

Microcontroladores

Semana 3

Semestre 2023-1

Profesor: Kalun José Lau Gan

1

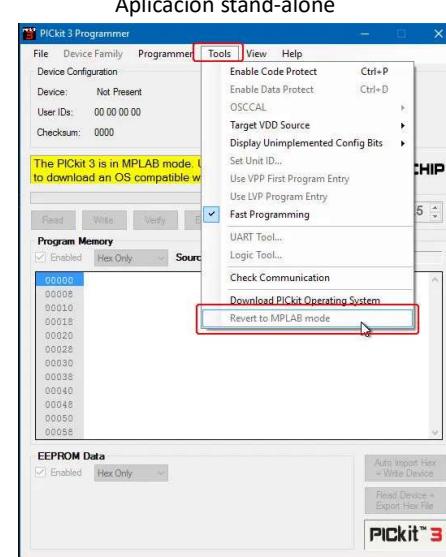
Preguntas previas:

- Sigo teniendo problemas en la implementación de los ejemplos vistos en clases anteriores
 - Verificar que al multímetro le funcione correctamente la continuidad y sus puntas de prueba (probes) estén en buen estado.
 - Asegurarse que los cables jumper tengan continuidad (emplear el multímetro)
 - Cuando tengan algún mensaje de error 0x0 en el MPLABX es producto de que no hay conexión entre el PICKIT3
 - Revisar conexiones entre el pickit3 y el protoboard (los cinco cables: Vpp, Vdd, Vss, PGD y PGC)
 - Revisar si PGD del pickit3 esté conectado al pin40 y PGC del pickit3 esté conectado al pin39.
 - Cerrar el MPLABX y abrirlo de nuevo
 - Desconectar el PICKIT3 de la PC y volverlo a conectar
 - Me salen mensajes de que el PICKIT3 no esta entregando energía suficiente
 - Revisen la configuración de ahorro de energía de tu laptop/PC
 - En la configuración del PICKIT3 (parámetros de POWER), manipular el valor del voltaje que el PICKIT3 entrega (por defecto es 5.00V, pueden bajarlo a 4.75V, 4.5V, 4.25V)
 - Como último recurso, emplear una fuente externa, teniendo en cuenta en desactivar la opción de Power en el PICKIT3. No olvidar conectar la tierra de la fuente externa al circuito.
 - En algunos casos los protoboard tienen las líneas de alimentación partidas en la mitad, asegurarse de conectarlos antes de implementar los circuitos.
 - Emplear el canal de contacto de email UPC para enviar imágenes (fotos y/o capturas de pantalla) y videos ya que la plataforma del blackboard no permite el envío de archivos multimedia.

2

Preguntas previas:

- El PICKIT3 al conectarlo a la PC el software MPLABX no lo reconoce.
 - Leer la consulta siguiente.
- PICKIT 3 lo operaba antes con el software stand-alone y no funciona con el MPLABX. ¿Qué hay que hacer?
 - Entrar a la aplicación stand-alone de pickit3 y cambiar el firmware para que opere en modo MPLAB
 - Cerrar la aplicación stand-alone, desconectar el pickit3, regresar al MPLABX y reconectar el pickit3.



3

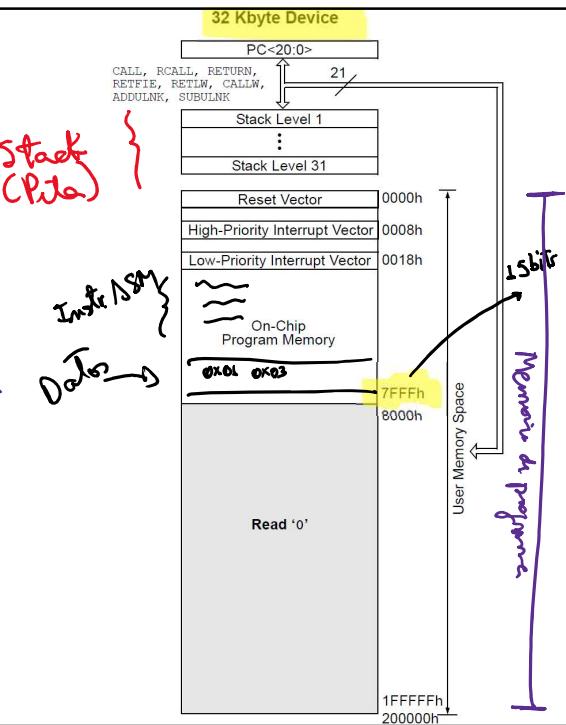
Agenda:

- Memoria de programa del microcontrolador PIC18F45K50
- El contador de programa (PC) del CPU del microcontrolador PIC18F45K50
- Acceso a datos almacenados en la memoria de programa empleando el puntero del tabla (TBLPTR)
- Memoria de datos del microcontrolador PIC18F45K50: acceso mediante punteros FSRx/INDFx
- Interface a display de siete segmentos
- Instrucciones CPFSEQ, CPFSLT, CPFSGT en MPASM

4

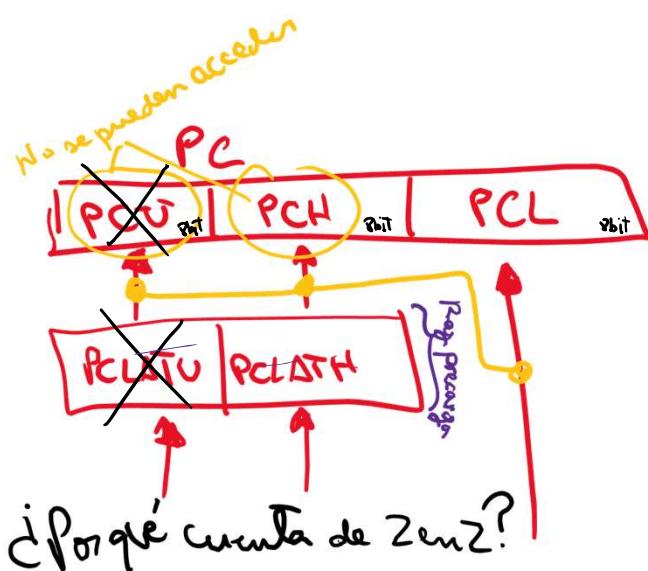
Memoria de programa del PIC18F45K50

- Referencia: Item 6, página 73 de hoja técnica.
- Tamaño: 32KByte (0x0000 a 0xFFFF).
- Instrucciones ocupan 2bytes, en casos especiales ocupan hasta 4bytes (revisar Item 27 de hoja técnica)
- Se puede usar esta memoria para almacenar constantes de 8 bits.
- Se presenta el bloque de pila (stack) de 30 niveles, usado para el almacenamiento temporal de la dirección de retorno.
- A partir de 0x8000 si se escribe algo y luego se lee, se obtendrá 0.
- Tamaño máximo de la memoria de programa de la familia PIC18: 2Mbyte (21 bits en direcciones)
- Esta memoria es NO VOLÁTIL (Flash EEPROM)



5

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.
- Al escribir en PCL se subirán PCLATH y PCLATU para así subir los 21 bits a la vez
- En el PIC18F45K50 no está implementado PCU debido al tamaño de la memoria de programa (32KB ó 15 bits de direccionamiento).

