

# Microcontroladores

## Semana 3

Semestre 2022-1

Profesor: Kalun José Lau Gan

1

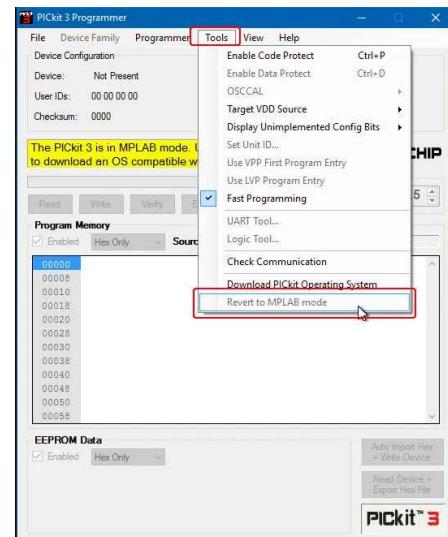
### Preguntas previas:

- Sigo teniendo problemas en la implementación de los ejemplos vistos en clases anteriores
  - Verificar que el multímetro 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
    - Cerrar el MPLABX y abrirlo de nuevo
    - Desconectar el PICKIT3 de la PC y volverlo a conectar
  - Me sale 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:

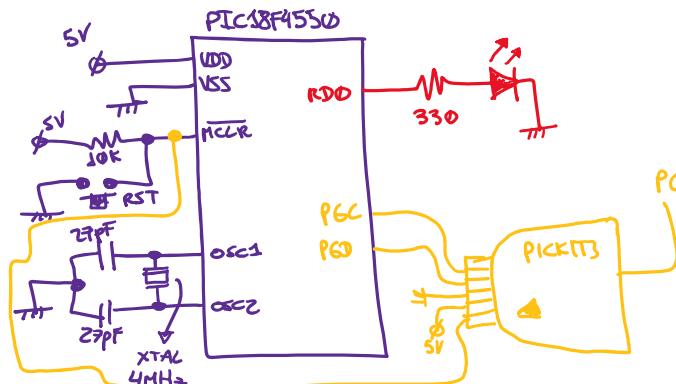
- El PICKIT3 al conectarlo a la PC el software MPLABX no lo reconoce.
  - Leer la consulta siguiente.
- PICKIT 3 lo operaba antes con el software stand-alone y no funciona con el MPLABX. ¿Qué hay que hacer?
  - Entrar a la aplicación standalone de pickit3 y cambiar el firmware para que opere en modo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



3

## Preguntas previas:

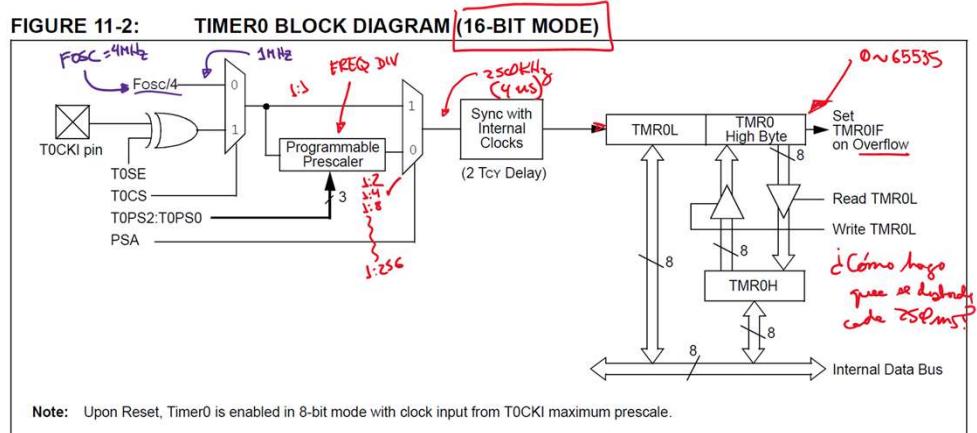
- ¿Cómo puedo saber si el cristal de 4MHz solicitado esta funcionando o no?
- Implementaremos un titilador de LED conectado en RD0 y utilizando el modulo Timer0 como fuente de reloj



4

- ¿Cómo puedo saber si el cristal de 4MHz solicitado esta funcionando o no?

• Luego hacemos el cálculo para que el Timer0 temporice aproximadamente 250ms al desborde, eso se consigue configurándolo en modo 16bit y con prescaler 1:4. Dicho desborde será una señal para que alterne el valor lógico del puerto donde esté conectado el LED



5

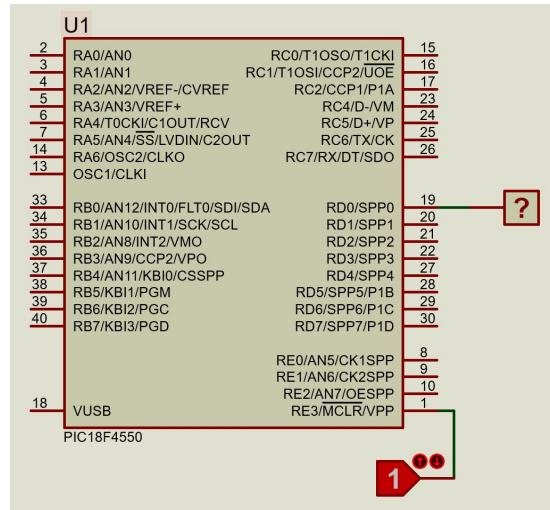
- ¿Cómo puedo saber si el cristal de 4MHz solicitado esta funcionando o no?

