeta. Espero sepa comprender nuestro error para la calificación. Los integrantes de mi grupo son: Villanueva Gonzales, Sebastián Alonso - u202013651, Paredes Eche, Alessandro - u20211055, 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

7

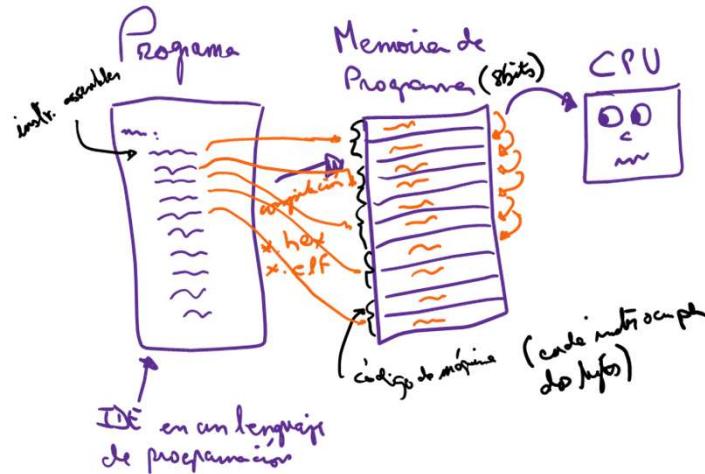
Agenda

- 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

8

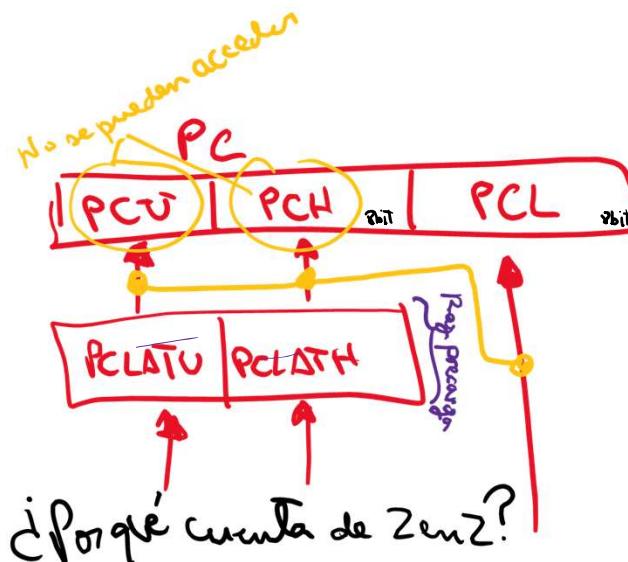
Recordando:

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



9

El Contador de Programa (PC)

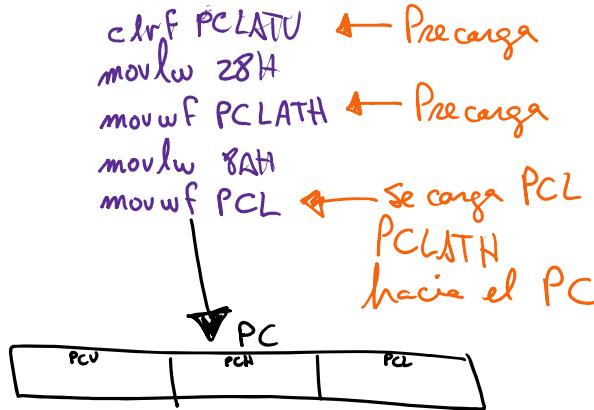


- Indispensable para la ejecución secuencial de las instrucciones.
- Su función es de alojar la dirección de la siguiente instrucción que el CPU va a ejecutar.
- Según hoja técnica, consta de 21 bits separados en tres registros: PCU, PCH y PCL.
- PCU y PCH no son accesibles, para que se pueda escribir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

10

Ejemplo

- Cargar dirección 00288AH en PC:



Nota: El banco externo debe de ser el 4

9.8 Register Summary - Memory Organization

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x4040	FSR2	7:0								FSRL[7:0] FSRH[5:0]
0x4048	PLUSW2	7:0								PLUSW[7:0]
0x404C	PREN2	7:0								PREN[7:0]
0x404D	POSTDEC2	7:0								POSTDEC[7:0]
0x404E	POSTINC2	7:0								POSTINC[7:0]
0x404F	INDF2	7:0								INDF[7:0]
0x4050	BSR	7:0								BSR[5:0]
0x4051	FSR1	7:0								FSR1[7:0] FSRH[5:0]
0x4058		15:8								
0x4059	PLUSW1	7:0								PLUSW[7:0]
0x405C	PREN1	7:0								PREN[7:0]
0x405D	POSTDEC1	7:0								POSTDEC[7:0]
0x405E	POSTINC1	7:0								POSTINC[7:0]
0x405F	INDF1	7:0								INDF[7:0]
0x4060	WREG	7:0								WREG[7:0]
0x4065	FSR0	7:0								FSR0[7:0] FSRH[5:0]
0x4066		15:8								
0x406B	PLUSW0	7:0								PLUSW[7:0]
0x406C	PREN0	7:0								PREN[7:0]
0x406D	POSTDEC0	7:0								POSTDEC[7:0]
0x406E	POSTINC0	7:0								POSTINC[7:0]
0x406F	INDF0	7:0								INDF[7:0]
0x4070	Reserved									
0x407F										
0x4080	PCL	7:0								PCL[7:0]
0x408A	PCLAT	7:0								PCLATH[7:0]
0x40FC	STKPTR	7:0								STKPTR[0:0]
0x40FD	TOS	7:0								TOS[7:0]
		15:8								TOS[16:0]
		23:16								TOS[20:16]

11

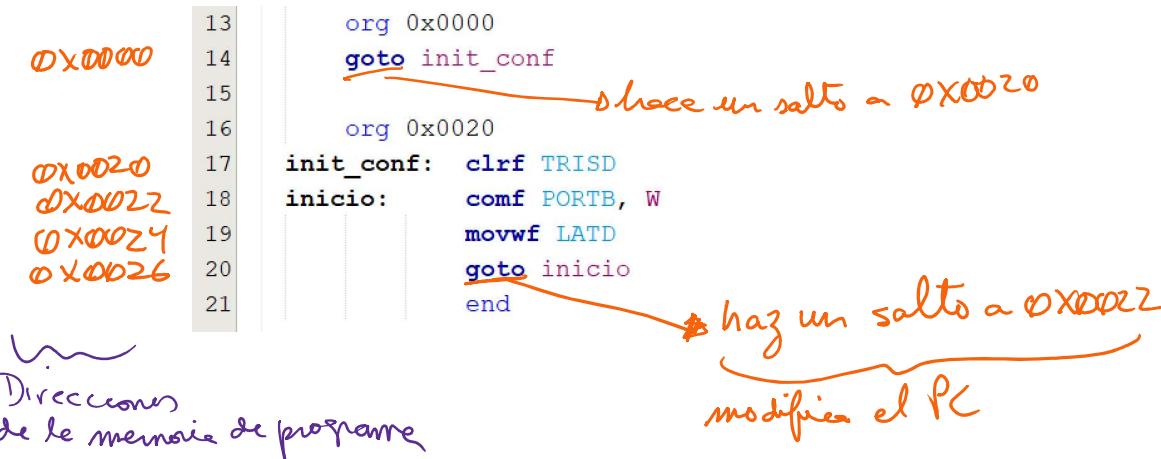
Ejemplo

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

El mismo código de abajo pero sin etiquetas:

```

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



12

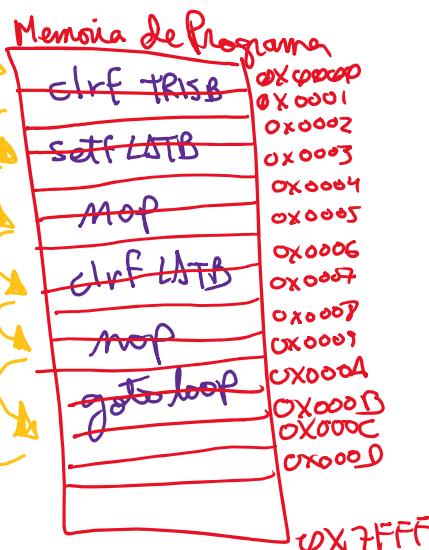
¿Cómo funciona el PC?

- Un programa simple:

```

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

Nota: PC va de dos en dos



13

¿Cómo funciona el PC?