6

Ejemplo

- Cargar dirección 00288AH en PC:

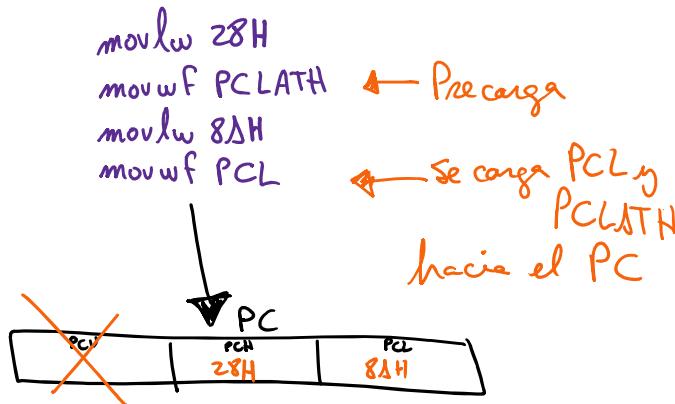


TABLE 5-1: SPECIAL FUNCTION REGISTER MAP			
Address	Name	Address	Name
FFfh	TOSU	FDFh	INDF2 ⁽¹⁾
FFEh	TOSH	FDEh	POSTINC2 ⁽¹⁾
FFDh	TOSL	FDDh	POSTDEC2 ⁽¹⁾
FFCh	STKPTR	FDCh	PREINC2 ⁽¹⁾
FFBh	PCLATU	FDBh	PLUSW2 ⁽¹⁾
FFAh	PCLATH	FDAh	FSR2H
FFJh	PCL	FD9h	FSR2L
FF8h	TBLPTRU	FD8h	STATUS
FF7h	TBLPTRH	FD7h	TMR0H
FF6h	TBLPTRL	FD6h	TMR0L
FESh	TARLAT	FD5h	TOCON

7

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

```

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

```

13 14 15 16 17 18 19 20 21

0x0000 0x0020 0x0022 0x0024 0x0026

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Direcciones de la memoria de programación

shoe un salto a 0x0020

haz un salto a 0x0022

modifica el PC

8

En el Proteus (Solo en MPASM con archivo COF):

```

----- ;este es un comentario
----- list_p=18f4550      ;modelo de procesador
----- #include <p18f4550.inc> ;libreria de nombre de registros
----- ;Bits de configuracion del microcontrolador
----- CONFIG FOSC = XT_XT      ; Oscillator Selection bits (XT oscil
----- CONFIG PWRT = ON          ; Power-up Timer Enable bit (PWRT ena
----- CONFIG BOR = OFF           ; Brown-out Reset Enable bits (Brown-
----- CONFIG WDT = OFF           ; Watchdog Timer Enable bit (WDT disa
----- CONFIG PBADEN = OFF        ; PORTB A/D Enable bit (PORTB<4:0> pi
----- CONFIG LVP = OFF           ; Low Voltage Programming (LVP) enable
----- CONFIG MCLRE = OFF         ; MCLR Pin Function Selection (MCLRE)
----- org 0x0000
0000  goto init_conf
----- org 0x0020
0020  init_conf: clrf TRISD
0022  inicio:    comf PORTB, W
0024  movwf LATD
0026  goto inicio
----- end
-----
```

Watch Window			
Name	Address	Value	Watch Expression
TRISD	0XF95	0b00000000	
TRISB	0XF93	0b11111111	
PORTB	0XF81	0b10111110	
PORTD	0XF83	0b00000000	
LATB	0XF8A	0b00000000	
LATD	0XF8C	0b01000001	
STATUS	0xFD8	0b00000000	
PCL	0XE9	0x24	
PCLATH	0xFFA	0b00000000	
PCLATU	0xFFB	0b00000000	

9

¿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



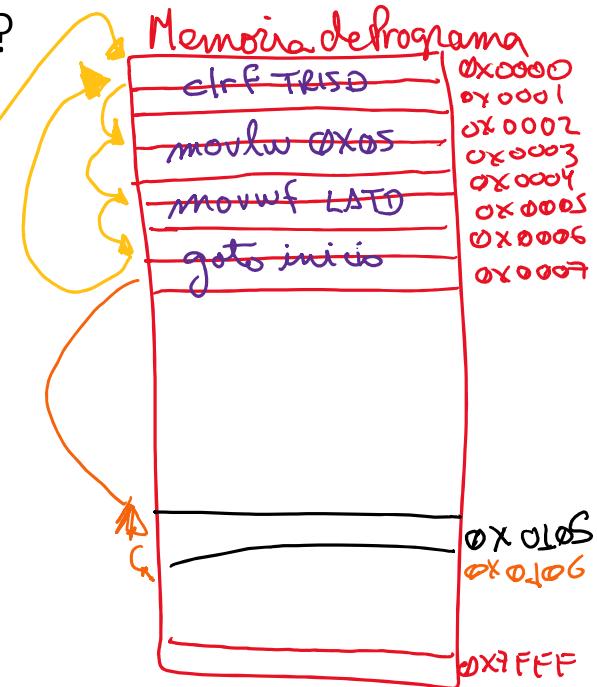
10

¿Cómo funciona el PC?