• Obtenemos el valor para el registro del Timer0 llamado TOCON según lo analizado en la vista anterior

REGISTER 11-1: TOCON: TIMER0 CONTROL REGISTER							
R/W-1	R/W-1	R/W-1	R/W	R/W-1	R/W-1	R/W-1	R/W-1
TMROON	T0SBIT	TMCS	TOSE	PSA	TOPS2	TOPS1	TOPS0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
<b>Legend:</b>							
R = Readable bit -n = Value at POR	W = Writable bit '1' = Bit is set	U = Unimplemented bit, read as '0' '0' = Bit is cleared	x = Bit is unknown				
bit 7: TMROON: Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0							
bit 6: T08BIT: Timer0 8-bit/16-Bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter							
bit 5: TMCS: Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKO)							
bit 4: TOSE: Timer0 Source Edge Select bit 1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin							
bit 3: PSA: Timer0 Prescaler Assignment bit 1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.							
bit 2-0: TOPS2:TOPS0: Timer0 Prescaler Select bits 111 = 1:256 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value							

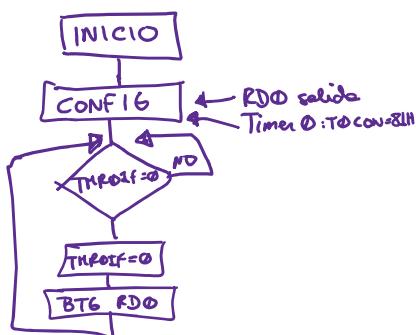
6

- ¿Cómo puedo saber si el cristal de 4MHz solicitado esta funcionando o no?
  - Hacemos el circuito en Proteus para hacer simulación



7

- ¿Cómo puedo saber si el cristal de 4MHz solicitado esta funcionando o no?
  - Código fuente en XC8 PIC Assembler



```

1  PROCESSOR 18F4550
2  #include "cabecera.inc"
3
4  PSECT principal, class=CODE, reloc=2, abs
5
6  principal:
7    ORG 0000H
8    goto configuro
9    ORG 0020H
10
11 configuro:
12    bcf TRISD, 0      ;RD0 como salida
13    movlw 81H
14    movwf T0CON        ;Timer0 activado, 16bit, PSC1:4, FOSC/4
15
16 loop:
17    btfss INTCON, 2
18    goto loop
19    bcf INTCON, 2
20    btg LATD, 0
21    goto loop
22
23    end principal
  
```

8

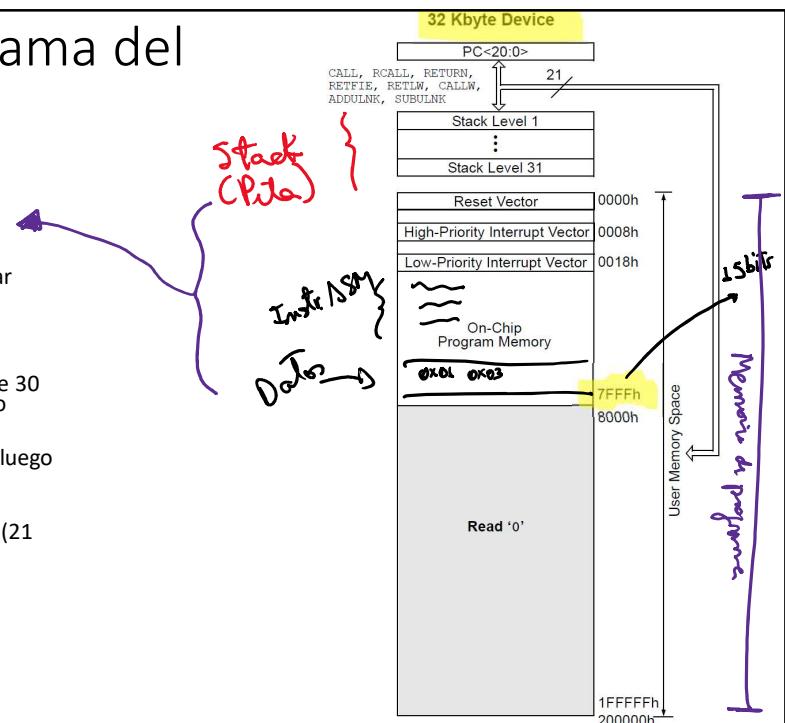
## Agenda:

- Memoria de programa del microcontrolador PIC18F4550
- El contador de programa (PC) del CPU del microcontrolador PIC18F4550
- Acceso a datos almacenados en la memoria de programa empleando el puntero del tabla (TBLPTR)
- Memoria de datos del microcontrolador PIC18F4550: acceso mediante punteros FSRx/INDFx
- Interface a display de siete segmentos
- Instrucciones CPFSEQ, CPFSLT, CPFSGT en MPASM

9

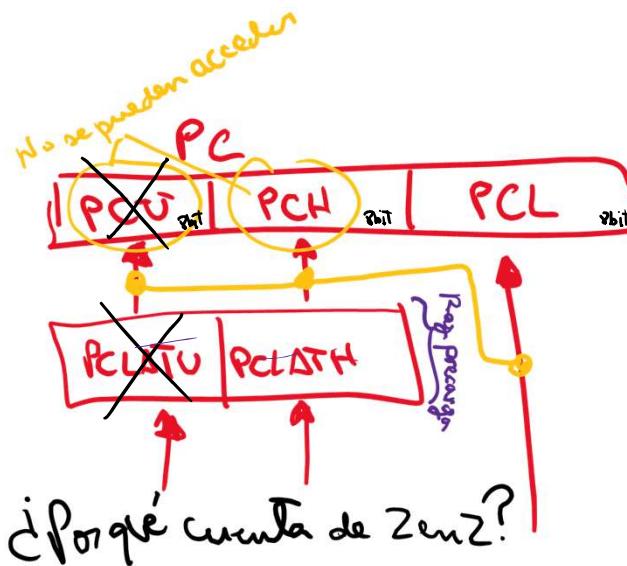
## Memoria de programa del PIC18F4550

- Referencia: Item 5, página 59 de hoja técnica rev. E.
- Tamaño: 32KByte (0x0000 a 0xFFFF).
- Instrucciones ocupan 2bytes, en casos especiales ocupan hasta 4bytes (revisar Item 16 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)



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.
- Al escribir en PCL se subirán PCLATH y PCLATU para así subir los 21 bits a la vez
- En el PIC18F4550 no está implementado PCU debido al tamaño de la memoria de programa (32KB ó 15 bits de direccionamiento).

