

# Microcontroladores

## Semana 3

Semestre 2021-1

Profesor: Kalun José Lau Gan

*Empezamos en breve.*

1

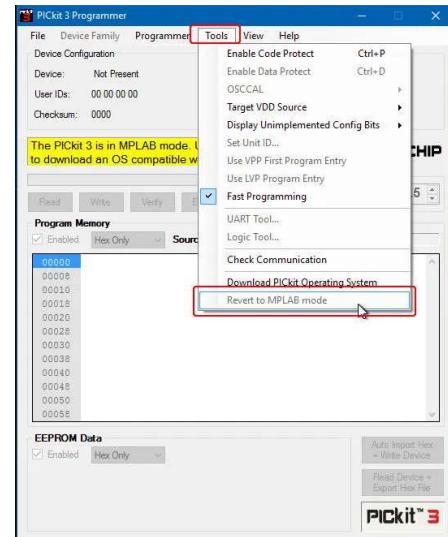
### Preguntas previas:

- 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
  - Emplear el canal de contacto de Whatsapp para enviar imágenes y videos ya que la plataforma del blackboard no permite
  - En algunos casos los protoboard tienen las líneas de alimentación partidas en la mitad, asegurarse de conectarlos antes de implementar los circuitos.

2

## Preguntas previas:

- 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:

- (Hagan sus consultas)

4

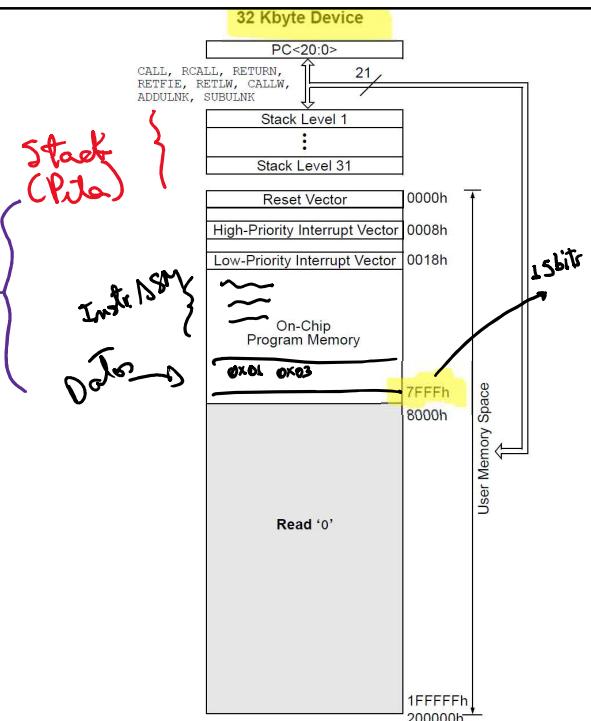
## Agenda:

- Memoria de programa del microcontrolador PIC18F4550
- El contador de programa 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

5

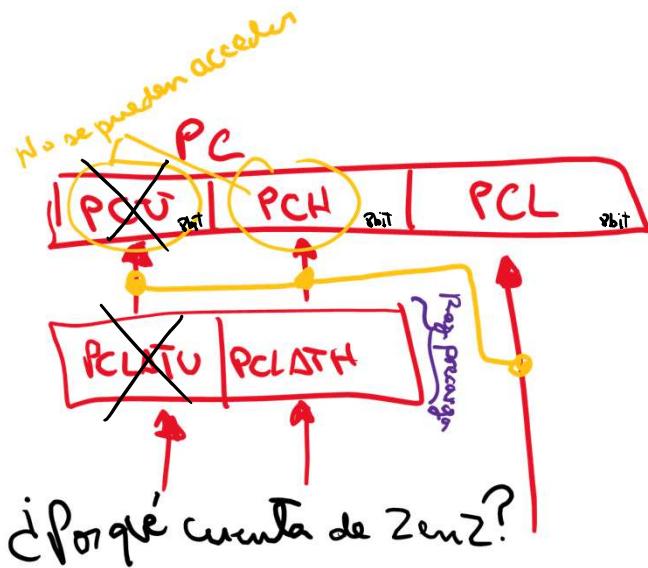
## Memoria de programa del PIC18F4550

- Referencia: Item 5, página 59 de hoja técnica rev. E.
- Tamaño: 32KByte (0x0000 a 0x7FFF).
- 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 y luego se lee, se obtendrá 0.
- Tamaño máximo de la memoria de programa de la familia PIC18: 2Mbyte (21 bits en direcciones)