- Se tiene el siguiente programa

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

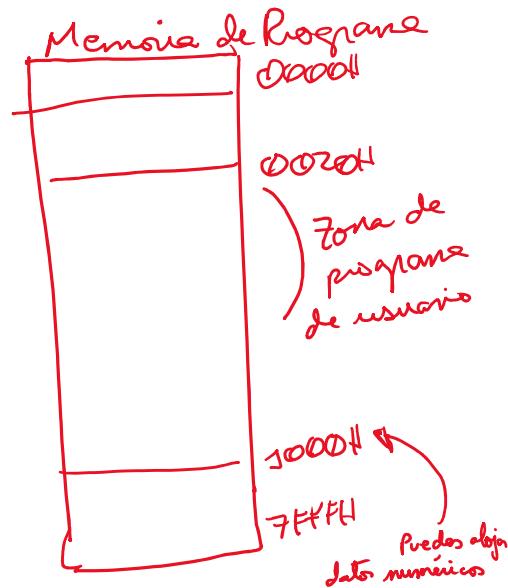
Nota: Las instrucciones en MPASM ocupan 2 bytes
 Nota: Se comprueba que PC va de dos en dos, no puede contener una dirección impar



11

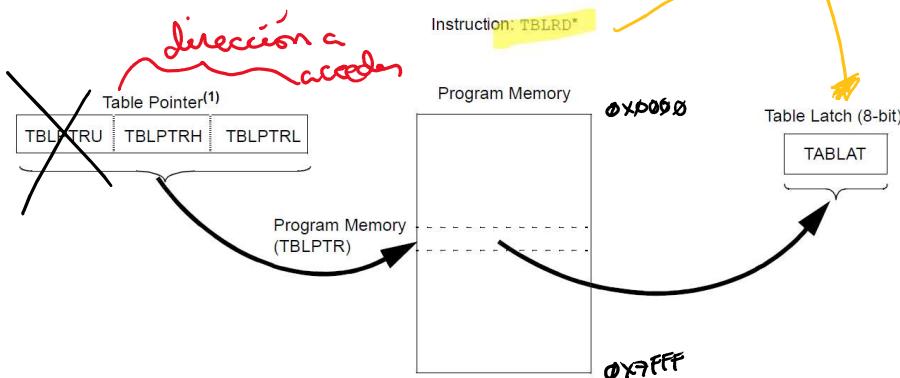
Comentarios:

- 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)



12

El puntero de tabla (TBLPTR)

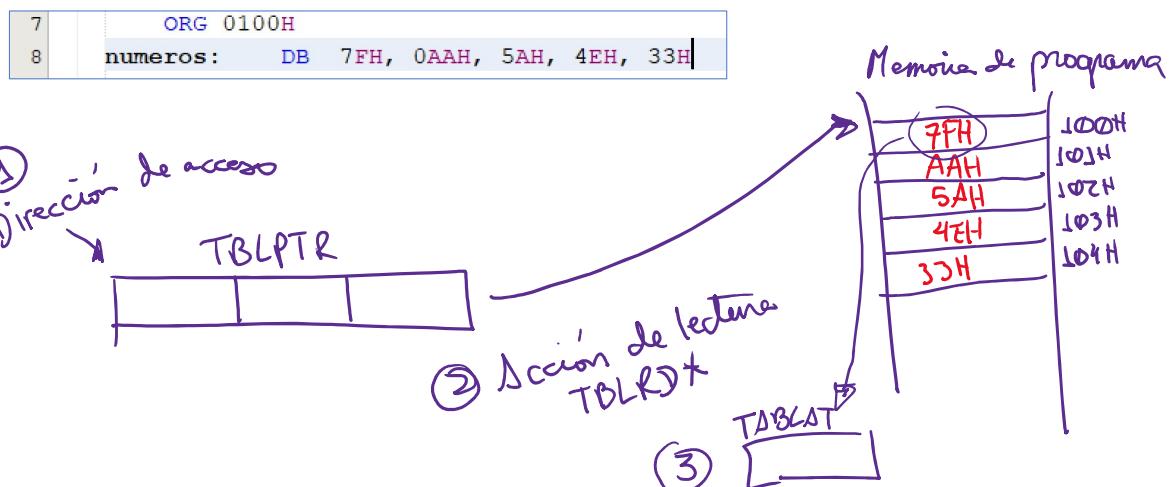


Note 1: Table Pointer register points to a byte in program memory.

- Al igual que el PC, el TBLPTR también es de 21 bits (para el caso del PIC18F45K50 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

13

Operación con el TBLPTR



14

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 configuro
12         ORG 0020H
13
14         configuro:
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 modificado a 000102H
25             TBLRD*              ;Leo contenido apuntado por TBLPTR
26             movf TABLAT, 0        ;Muevo el contenido de TABLAT hacia WReg
27             movwf LATD            ;Muevo contenido de WReg hacia LATD
28             goto loop
29         end principal

```

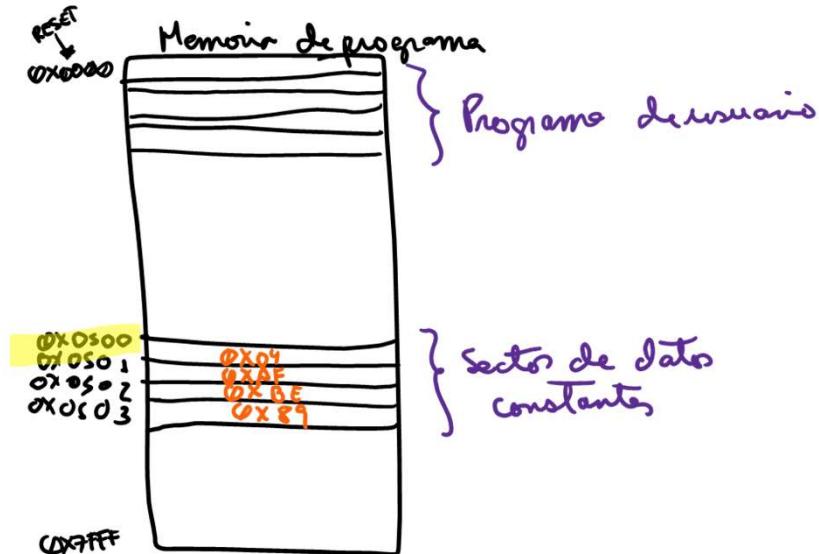
15

Ejemplo:

- Desarrollar un programa donde se tenga almacenado los siguientes datos en la **memoria de programa**:
 - 0x00500: 0x04
 - 0x00501: 0xAF
 - 0x00502: 0xBE
 - 0x00503: 0x89
- Elaborar un algoritmo que permita leer los datos anteriores y arrojarlas de manera secuencial a través de RD con periodo de NOP

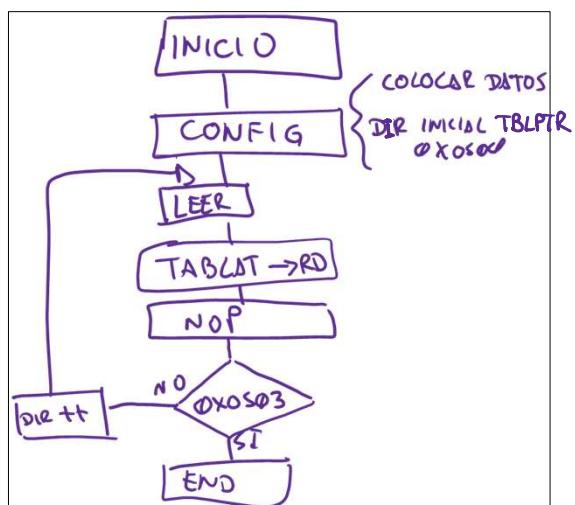
16

Desarrollo del ejemplo:



17

Diagrama de flujo y código del ejemplo (MPASM):