11

## Ejemplo

- Cargar dirección 0x00288A en PC:

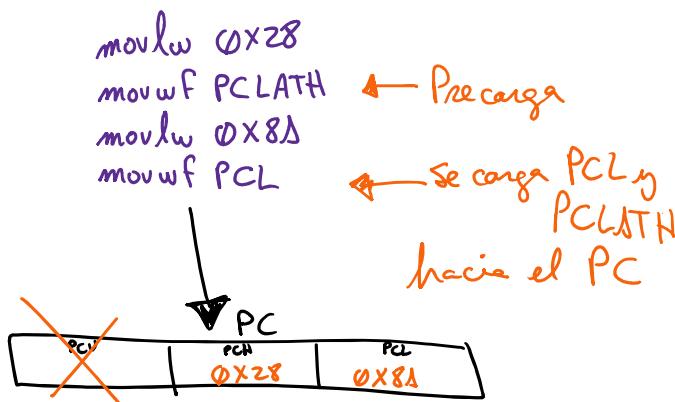


TABLE 5-1: SPECIAL FUNCTION REGISTER MAP							
Address	Name	Address	Name	Address	Name	Address	Name
FF9h	TOSU	FDFh	INDF2 <sup>(1)</sup>	FB8h	CCPR1H		
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FB8h	CCPR1L		
FFDh	TOSL	FDCh	POSTDEC2 <sup>(1)</sup>	FB8h	CCPICON		
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FB8h	CCPR2H		
FFBh	PCLATH	FDBh	PLUSW2 <sup>(1)</sup>	FB8h	CCPR2L		
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON		
FFBh	PCL	FD8h	FSR2L	FB8h	— <sup>(2)</sup>		
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON		
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL		
FF6h	TBLPTRL	FD6h	TMR0L	FB8h	ECCP1IAS		
FE5h	TARLAT	FD5h	TOCON	FB8h	CVRCON		

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

```

↓ hace un salto a 0x0020

↓ haz un salto a 0x0022

↓ modifica el PC

*Direcciones de la memoria de programa*

13

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             ; PORTB A/D Enable bit (PORTB<4:0>.pi
----- CONFIG MCLRE = OFF
----- 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	0x0F95	0b00000000	
TRISB	0x0F93	0b11111111	
PORTB	0x0F81	0b10111110	
PORTD	0x0F83	0b00000000	
LATB	0x0F8A	0b00000000	
LATD	0x0F8C	0b01000001	
STATUS	0x0FD8	0b00000000	
PCL	0x0FF9	0x24	
PCLATH	0x0FFA	0b00000000	
PCLATU	0x0FFB	0b00000000	

14

## ¿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



15

## ¿Cómo funciona el PC?

- Se tiene el siguiente programa

```

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

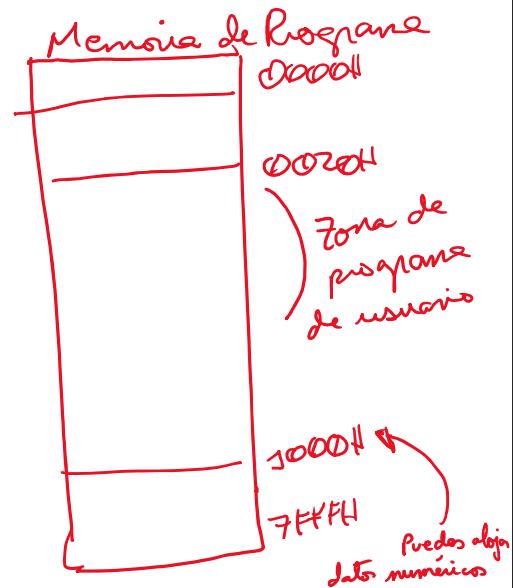
Nota: Las instrucciones en MPASM ocupan 2 bytes  
Nota: Se comprueba que PC va de dos en dos, no puede contener una dirección impar



16

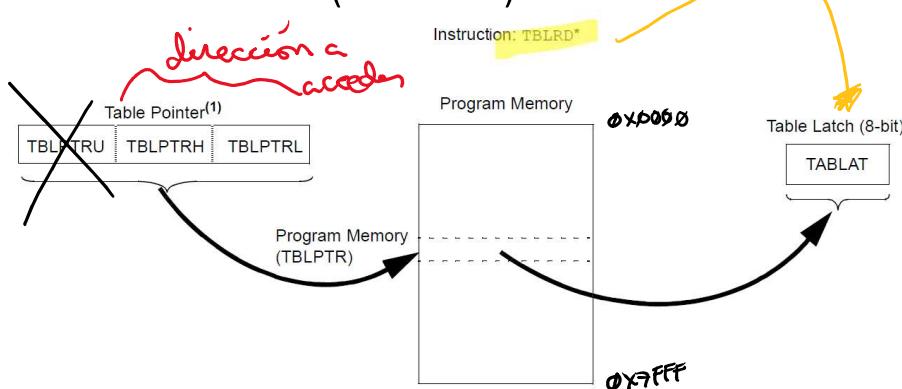
## Comentarios:

- 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 esté en operación)