6

## 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 PCLTU 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).

7

## Ejemplo

- Cargar dirección 0x00288A en PC:

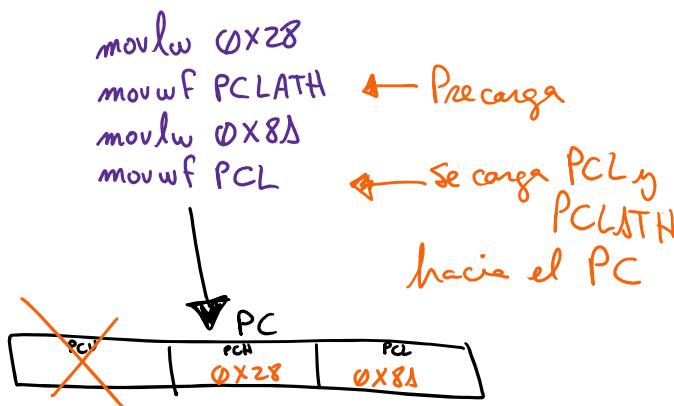


TABLE 5-1: SPECIAL FUNCTION REGISTER MAP							
Address	Name	Address	Name	Address	Name	Address	Name
FF9h	TOSU	FDFh	INDF2 <sup>(1)</sup>	FB8h	CCPR1H		
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FB8h	CCPR1L		
FFDh	TOSL	FDCh	POSTDEC2 <sup>(1)</sup>	FB8h	CCPICON		
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FB8h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

8

## 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

```

0x0000  
0x0020  
0x0022  
0x0024  
0x0026

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

9

## En el Proteus:

```

----- ;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 Enable bit (MCLRE<4>)
-----
----- 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

```

Name	Address	Value	Watch Expression
TRISD	0x0F95	0b00000000	
TRISB	0x0F93	0b11111111	
PORTB	0x0F81	0b10111110	
PORTD	0x0F83	0b00000000	
LATB	0x0F8A	0b00000000	
LATD	0x0F8C	0b10000001	