```

2      list p=18f4550
3      #include <p18f4550.inc>           ;libreria de nombre de los registros sfr
4
5      CONFIG FOSC = XT_XT             ; Oscillator Selection bits (XT oscillator (XT))
6      CONFIG PWRT = ON               ; Power-up Timer Enable bit (PWRT enabled)
7      CONFIG BOR = OFF               ; Brown-out Reset Enable bit (Brown-out Reset)
8      CONFIG WDT = OFF               ; Watchdog Timer Enable bit (WDT disabled (on))
9      CONFIG PBADEN = OFF            ; PORTB A/D Enable bit (PORTB<4:0> pins are config)
10     CONFIG IVP = OFF               ; Single-Supply ICSP Enable bit (Single-Supply
11
12     org 0x0500                   ;Sector de almacenamiento de constantes
13     numeros db 0x04, 0xAF, 0xBE, 0x89
14
15     org 0x0000                   ;Vector de reset
16     goto configuracion
17
18     org 0x0020                   ;Zona de programa de usuario
19     configuracion:
20     clrf TRISD                 ;Todo RD como salida
21     movwf TABLPRH
22     movwf TABLTRL
23     movlw LOW numeros
24     movwf TABLTRL                ;Carga de la dirección de apunte de TBLPTR (0x0500)
25
26     inicio:
27     TABLRD*
28     movwf TABLAT, LATD
29     nop
30     movlw 0x03
31     cpfseq TABLTRL
32     goto falso
33     verdadero:
34     nop
35     goto verdadero
36
37     falso:
38     incf TABLTRL, f
39     goto inicio
40
41     end

```

18

PIC18 CPU Source Code - U1

D:\Microcontroladores\20202_tb1ptr2.X\maincode2.asm

```

----- CONFIG_WDT = OFF ; Watchd ~
----- CONFIG_CCP2MX = ON ; CCP2_M
----- CONFIG_PBADEN = OFF ; PORTB
----- CONFIG_MCLRE = ON ; MCLR_P
----- CONFIG_LVP = OFF ; Single

----- cuenta EQU 0x020 ; La posicion 0x100
----- org 0x0000 goto init_conf ; Vector de RESET
----- org 0x0500 datos db 0x04, 0xAF, 0xBE, 0x89
----- org 0x0020 ; Zona de programa c
----- init_conf:
0020    clrf TRISD ; RD como salida
0022    movlw HIGH datos
0024    movwf TBLPTRH ; Dirección del TBLF
0026    movlw LOW datos
0028    movwf TBLPTRL ; Dirección del TBLF
----- loop:
002A    TBLRD* ; Acción de lectura
002C    movff TABLAT, LATD ; Muevo el contenido
0030    nop ; Microretardo
0032    incf TBLPTRL,f ; Incremento la posición
0034    TBLRD* ; Acción de lectura
0036    movff TABLAT, LATD
003A    nop
003C    incf TBLPTRL,f
003E    TBLRD* ; Acción de lectura
0040    movff TABLAT, LATD
0044    nop
0046    incf TBLPTRL,f
0048    TBLRD* ; Acción de lectura
004A    movff TABLAT, LATD
end

```

RC0/T1OSO/T1CKI ■ 15
I/T1OSI/CCP2/UOE ■ 16
RC2/CCP1/P1A ■ 17
RC4/D/V/M ■ 23
RC5/D+V/P ■ 24
RC6/TX/CK ■ 25
RC7/RX/DT/SDO ■ 26

RD0/SPP0 ■ 19
RD1/SPP1 ■ 20
RD2/SPP2 ■ 21
RD3/SPP3 ■ 22
RD4/SPP4 ■ 23
RD5/SPP5/P1B ■ 24
RD6/SPP6/P1C ■ 25
RD7/SPP7/P1D ■ 26

RE0/AN5/CK1SPP ■ 8
RE1/AN6/CK2SPP ■ 9
RE2/AN7/QESPP ■ 10
RE3/MCLR/VPP ■ 11

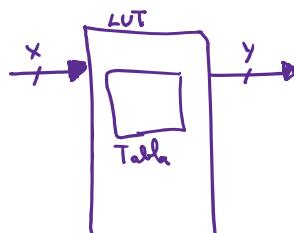
Watch Window

Name	Address	Value
TRISD	0x0F95	0b00000000
PORTD	0x0F83	0b00000000
LATD	0x0F8C	0x89
STATUS	0x0FD8	0b00000000
PCLATH	0x0FFB	0b00000000
PCL	0xFF9	0x22
TBLPTRU	0xFF8	0b00000000
TBLPTRH	0xFF7	0x05
TBLPTRL	0xFF6	0x03
TABLAT	0xFF5	0x89
cuenta	0x0020	0x00
WREG	0xFE8	0x00

19

Tablas de búsqueda (lookup tables - LUT)

- Decodificadores implementados en software

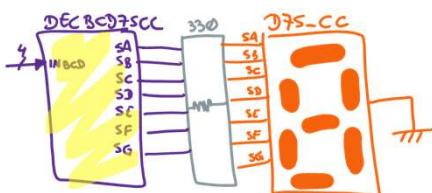


- Dos maneras de implementar LUT en MPASM-PIC18
 - Utilizando la funcionalidad del PC
 - Utilizando el TBLPTR

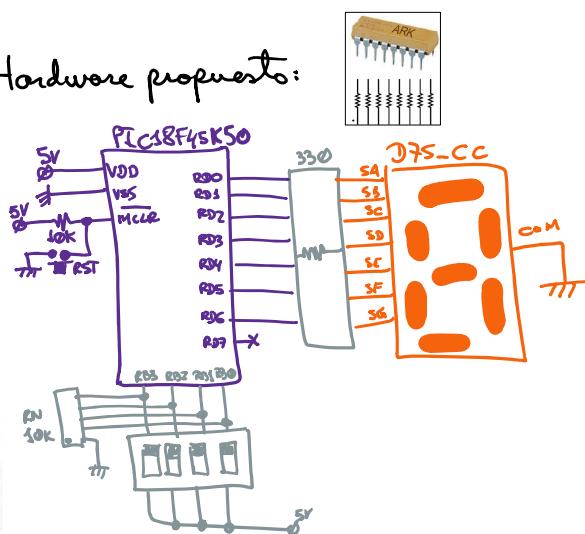
20

Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:



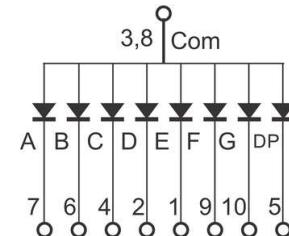
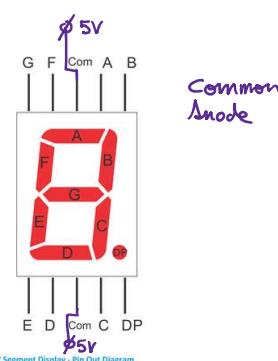
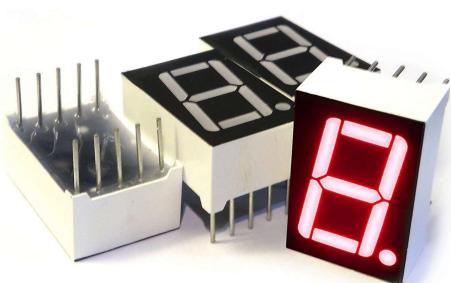
Hardware propuesto:



21

Observación:

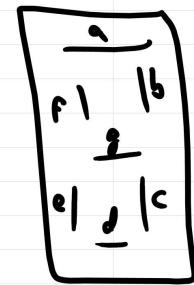
- Los pines denominados “comunes” (Com) son el mismo nodo, la misma conexión!



22

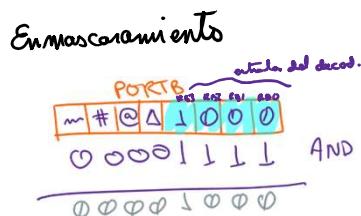
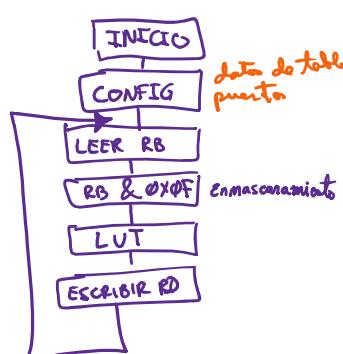
Desarrollo de la tabla de decodificación para display de 7 segmentos cátodo común:

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



23

Diagrama de flujo



Código previo (empleando PC como LUT)