17

## El puntero de tabla (TBLPTR)

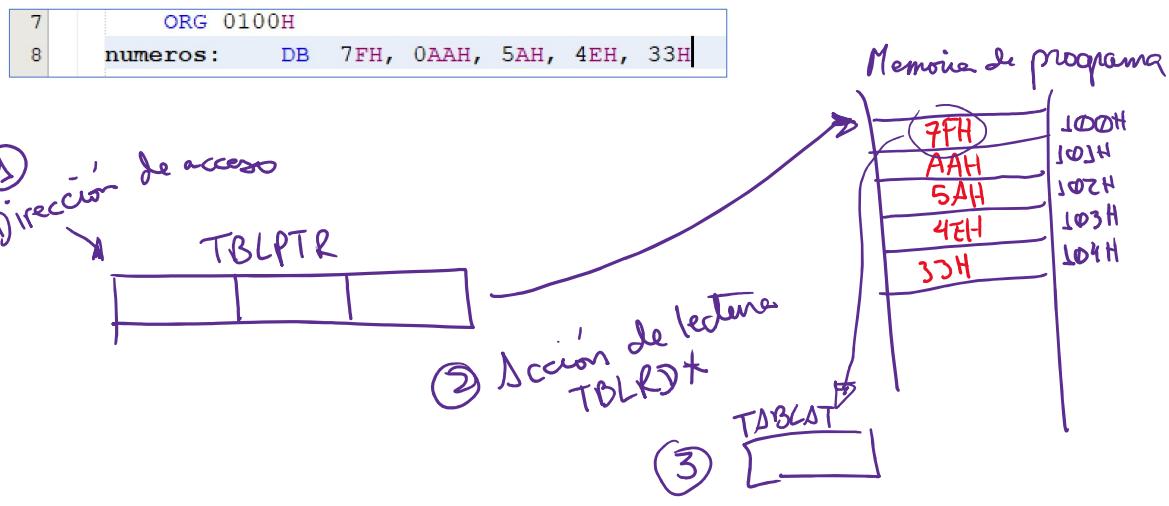


Note 1: Table Pointer register points to a byte in program memory.

- 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 más 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

18

## Operación con el TBLPTR



19

## 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 modificada 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

```

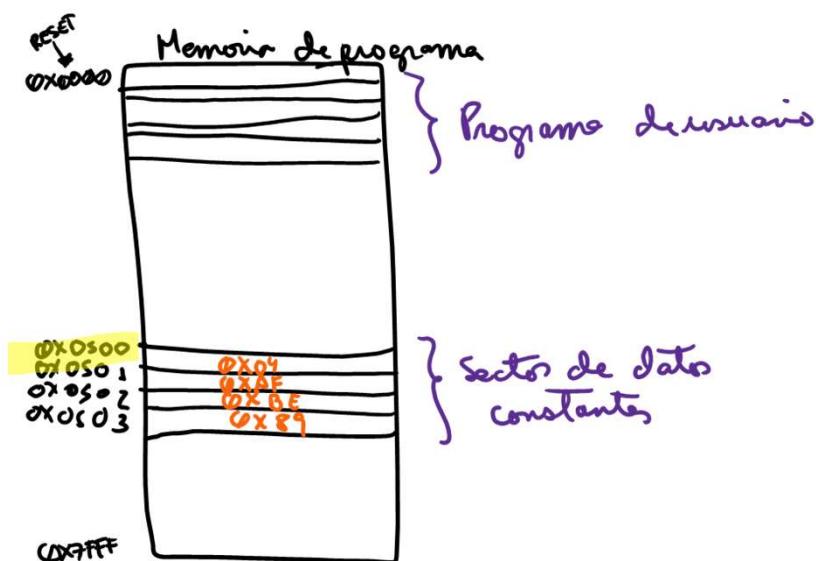
20

## 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

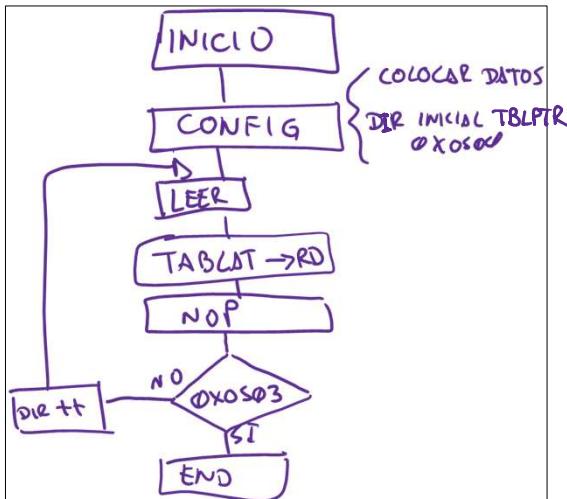
21

## Desarrollo del ejemplo:



22

## Diagrama de flujo y código del ejemplo (MPASM):



```

list p=18f4550
#include <18f4550.inc> ;libreria de nombre de los registros sfr

CONFIG FOSC = XT_XT ; Oscillator Selection bits (XT oscillator (X))
CONFIG PWRT = ON ; Power-up Timer Enable bit (PWRT enabled)
CONFIG BOR = OFF ; Brown-out Reset Enable bits (Brown-out Reset)
CONFIG WDT = OFF ; Watchdog Timer Enable bit (WDT disabled (con
CONFIG PBADEN = OFF ; PORTB A/D Enable bit (PORTB<4:0> pins are co
CONFIG LVP = OFF ; Single-Supply ICSP Enable bit (Single-Supply

org 0x0500 ;Sector de almacenamiento de constantes
numeros db 0x04, 0xAF, 0xBE, 0x89

org 0x0000 ;Vector de reset
goto configuro

org 0x0020 ;Zona de programa de usuario
configuro:
  clrf TRISD ;Todo RD como salida
  movwf TBLPTRH
  movlw HIGH numeros
  movwf TBLPTRL
  movlw LOW numeros
  movwf TBLPTRL ;Carga de la dirección de apunte de TBLPTR (0x0500)

inicio:
  TBLRD*
  movff TABLAT, LATD
  nop
  movlw 0x03
  cpfseq TBLPTRL
  goto falso
verdadero:
  nop
  goto verdadero
falso:
  incf TBLPTRL, f
  goto inicio
end
  
```

23

Captura de pantalla del IDE MPASM mostrando el código fuente y el monitor de observación.

**Código Fuente:**