STATUS	0x0FD8	0b00000000	
PCL	0x0FF9	0x24	
PCLATH	0x0FFA	0b00000000	
PCLATU	0x0FFB	0b00000000	

10

## ¿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



11

## ¿Cómo funciona el PC?

- Se tiene el siguiente programa

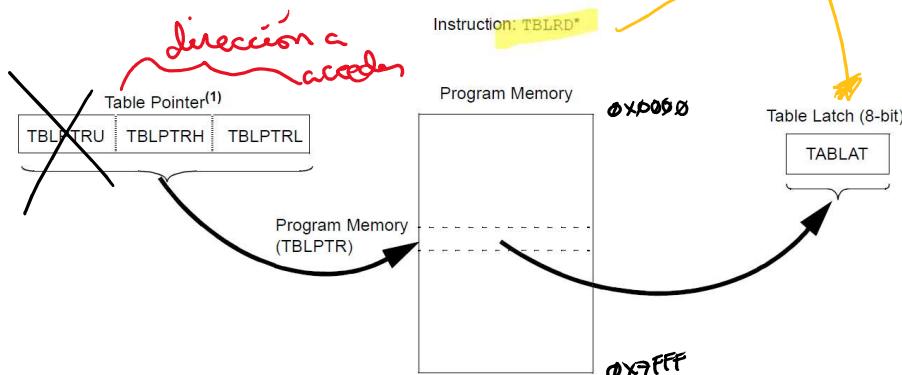
```
org 0x0000
inicio: clrf TRISD
        movlw 0x05
        movwf LATD
        goto inicio 0x0105
end
```

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



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 PIC18F4550 15 bits)
- Se emplea para acceder a la memoria de programa y leer (también escribir pero es mas complicado)
- 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

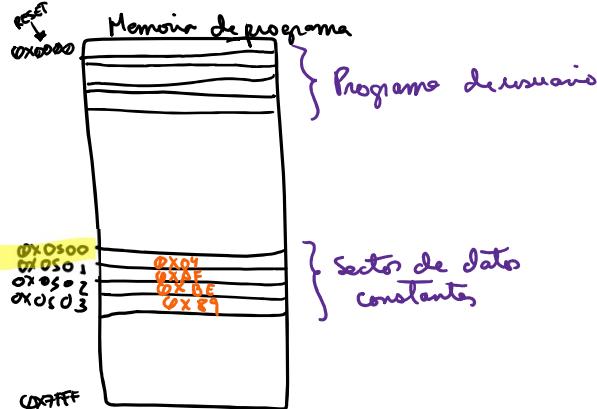
13

## Ejemplo:

- Desarrollar un programa donde se tenga almacenado los siguientes datos en la memoria de programa:
  - 0x500: 0x04
  - 0x501: 0xAF
  - 0x502: 0xBE
  - 0x503: 0x89
- Elaborar un algoritmo que permita leer los datos anteriores y arrojarlas de manera secuencial a través de RD con periodo de NOP

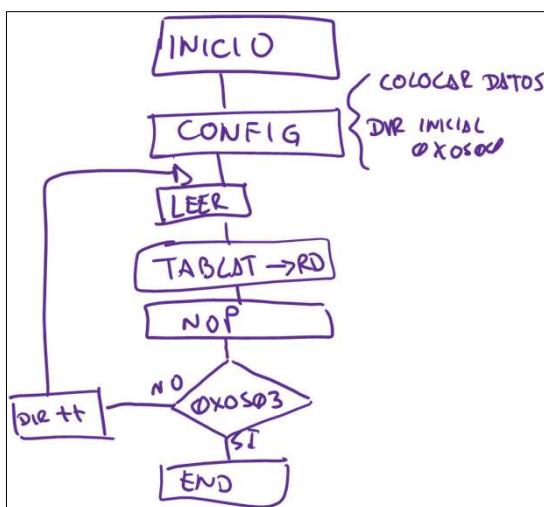
14

## Desarrollo del ejemplo:



15

## Diagrama de flujo y código del ejemplo:



```

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 (con
9      CONFIG PBADEN = OFF            ; PORTB A/D Enable bit (PORTB<4:0> pins are co
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 configurando
17
18     org 0x0020                   ;Zona de programa de usuario
19     configurando:
20     clrf TRISD                 ;Todo RD como salida
21     movlw HIGH numeros
22     movwf TBLPTRH
23     movlw LOW numeros
24     movwf TBLPTRL               ;Carga de la dirección de apunte de TBLPTR (0x0500)
25
26     inicio:
27     TBLRD*
28     movff TABLAT, LATD
29     nop
30     movlw 0x03
31     cpfseq TBLPTRL
32     goto falso
33     verdadero:
34     nop
35     goto verdadero
36
37     falso:
38     incf TBLPTRL, f
39     goto inicio
40
41     end

```

16

The screenshot shows the MPASM PIC18 CPU Source Code window. The assembly code is as follows:

```

----- 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
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
-----
```

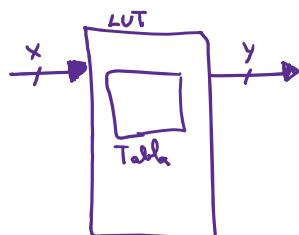
On the right, there is a memory dump window titled "Watch Window" showing the state of various registers and memory locations. A yellow callout box highlights the TBLPTRH and TBLPTRL registers, which are connected to a truth table diagram below. The truth table shows the relationship between the address bits (15-16, 17-20, 21-24, 25-26) and the output bits (RD0-SPP0 to RD7-SPP7). The output bits are labeled with values 0 or 1, corresponding to the states shown in the memory dump.

Name	Address	Value
TRISD	0x0F95	0b00000000
PORTD	0x0F83	0b00000000
LATD	0x0F8C	0x89
STATUS	0x0FD8	0b00000000
PCLATH	0x0FFB	0b00000000
<b>PCL</b>	<b>0xFF9</b>	<b>0x22</b>
TBLPTRU	0xFF8	0b00000000
TBLPTRH	0xFF7	0x05
TBLPTRL	0xFF6	0x03
TABLAT	0xFF5	0x89
cuenta	0x020	0x00
WREG	0xFE8	0x00

17

## Tablas de búsqueda (lookup tables)

- Decodificadores implementados en software

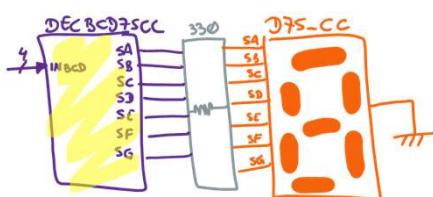


- Dos maneras de implementar LUT en MPASM-PIC18
  - Utilizando la funcionalidad del PC
  - Utilizando el TBLPTR

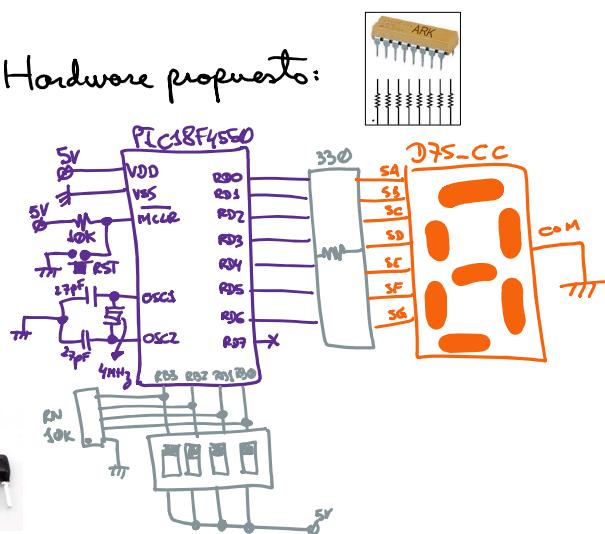
18

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

Idea:

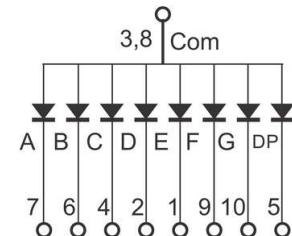
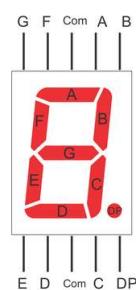
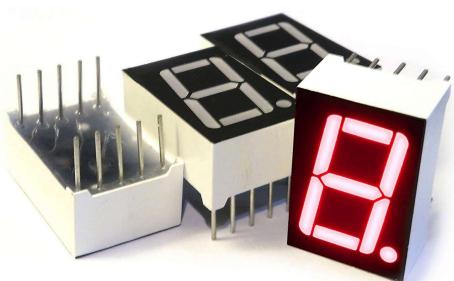


Hardware propuesto:



19

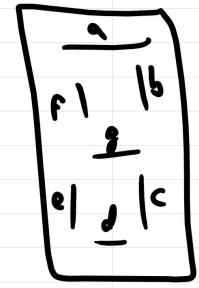
Observación:



20

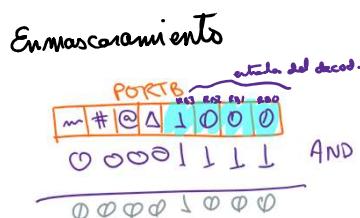
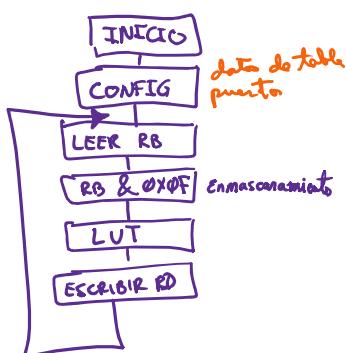
## Desarrollo de la tabla de decodificación

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

## Diagrama de flujo



### Código previo