- Se tiene el siguiente programa

```

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

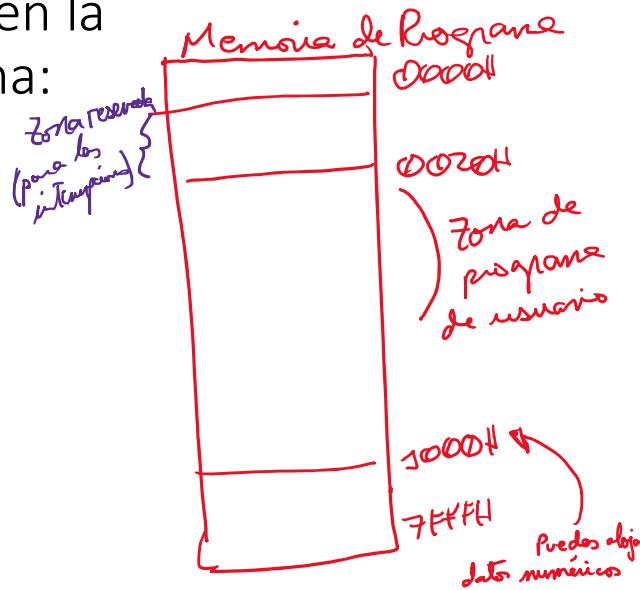
Nota: Las instrucciones en MPASM ocupan 2 bytes
Nota: Se comprueba que PC va de dos en dos, no puede contener una dirección impar



14

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)



15

Otro ejemplo de PC:

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

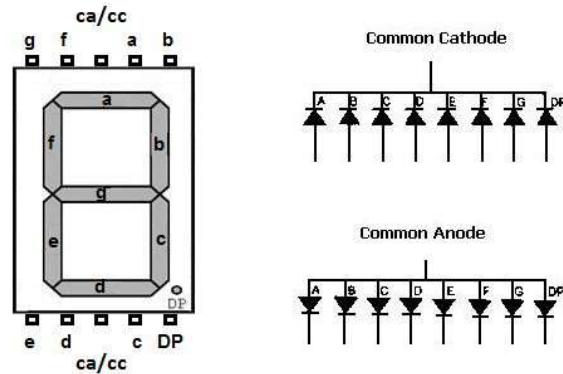
bra 86H {

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

16

El display de siete segmentos

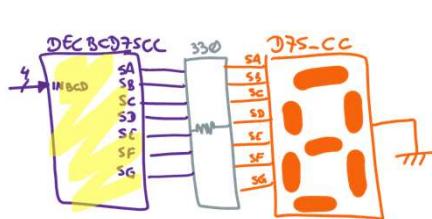
- Dos tipos: ánodo común y cátodo común
- Tablas de decodificación diferentes entre ellos.
- Es necesario colocarle resistencias limitadoras de corriente para cada segmento que compone el display (100ohm a 1kohm).
- Evitar usar solo una resistencia en el pin común del display ya que al cambiar de carácter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



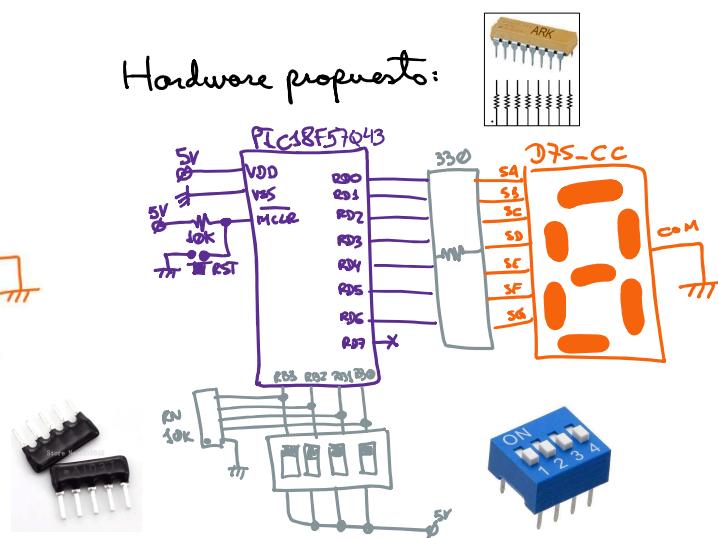
17

Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:



Hardware propuesto:



18

Ejemplo 2024-1 Sem4