```

----- 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 ;Vector de RESET
0000  goto init_conf
----- 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
0026  movlw LOW datos
0028  movwf TBLPTRL ;Dirección del TBLF
----- loop:
002A  TBLRD* ;Accion de lectura
002C  movff TABLAT, LATD ;Muevo el contenido
0030  nop ;Micropausa
0032  incf TBLPTRL, f ;Incremento la posici
0034  TBLRD* ;Accion de lectura
0036  movff TABLAT, LATD
003A  nop
003C  incf TBLPTRL, f
003E  TBLRD* ;Accion de lectura
0040  movff TABLAT, LATD
0044  nop
0046  incf TBLPTRL, f
0048  TBLRD* ;Accion de lectura
004A  movff TABLAT, LATD
end
  
```

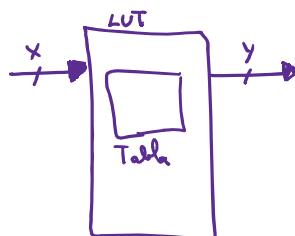
**Monitor de Observación (Watch Window):**

Name	Address	Value
TRISD	0x0F95	0b00000000
PORTD	0x0F83	0b00000000
LATD	0x0F8C	0x89
STATUS	0x0FD8	0b00000000
PCLATH	0x0FFB	0b00000000
PCLATH	0x0FFA	0x00
PCL	0x0FF9	0x22
TBLPTRU	0x0FF8	0b00000000
TBLPTRH	0x0FF7	0x05
TBLPTRL	0x0FF6	0x03
TABLAT	0x0FF5	0x89
cuenta	0x0020	0x00
WREG	0x0FE8	0x00

24

## Tablas de búsqueda (lookup tables - LUT)

- Decodificadores implementados en software

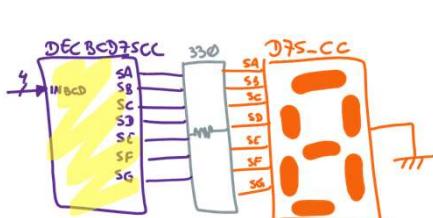


- Dos maneras de implementar LUT en MPASM-PIC18
  - Utilizando la funcionalidad del PC
  - Utilizando el TBLPTR

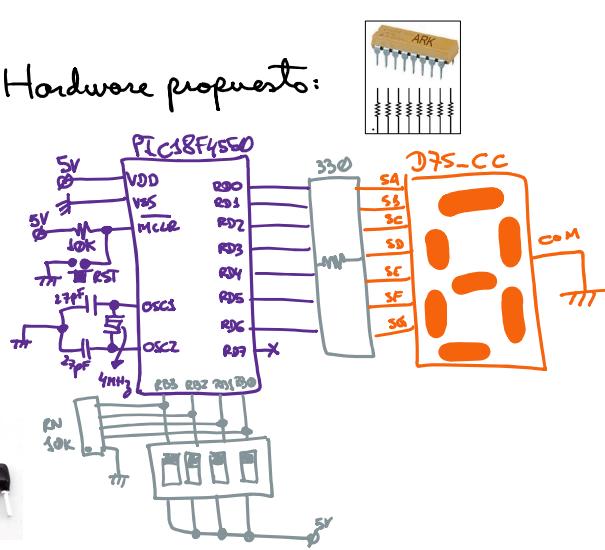
25

## Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:



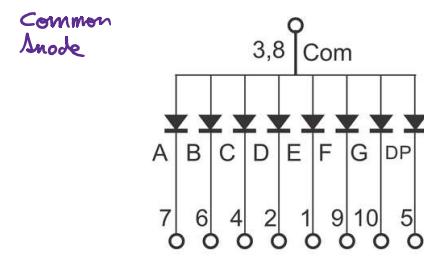
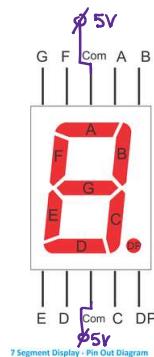
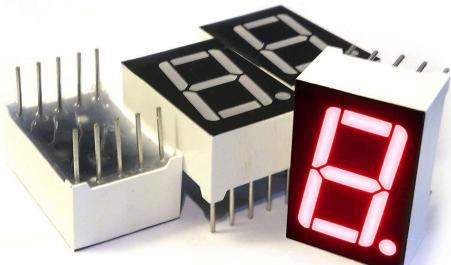
Hardware propuesto:



26

## Observación:

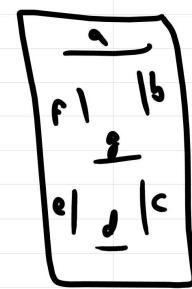
- Los pines denominados “comunes” (Com) son el mismo nodo, la misma conexión!



27

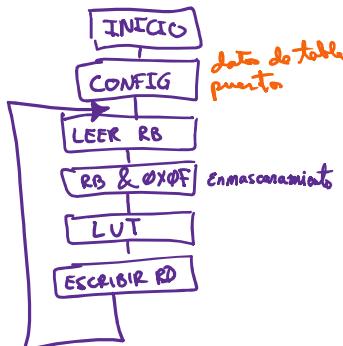
Desarrollo de la tabla de decodificación para display de 7 segmentos cátodo común:

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

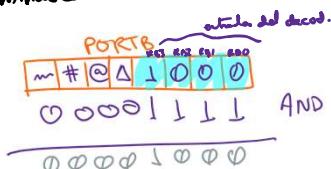


28

## 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

39      loop:
40          movf PORTB, w
41          andlw 0x0F
42          addwf PCL, f
43          (w)retlw 0x3F
44          (s)retlw 0x06
45          (z)retlw 0x5B
46          (c)retlw 0x4F
47          retlw 0x66
48          retlw 0x6D
49          retlw 0x7D
50          retlw 0x07
51          retlw 0x7F
52          retlw 0x67
53      end

```

Mem\_frog

movwf 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!