```

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          (0)retlw 0x3F
42          (1)retlw 0x06
43          (2)retlw 0x5B
44          (3)retlw 0x4F
45          retlw 0x66
46          retlw 0x6D
47          retlw 0x7D
48          retlw 0x07
49          retlw 0x7F
50          retlw 0x67
51
52
53      end
  
```

Mem brog

loop:

- Movf val... ,W
- addwf PCL, F
- retlw 0x3F
- retlw 0x06
- retlw 0x5B
- retlw 0x4F
- retlw 0x66
- retlw 0x6D
- retlw 0x7D
- retlw 0x07
- retlw 0x7F
- retlw 0x67

enmascaramiento

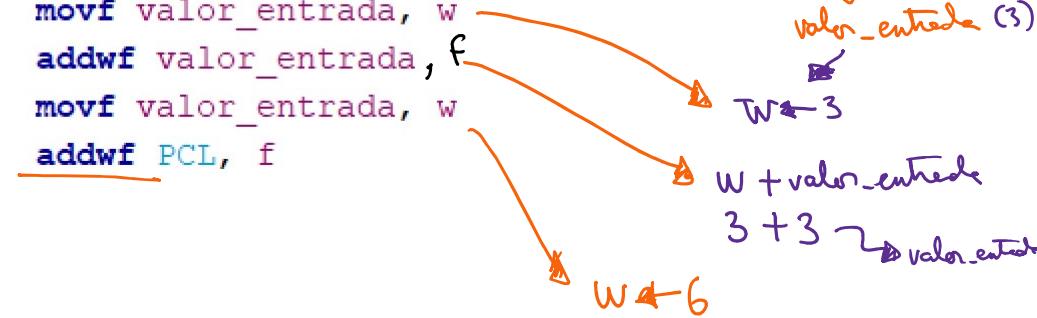
0x+0  
0x+1  
0x+2  
0x+3  
0x+4  
0x+5  
0x+6  
0x+7  
0x+8  
0x+9

22

## 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
```



23

## Modificación del decodificador empleando TBLPTR

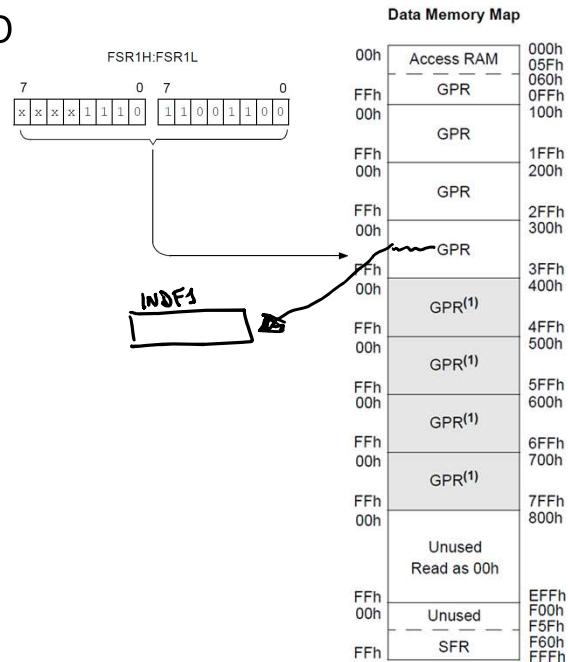
```
2     list p=18f4550      ;Modelo del microcontrolador
3     #include <18f4550.inc>    ;Llamada a la librería de nombre de los registros
4
5     ;Directivas de preprocesador o bits de configuración
6     CONFIG PLLDIV = 1          ; PLL Prescaler Selection bits (No prescale (4 MHz oscillator input drive)
7     CONFIG CFUDIV = 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 PWRT = 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 MCLRE = ON          ; MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)
15    CONFIG LVF = 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
```