```

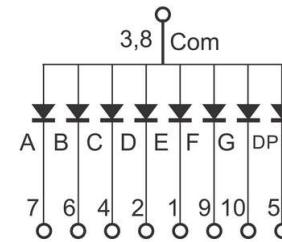
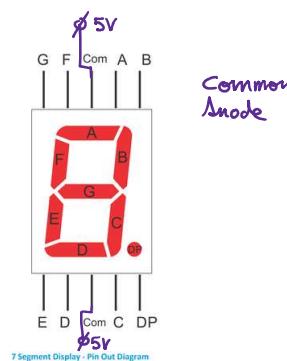
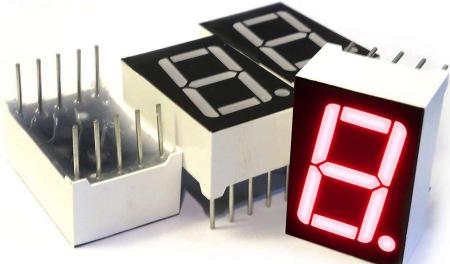
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6    ORG 000000H
7    bra configuro
8
9    ORG 001000H
10   tabla_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12   ORG 000100H
13   configuro:
14     movlb 0H
15     movlw 60H
16     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

```

19

Observación:

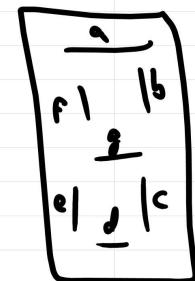
- Los pines denominados “comunes” (Com) son el mismo nodo, la misma conexión!



20

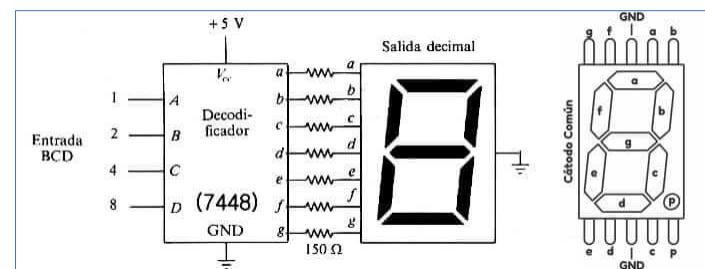
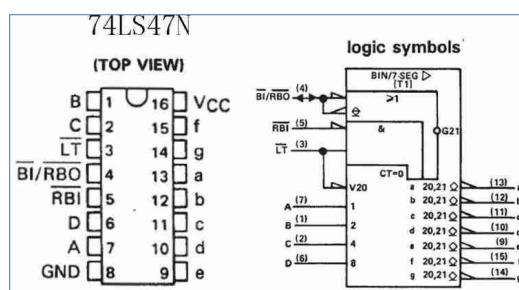
Desarrollo de la tabla de decodificación para display de 7 segmentos cátodo común:

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



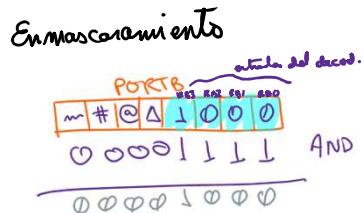
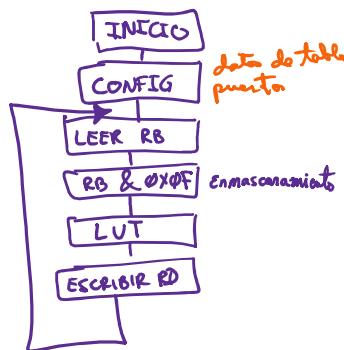
21

Circuitos TTL decodificadores de siete segmentos



22

Diagrama de flujo



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
      
```

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 :

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!**

23

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

24

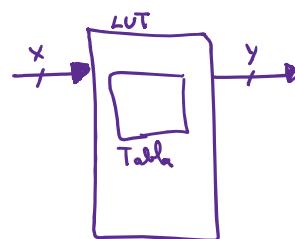
Acerca de la memoria de programa

- 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.

25

Tablas de búsqueda (lookup tables - LUT)

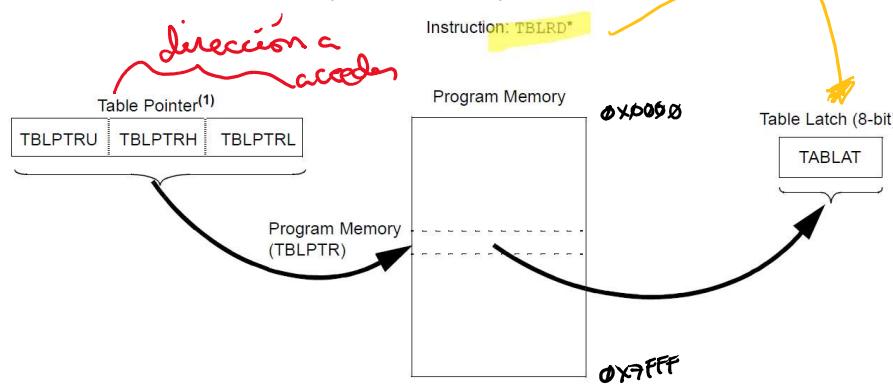
- Decodificadores implementados en software



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

26

El puntero de tabla (TBLPTR)