29

## 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

30

## 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 MCRLRE = ON         ; MCRLR Pin Enable bit (MCRL 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, 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

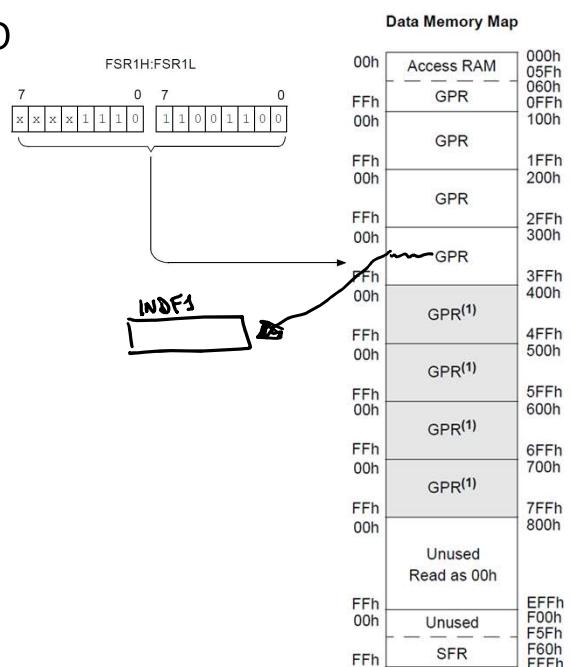
```

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

31

## 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)



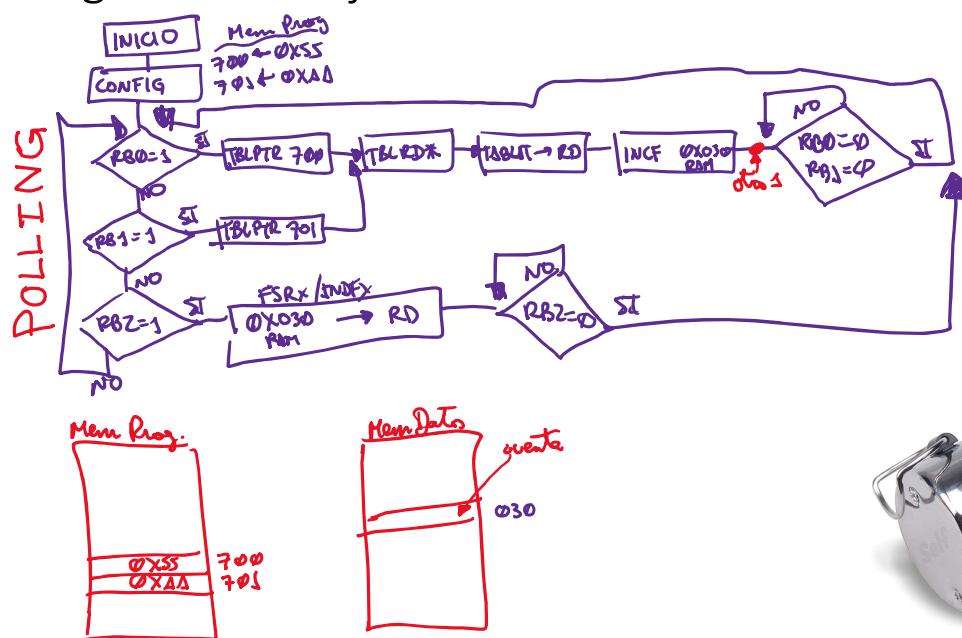
32

## Ejercicio:

- Un dato 0x55 estará almacenado en la dirección 0x0700 y otro dato 0xAA en la dirección 0x0701 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 0x0700 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 0x0701 de la memoria de programa.
- Cada pulsación que se haga deberá de contarse y registrarse esa cuenta en la dirección 0x030 de la **memoria de datos**.
- Adicionalmente se contará con un botón en RB2 el cual al ser accionado leerá el contenido de la dirección 0x030 de la memoria de datos y lo arrojará por el RD.

33

## Diagrama de flujo



34

## Código en MPASM