```

31    loop:
32        movf PORTB, w
33        andlw 0x0F
34        movwf valor_entrada
35        call tabla_pc
36        movwf LATD
37        goto loop

38    tabla_pc:
39        movf valor_entrada, w
40        addwf PCL, f
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
  
```

Diagram illustrating the PC as a LUT. The PC is used to index into a table of addresses (0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F). The PC is modified by adding the current value of W (which is the masked input) to it twice. The resulting address is then used to jump to the corresponding table entry.

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!

24

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

```

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

25

Modificación del decodificador empleando TBLPTR

- 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. (Código en MPASM)

```

2     list p=18f4550      ;Modelo del microcontrolador
3     #include <p18f4550.inc>      ;Llamada a la librería de nombre de los registros
4
5     ;Directivas de preprocesador o bits de configuración
6     CONFIG PLLDIV = 1          ; PLL Prescaler Selection bits (No prescale (4 MHz oscillator input drive)
7     CONFIG CPUDIV = OSC1_PLL2  ; System Clock Postscaler Selection bits ([Primary Oscillator Src: 1]/[9]
8     CONFIG FOSC = XT_XT       ; Oscillator Selection bits (XT oscillator (XT))
9     CONFIG FWRT = ON          ; Power-up Timer Enable bit (PWRT enabled)
10    CONFIG BOR = OFF          ; Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and
11    CONFIG WDT = OFF          ; Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDT)
12    CONFIG CCP2MX = ON         ; CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
13    CONFIG PBADEN = OFF        ; PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on
14    CONFIG MCLE = ON          ; MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)
15    CONFIG LVP = OFF          ; Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
16
17    org 0x0500
18    tabla_7s db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x79, 0x79, 0x79, 0x79, 0x79
19
20    org 0x0000
21    goto init_conf
22
23    ;Aqui se pueden declarar las constantes en la memoria de programa
24
25    org 0x0020
26    init_conf:
27        clrf TRISD      ;Todo RD como salida
28        movlw high tabla_7s
29        movwf TBLPTRH
30        movlw low tabla_7s
31        movwf TBLPTRL   ;TBLPTR apuntando a 0x0500

```

```

33    loop:
34        movf PORTB, w
35        andlw 0x0F
36        movwf TBLPTRL
37        TBLRD*
38        ;        comf TABLAT, w
39        ;        movwf LATD
40        movff TABLAT, LATD
41        goto loop
42
43        end

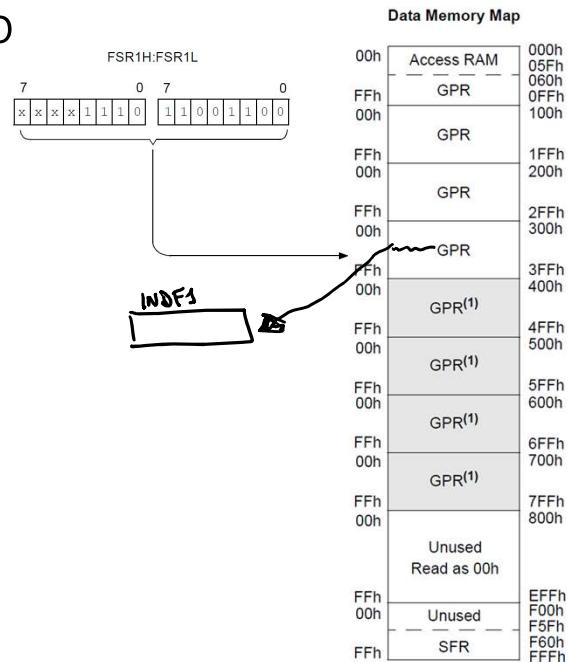
```