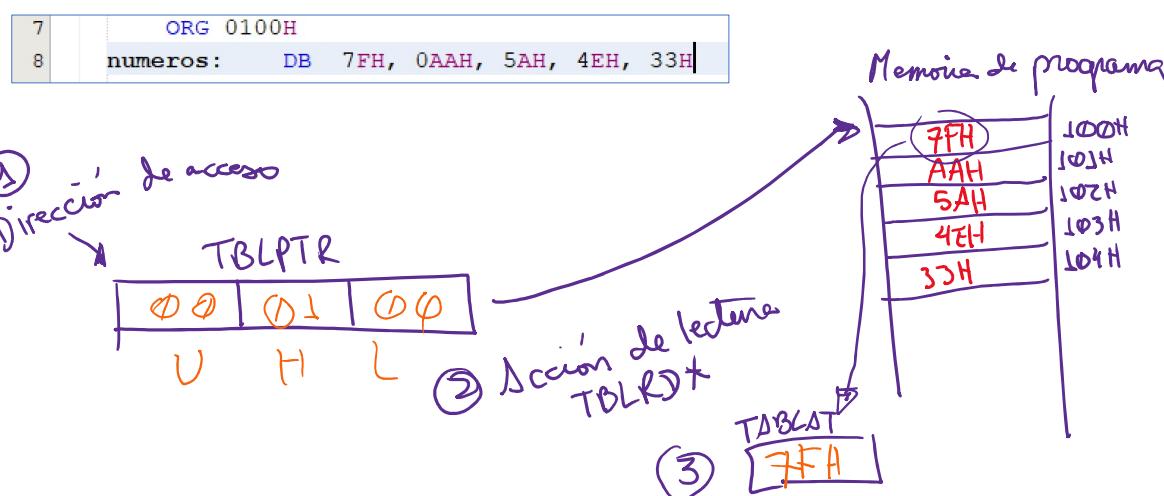


Note 1: Table Pointer register points to a byte in program memory.