```

1 ;Este es un comentario, se le antecede un punto
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 PLLDIV = 1 ; PLL Prescaler Division by 1
7 CONFIG CPUDIV = OSC1_PLL2 ; System Clock Division by 1
8 CONFIG FOSC = XT_XT ; Oscillator Selection: XT
9 CONFIG PWRT = ON ; Power-up Timer On
10 CONFIG BOR = OFF ; Brown-out Reset Off
11 CONFIG WDT = OFF ; Watchdog Timer Off
12 CONFIG CCP2MX = ON ; CCP2 MUX: MUX1/CCP2
13 CONFIG PBADEN = OFF ; PORTB A/D Enable: Off
14 CONFIG MCORE = ON ; MCLR Pin Configuration: On
15 CONFIG LVP = OFF ; Single-Supply Operation: Off
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 init_conf:
27     clrf TRISD ;Por acá saldrán los datos que
28 ;Vamos a colocarle la dirección inicial de TBLPTRH
29     movlw HIGH datos
30     movwf TBLPTRH
31     movlw LOW datos
32     movwf TBLPTRL ;Aqui el punto YA SE ENCUENTRA
33     clrf cuenta ;Forzamos al registro GPR que
34
35         loop:
36             btfss PORTB, 0 ;Preguntamos si se presionó botón 0
37             goto next1
38             goto accion1
39
40         next1:
41             btfss PORTB, 1 ;Preguntamos si se presionó botón 1
42             goto next2
43             goto accion2
44
45         next2:
46             btfss PORTB, 2 ;Preguntamos si se presionó botón 2
47             goto loop
48             goto accion3
49
50         accion1:
51             movlw 0x00
52             movwf TBLPTRL ;Para acceder a 0x0700 de la memoria programada
53             TBLRD*
54             movff TABLAT, LATD
55             incf cuenta, f ;Incremento la cuenta y se almacena en el registro
56
57         otro1:
58             btfsc PORTB, 0
59             goto otro1
60             goto loop
61
62         accion2:
63             movlw 0x01
64             movwf TBLPTRL ;Para acceder a 0x0700 de la memoria programada
65             TBLRD*
66             movff TABLAT, LATD
67             incf cuenta, f ;Incremento la cuenta y se almacena en el registro
68
69         otro2:
70             btfsc PORTB, 1
71             goto otro2
72             goto loop
73
74         accion3:
75             lfsr 0, 0x030 ;Asignación una dirección de memoria de datos a FSR0
76             movff INFO, LATD
77
78         otro3:
79             btfsc PORTB, 2
80             goto otro3
81             goto loop
82             end

```

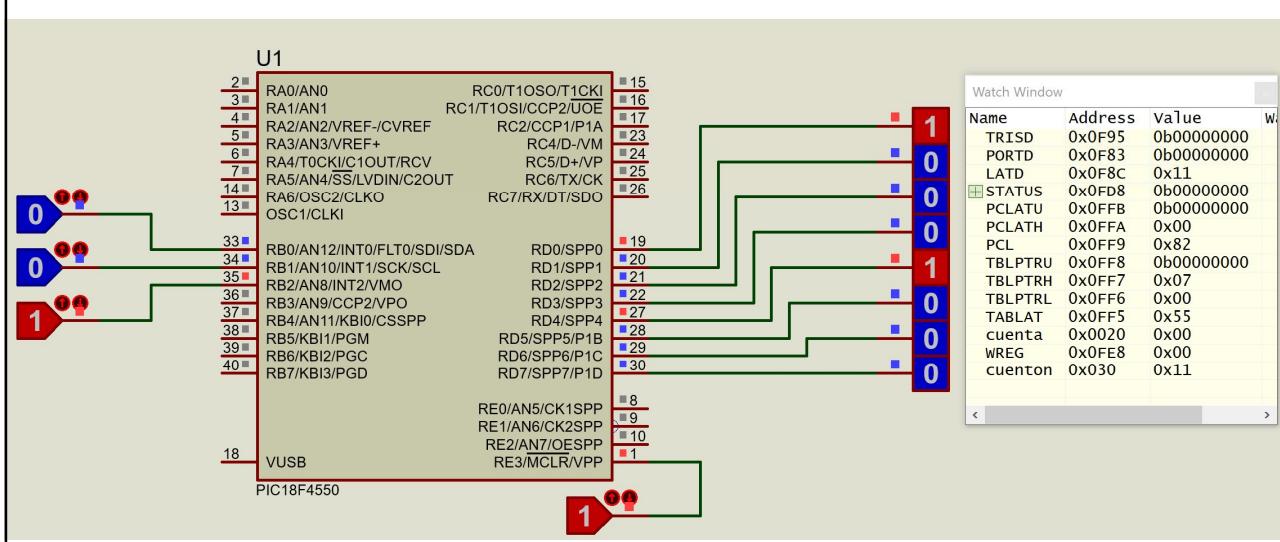
35

## Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler

36

## Simulación:



37

## Ejercicio:

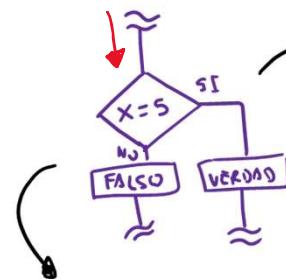
- Se tienen los siguiente datos que deberán encontrarse en la dirección de memoria de programa 0x0421:
   
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C,  
0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58,  
0x59, 0x5A
- Dichos números corresponden a letras 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	&#64;	Ø
65	41	101	&#65;	A
66	42	102	&#66;	B
67	43	103	&#67;	C
68	44	104	&#68;	D
69	45	105	&#69;	E
70	46	106	&#70;	F
71	47	107	&#71;	G
72	48	110	&#72;	H
73	49	111	&#73;	I
74	4A	112	&#74;	J
75	4B	113	&#75;	K
76	4C	114	&#76;	L
77	4D	115	&#77;	M
78	4E	116	&#78;	N
79	4F	117	&#79;	O
80	50	120	&#80;	P
81	51	121	&#81;	Q
82	52	122	&#82;	R
83	53	123	&#83;	S
84	54	124	&#84;	T
85	55	125	&#85;	U
86	56	126	&#86;	V
87	57	127	&#87;	W
88	58	130	&#88;	X
89	59	131	&#89;	Y
90	5A	132	&#90;	Z

38

## Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

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



En MPASM:

Usaremos CPFSEQ:

$\text{CPFSEQ } [\text{reg}]$

En lenguaje de alto nivel:

```

if (x = 5) {
    [VERDAD]
} else {
    [Falso]
}
  
```

$$(f) = w$$

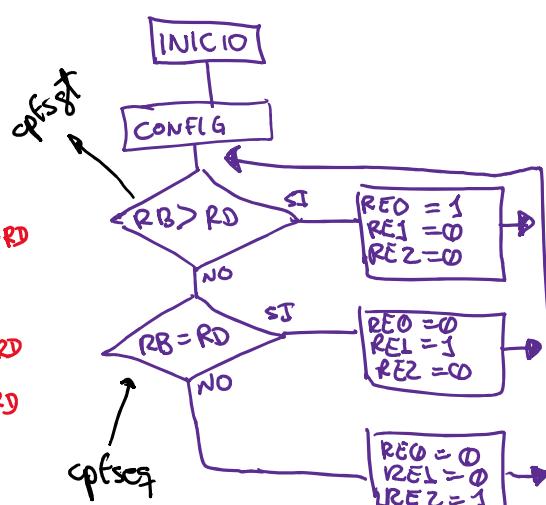
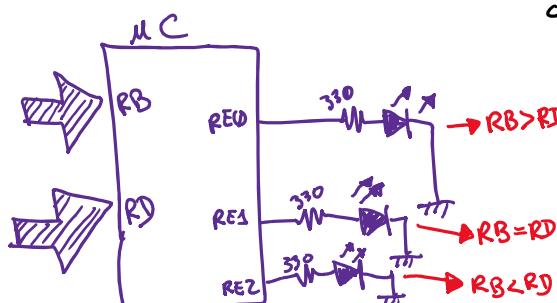
```

movlw .5
cpfseq x-reg
    goto falso
    goto verdad
  
```

39

## Ejemplo:

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



40

## Código MPASM:

```

1 ;Este es un comentario, se le asigna
2 list p=18f4550      ;Modelo
3 #include <p18f4550.inc>
4
5 ;Directivas de preprocessador
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 ;Aquí falta algo que no me acuerdo
25
26         loop:
27             movf PORTD, W
28             cpfsag PORTB
29             goto next1
30             bsf LATE, 0
31             bcf LATE, 1
32             bcf LATE, 2
33             goto loop
34 next1:
35             movf PORTD, W
36             cpfseq PORTB
37             goto next2
38             bcf LATE, 0
39             bsf LATE, 1
40             bcf LATE, 2
41             goto loop
42 next2:
43             bcf LATE, 0
44             bcf LATE, 1
45             bsf LATE, 2
46             goto loop
47         end

```

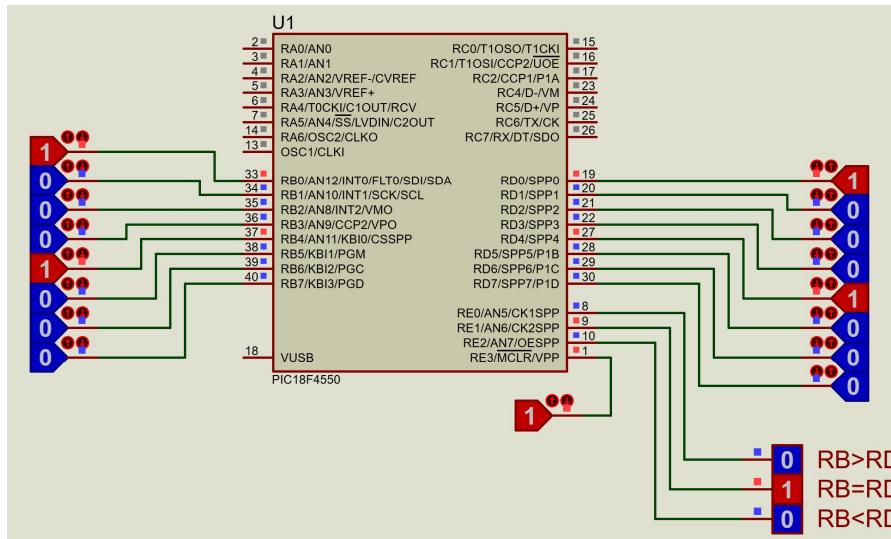
41

## Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler

42

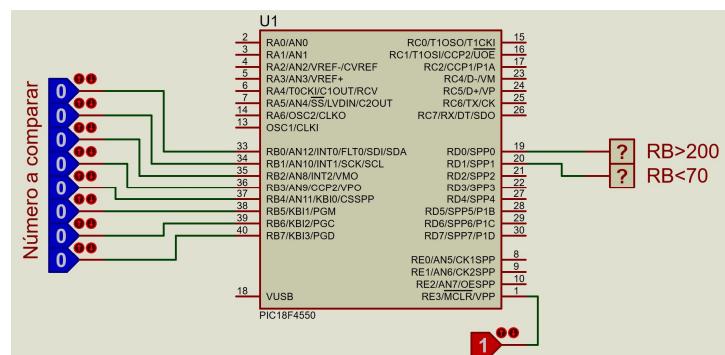
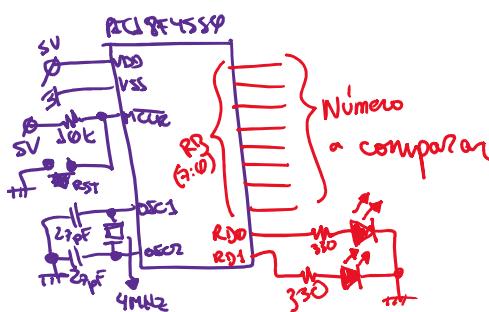
## Simulación



43

## Ejemplo:

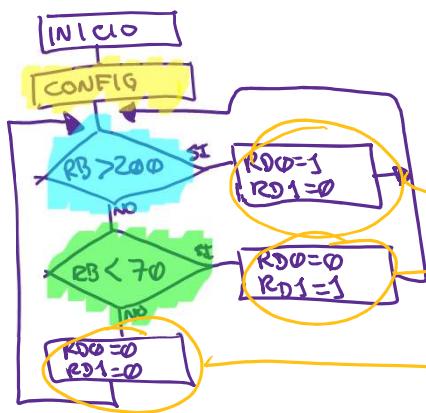
- Desarrollar un programa para que compare lo que se esta ingresando en RB y arroje lo siguiente: RD0=1 cuando RB>200 y RD1=1 cuando RB<70, cuando no se cumplan las dos condiciones las dos salidas permanecerán en cero.



44

Cont.

Diagrama de flujo:



```

1 ;Este es un comentario, se le asigna el numero 1
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 PWRT = 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     bsf 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     bsf LATD, 1
38     goto loop
39 next2:
40     bcf LATD, 0
41     bcf LATD, 1
42     goto loop
43 end
  
```

45

Asignación:

- Pasar el código anterior al nuevo formato XC8 PIC Assembler

46

Fin de la sesión