Nota: Si tienes display de ánodo común deberás de activar las líneas 38 y 39, y comentar la línea 40 del presente código

26

Memoria de datos: Acceso con punteros FSRx/INDFx

- En la memoria de datos se encuentra mapeado los 2Kbyte de RAM (0x000 – 0x7FF) y los S.F.R. (0xF60 – 0xFFFF)
- Se tienen tres punteros:
 - FSR0 / INDF0
 - FSR1 / INDF1
 - FSR2 / INDF2
- Al igual que TBLPTR, en FSRx se coloca la dirección de la celda a apuntar y en IDFx se opera su contenido (lectura o escritura)



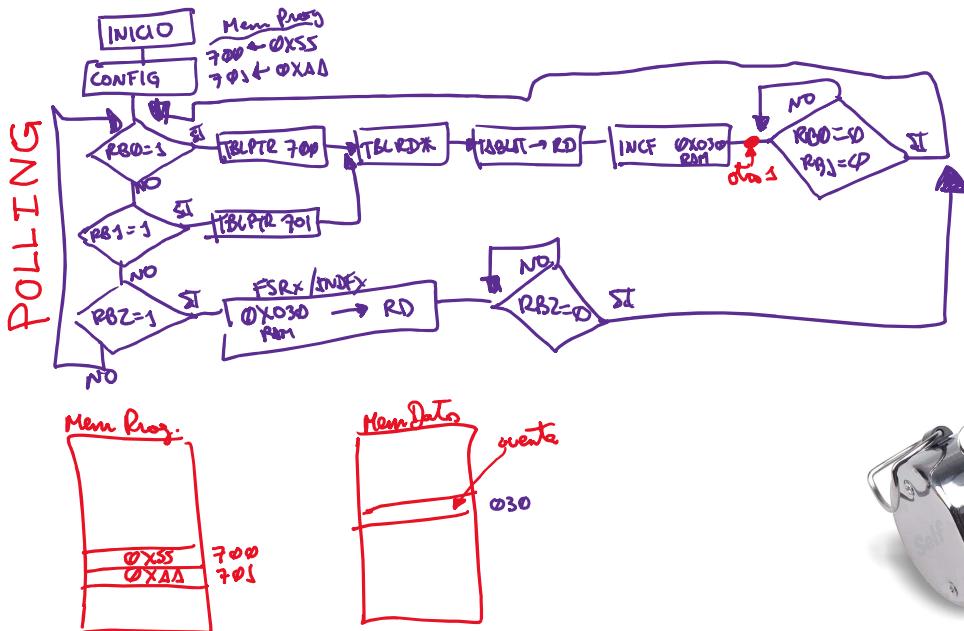
27

Ejercicio:

- Un dato 55H estará almacenado en la dirección 0700H y otro dato 0AAH en la dirección 0701H en la **memoria de programa**.
- Se tiene dos pulsadores en RB0 y RB1 respectivamente, si se pulsa el botón en RB0 se hará un proceso de lectura de la **memoria de programa** en la posición 0700H y el contenido leído será enviado a RD donde se tendrán 8 LEDs entre sus puertos. Si se pulsa el botón RB1 hará lo mismo que el otro botón pero leyendo el contenido de la dirección 0701H de la memoria de programa.
- Cada pulsación que se haga deberá de contarse y registrarse esa cuenta en la dirección 030H de la **memoria de datos**.
- Adicionalmente se contará con un botón en RB2 el cual al ser accionado leerá el contenido de la dirección 030H de la memoria de datos y lo arrojará por el RD.

28

Diagrama de flujo



29

Código en MPASM

```

1 ;Este es un comentario, se le antecede un punto y coma
2 list p=18f4550 ;Modelo del microcontrolador
3 #include <p18f4550.inc> ;Llamada a la biblioteca
4
5 ;Directivas de preprocesador o bits de configuración
6 CONFIG PLDIV = 1 ; PLL Prescaler
7 CONFIG CPUDIV = OSC1_PLL2 ; System Clock
8 CONFIG FOSC = XT_XT ; Oscillator Type
9 CONFIG PWRT = ON ; Power-up
10 CONFIG BOR = OFF ; Brown-out
11 CONFIG WDT = OFF ; Watchdog
12 CONFIG CCP2MX = ON ; CCP2 MUX
13 CONFIG PBADEN = OFF ; PORTB A/D
14 CONFIG MCLE = ON ; MCLR Pin
15 CONFIG LVP = OFF ; Single-Supply
16
17 cuenta EQU 0x030 ;La posición 0x030 de memoria
18
19 org 0x0000 ;Vector de RESET
20 goto init_conf
21
22 org 0x0700
23 datos db 0x55, 0xAA
24
25 org 0x0020 ;Zona de programa de usuario
26
27 init_conf:
28     clrf TRISD ;Por acá saldrán los datos que
29     ;Vamos a colocarle la dirección inicial de TBLPTR
30     movlw HIGH datos
31     movwf TBLPTRH
32     movlw LOW datos
33     movwf TBLPTRL ;Aquí el punto YA SE ENCUENTRA
34     clrf cuenta ;Forzamos al registro GPR que
35
36         loop:
37             btfss PORTB, 0 ;Preguntamos si se presionó botón0
38             goto next1
39             goto accion1
40
41 next1:
42     btfss PORTB, 1 ;Preguntamos si se presionó botón1
43     goto next2
44     goto accion2
45
46 next2:
47     btfss PORTB, 2 ;Preguntamos si se presionó botón2
48     goto loop
49     goto accion3
50
51 accion1:
52     movlw 0x00
53     movwf TBLPTRL ;Para acceder a 0x0700 de la mem prog
54     TBLRD*
55     movff TABLAT, LATD
56     incf cuenta, f ;Incremento la cuenta y se almacena en el registro
57
58     otro1:
59         btfsc PORTB, 0
60         goto otro1
61         goto loop
62
63 accion2:
64     movlw 0x01
65     movwf TBLPTRL ;Para acceder a 0x0700 de la mem prog
66     TBLRD*
67     movff TABLAT, LATD
68     incf cuenta, f ;Incremento la cuenta y se almacena en el registro
69
70     otro2:
71         btfsc PORTB, 1
72         goto otro2
73         goto loop
74
75 accion3:
76     lfsr 0, 0x030 ;Asignación una dirección de memoria de datos a FSR0
77     movff INDF0, LATD
78
79 otro3:
80     btfsc PORTB, 2
81     goto otro3
82     goto loop
83     end