- 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

27

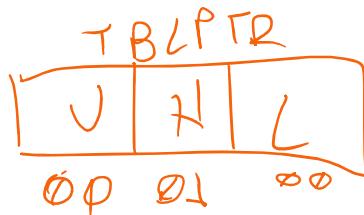
Operación con el TBLPTR



28

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 a WReg
27             movwf LATD            ;Muevo contenido de WReg hacia LATD
28             goto loop
29         end principal

```

29

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.

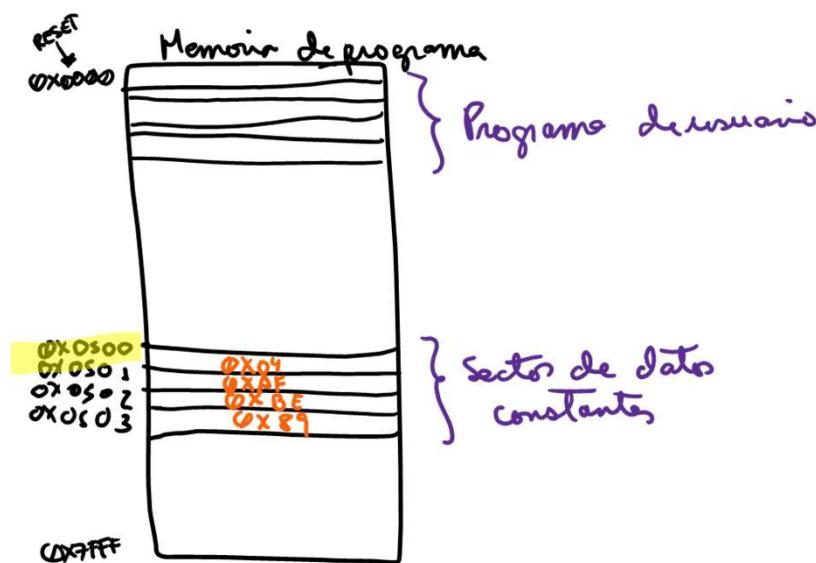
30

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

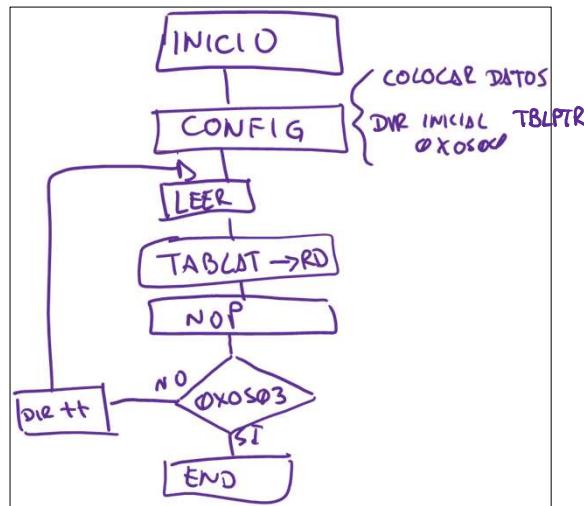
31

Desarrollo del ejemplo:



32

Diagrama de flujo:



33

Resumen de instrucciones con toma de decisión:



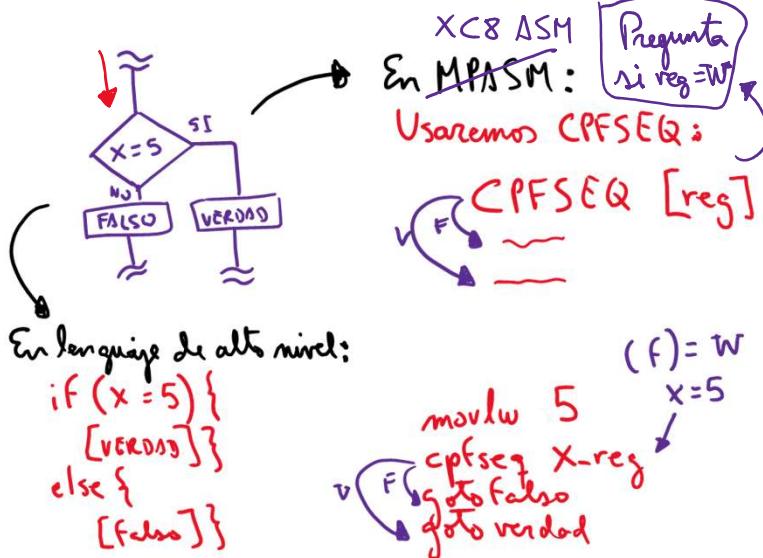
- btfss, btfsc
 - Prueba el #bit de [registro] si es 0 ó 1
- decfsz, incfsz
 - Decrementa o incrementa [registro] y pregunta si es cero.
- dcfsnz, icfsnz
 - Decrementa o incrementa [registro] y pregunta si **no** es cero
- cpfsgt, cpfseq, cpfslt
 - Comparaciones numéricas entre [registro] y W
- tstfsz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

34

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

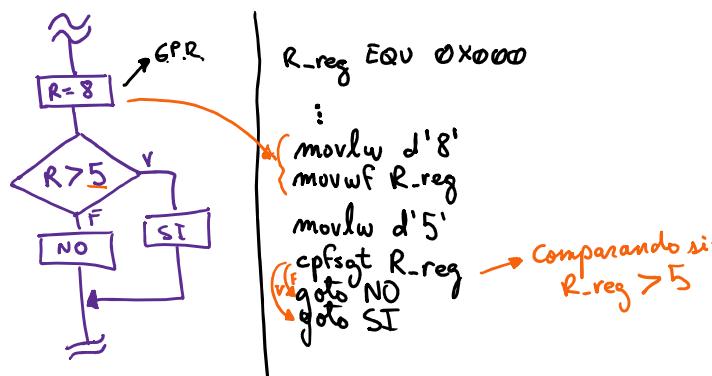
$\text{CPFSEQ} \rightarrow f = \text{Wreg}$
 $\text{CPFS LT} \rightarrow f < \text{Wreg}$
 $\text{CPFS GT} \rightarrow f > \text{Wreg}$

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



35

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

36

Ejercicio:

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

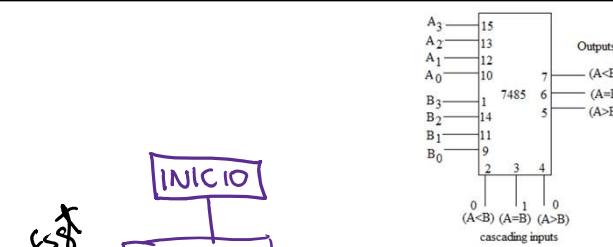
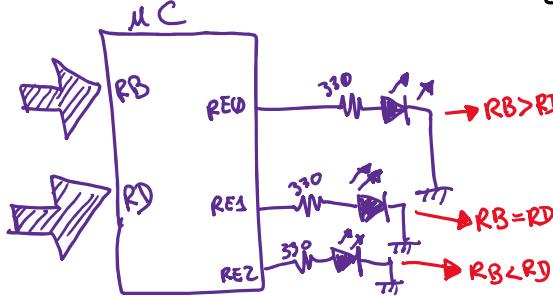
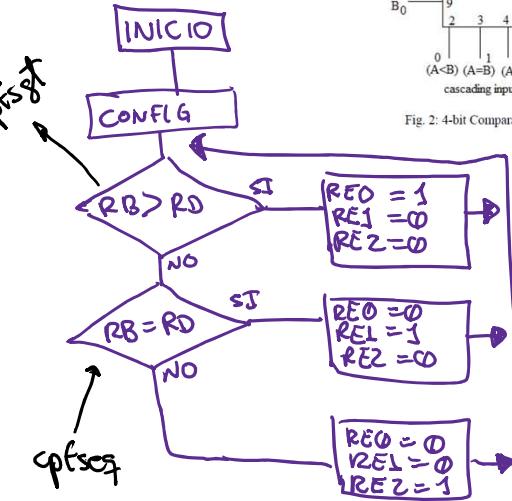
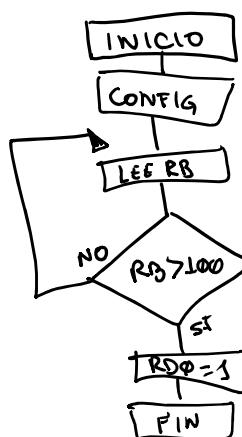
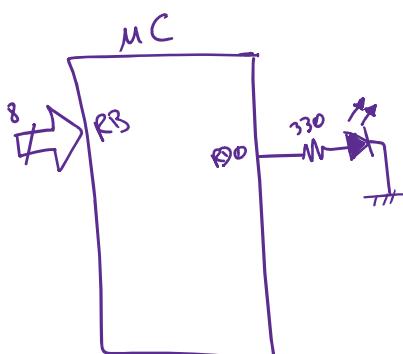


Fig. 2: 4-bit Comparator



37

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



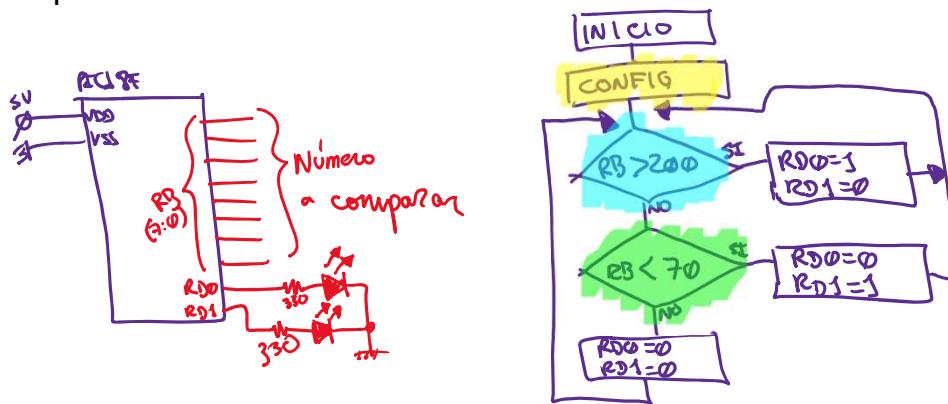
$f > w_{reg}$

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

38

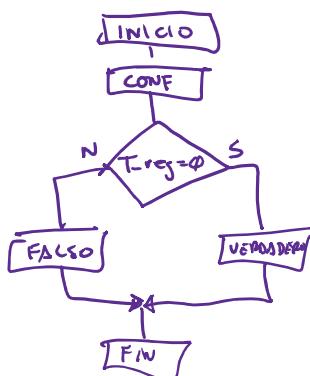
Ejercicio:

- 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.



39

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
d → goto VERDADERO

Opción 2:

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

Opción 3:

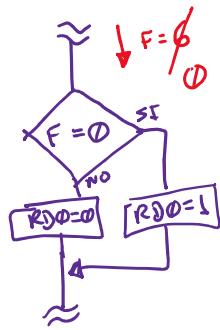
inicio: movf T-reg, W
sublw .0
btfs STATUS, Z
v → goto FALSO
d → goto VERDADERO

Opción 4:

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

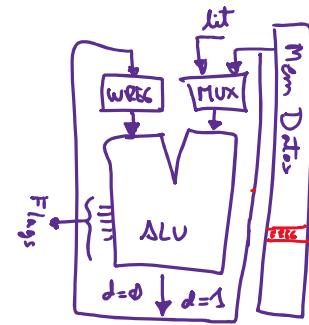
40

Aplicación de sublw para tomas de decisión:



~~morf F-reg, 0~~
~~sublw 00H~~
~~btfss STATUS, Z~~

~~F-reg~~
~~-F-reg~~
~~0-~~
~~6-~~
~~6~~
bit 2 (Z)
¿Se levantaría algún flag?
~~Z=0 N=1~~



41

Los saltos BRANCH

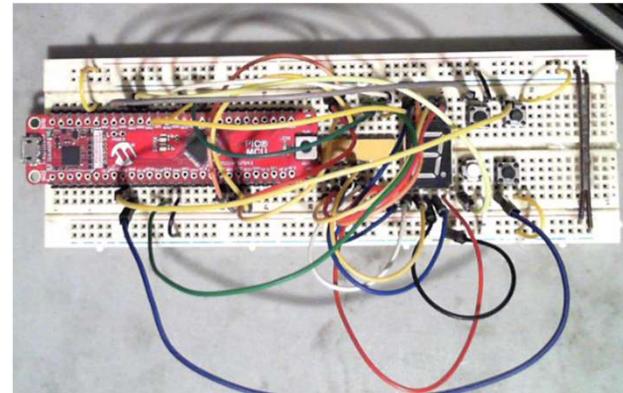
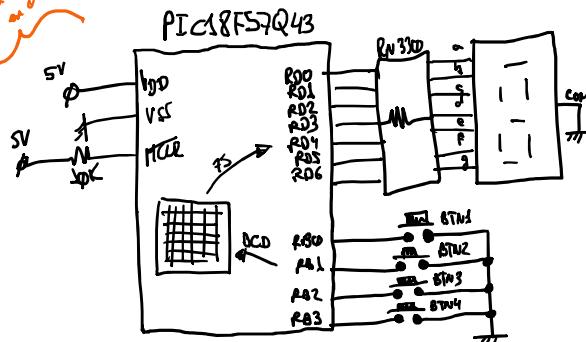
- Son saltos cortos condicionales en base a los flags del registro STATUS

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

42

Ejemplo 2024-1 Sem4

No tiene fuente de alimentación



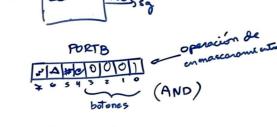
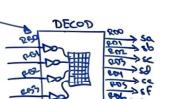
43

Ejemplo 2024-1 Sem4

Ejemplo de aplicación: Decodificador BCD → 7SEG

Observaciones del hardware:

- 1: Entradas del decodificador: RB0 ~ RB3
son activas en bajo
entradas digitales con pull up activo
- 2: Salidas del decodificador: RD0 ~ RD6
salidas digitales



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

- complemento
- cimarronamiento

Procediendo "LOOK UP TABLES" o tablas de respuesta

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

uint_8 salida;

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

vector de RST → out Memoria de Pags

Dirección apunta a TBLPTR (table pointer)

clr TBLPTRU movlw 0EH movwf TBLPTRH

clr TBLPTRL movwf TBLPTRL

tblptr* lec de memoria de pag

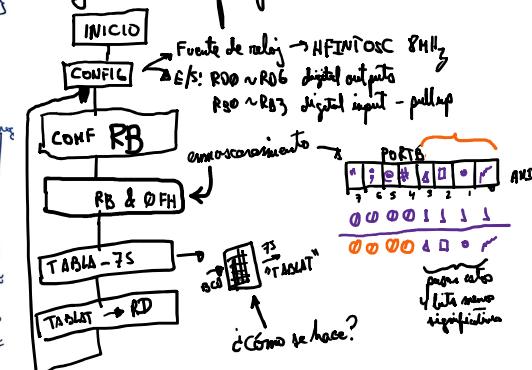
Memoria Datos

Bank 0 Bank 4

3FH TABLAT movff

LATD

Diagrama de flujo:



44

Ejemplo 2024-1 Sem4

```

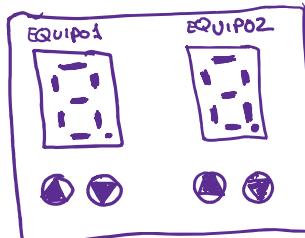
1  PROCESSOR 18F57Q43
2  #include "cabecera.inc"
3
4  PSECT upcino, class=CODE, reloc=2, abs
5  upcino:
6    ORG 000000H
7    bra configuro
8
9    ORG 001000H
10   tabla_7s: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 67H
11
12   ORG 000100H
13   configuro:
14     movlb 0H
15     movlw 60H
16     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

```

45

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.



46

Fin de la sesión