```

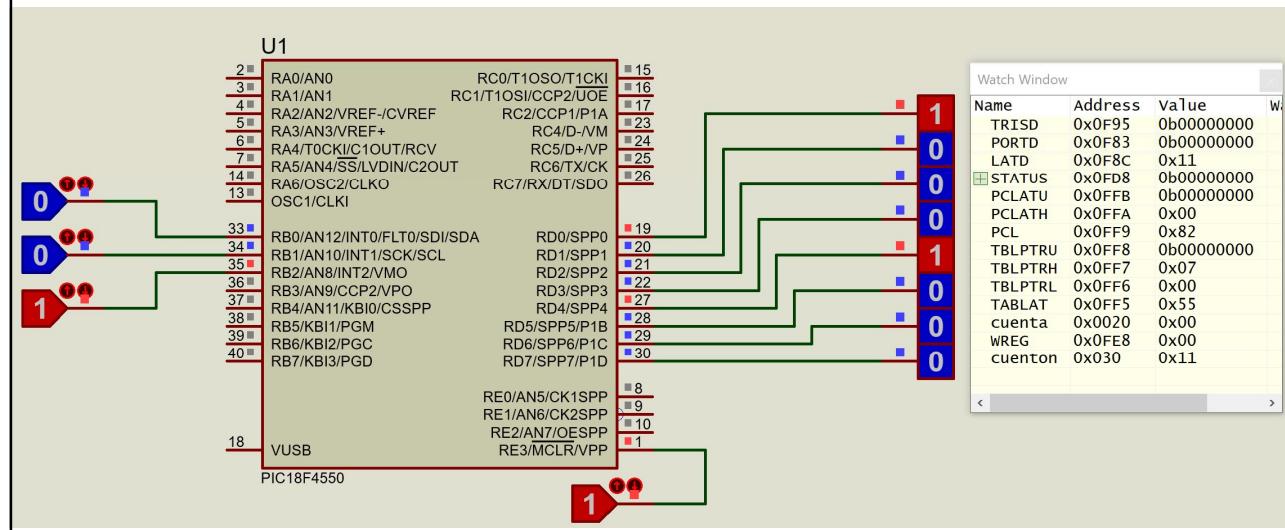
30

Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler y orientarlo al PIC18F45K50

31

Simulación:



32

Ejercicio:

- Se tienen los siguientes datos que deberán encontrarse en la dirección de memoria de programa 0x0421:

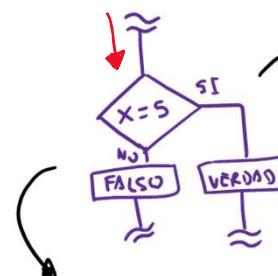
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A
- Dichos números corresponden a letras en mayúscula según tabla de caracteres ASCII (ver tabla de la derecha)
- Desarrollar un programa (previamente el diagrama de flujo) para obtener en binario las letras de tu nombre a través del puerto RB en forma secuencial y una letra a la vez.
- Desarrollar algoritmo y programa para almacenar dicho nombre en la memoria de datos desde la dirección 0x030, luego arrojar de manera secuencial el nombre al revés a través del puerto RB y de manera secuencial una letra a la vez

Dec	Hx	Oct	Html	Ch
64	40	100	@	Ø
65	41	101	A	A
66	42	102	B	B
67	43	103	C	C
68	44	104	D	D
69	45	105	E	E
70	46	106	F	F
71	47	107	G	G
72	48	108	H	H
73	49	109	I	I
74	4A	110	J	J
75	4B	111	K	K
76	4C	112	L	L
77	4D	113	M	M
78	4E	114	N	N
79	4F	115	O	O
80	50	120	P	P
81	51	121	Q	Q
82	52	122	R	R
83	53	123	S	S
84	54	124	T	T
85	55	125	U	U
86	56	126	V	V
87	57	127	W	W
88	58	130	X	X
89	59	131	Y	Y
90	5A	132	Z	Z

33

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

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



En MPASM:
 Usaremos CPFSEQ:
 CPFSEQ [reg]

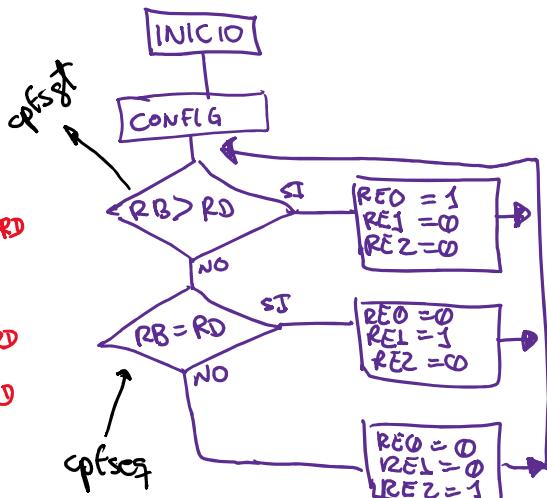
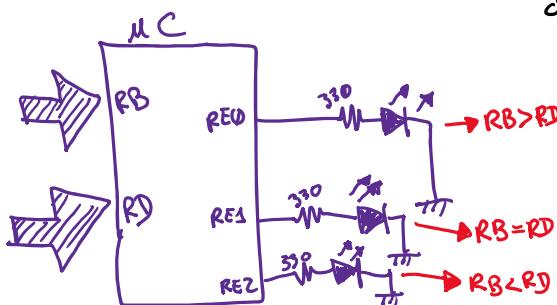
En lenguaje de alto nivel:
 if (x = 5){
 [VERDAD]}
 else {
 [Falso]}

(f) = w
 x = 5
 movlw .5
 cpfseq x-reg
 goto Falso
 goto verdad

34

Ejemplo:

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



35

Código MPASM:

```

1 ;Este es un comentario, se le asigna
2      list p=18f4550          ;Modelo
3      #include <p18f4550.inc>
4
5      ;Directivas de preprocessado:
6      CONFIG PLLDIV = 1
7      CONFIG CPUDIV = OSC1_PLL2
8      CONFIG FOSC = XT_XT
9      CONFIG FWRT = ON
10     CONFIG BOR = OFF
11     CONFIG WDT = OFF
12     CONFIG CCP2MX = ON
13     CONFIG PBADEN = OFF
14     CONFIG MCLRE = ON
15     CONFIG LVP = OFF
16
17     org 0x0000
18     goto init_conf
19
20     org 0x0020
21     init_conf:
22     movlw 0x08
23     movwf TRISE
24     ;Aqui falta algo que no me acuerdo
25
26     loop:
27     movf PORTD, W
28     cpfsgt PORTB
29     goto next1
30     bsf LATE, 0
31     bcf LATE, 1
32     bcf LATE, 2
33     goto loop
34
35     next1:
36     movf PORTD, W
37     cpfseq PORTB
38     goto next2
39     bcf LATE, 0
40     bsf LATE, 1
41     bcf LATE, 2
42     goto loop
43
44     next2:
45     bcf LATE, 0
46     bcf LATE, 1
47     bsf LATE, 2
48     goto loop
49
50     end

```

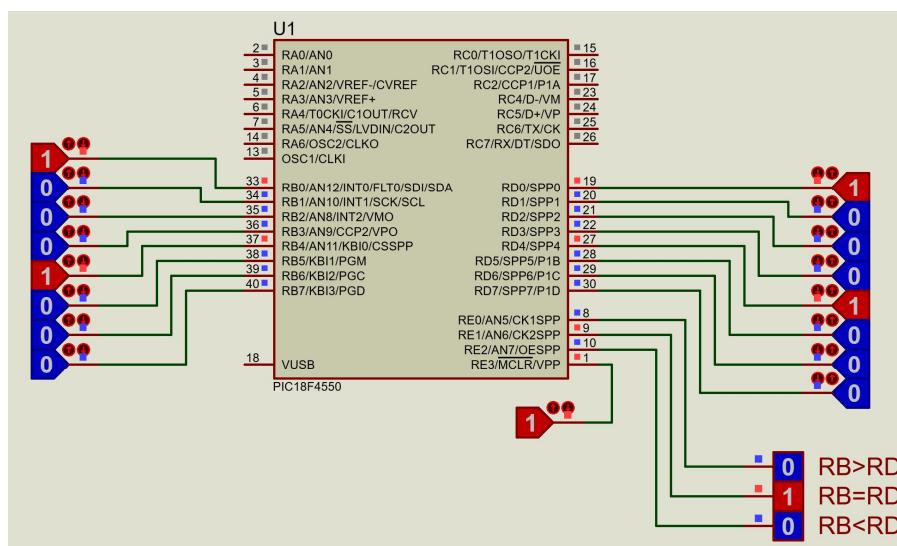
36

Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler y orientarlo al nuevo PIC18F45K50

37

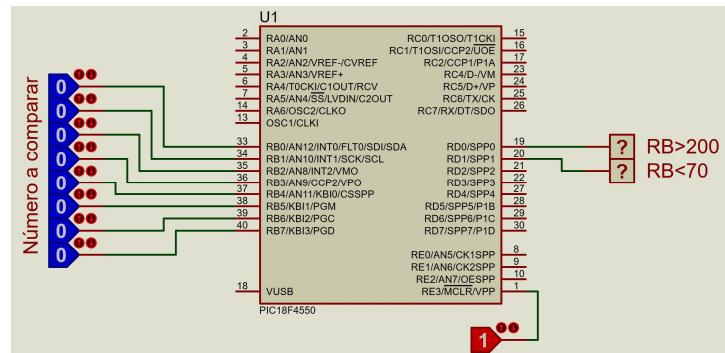
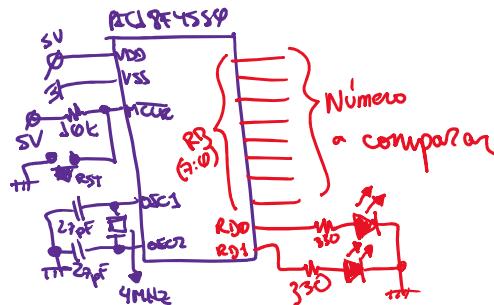
Simulación



38

Ejemplo:

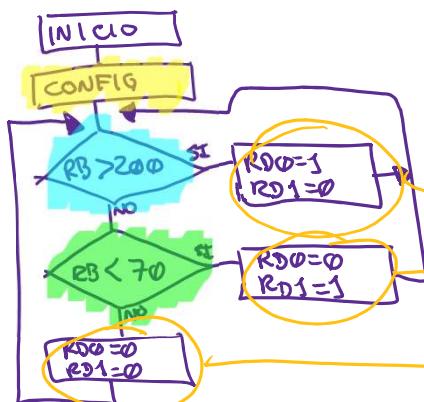
- Desarrollar un programa para que compare lo que se está ingresando en RB y arroje lo siguiente: RD0=1 cuando RB>200 y RD1=1 cuando RB<70, cuando no se cumplan las dos condiciones las dos salidas permanecerán en cero.



39

Cont.

Diagrama de flujo:



```

1 ;Este es un comentario, se le :
2 list p=18f4550           ;Modelo
3 #include <p18f4550.inc>
4
5 ;Directivas de preprocesador
6 CONFIG PLLDIV = 1
7 CONFIG CPUDIV = OSC1_PLL2
8 CONFIG FOSC = XT_XT
9 CONFIG FWRT = ON
10 CONFIG BOR = OFF
11 CONFIG WDT = OFF
12 CONFIG CCP2MX = ON
13 CONFIG PBADEN = OFF
14 CONFIG MCLRE = ON
15 CONFIG LVP = OFF
16
17 org 0x0000
18 goto init_conf
19
20 org 0x0020
21 init_conf:
22     movlw 0xFC
23     movwf TRISD
24
25 loop:
26     movlw .200
27     cpfsgt PORTB
28     goto next1
29     bcf LATD, 0
30     bcf LATD, 1
31     goto loop
32 next1:
33     movlw .70
34     cpfslt PORTB
35     goto next2
36     bcf LATD, 0
37     bcf LATD, 1
38     goto loop
39 next2:
40     bcf LATD, 0
41     bcf LATD, 1
42     goto loop
43 end

```

40

Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler y orientarlo al nuevo PIC18F45K50

41

Fin de la sesión

42