24

## 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)



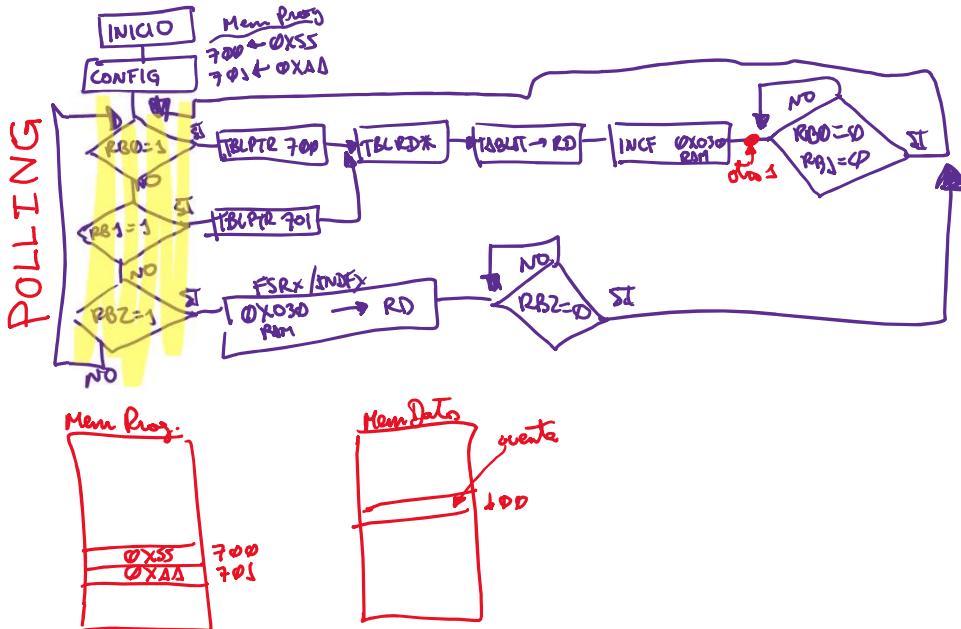
25

## 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 en la dirección 0x30 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.

26

## Diagrama de flujo



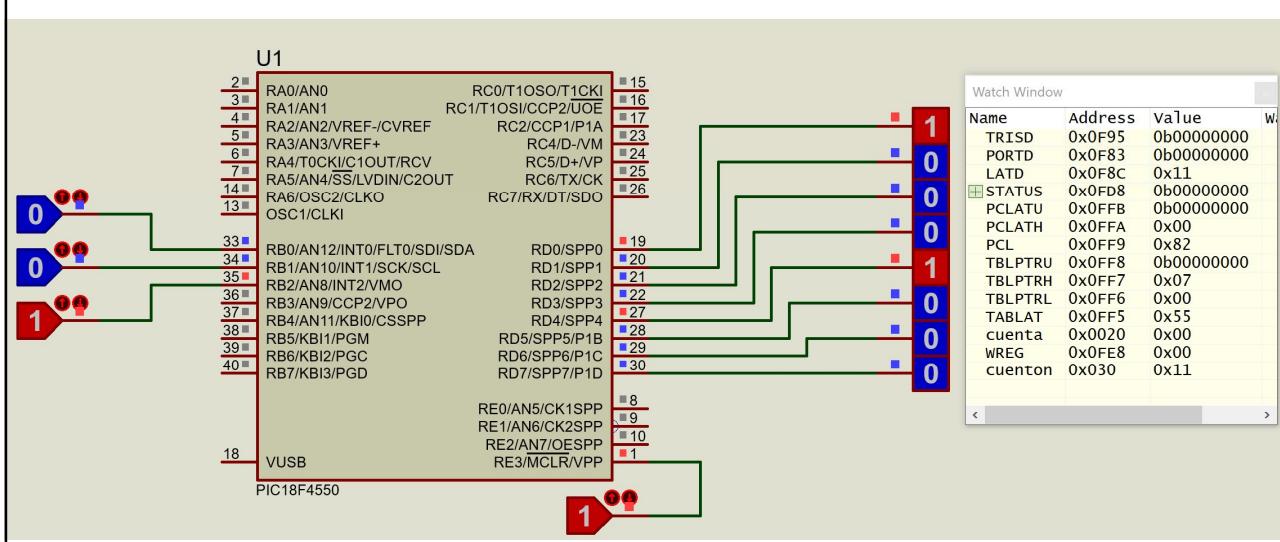
27

## Código en MPASM

<pre> 1 ;Este es un comentario, se le antecede un punto 2 list p=18f4550 ;Modelo del microcontrolador 3 #include &lt;p18f4550.inc&gt; ;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 Division 8 CONFIG FOSC = XT_XT ; Oscillator Selection 9 CONFIG PWRT = ON ; Power-up Timer 10 CONFIG BOR = OFF ; Brown-out Reset 11 CONFIG WDT = OFF ; Watchdog Timer 12 CONFIG CCP2MX = ON ; CCP2 MUX 13 CONFIG PBADEN = OFF ; PORTB A/D 14 CONFIG MC1RE = ON ; MCLR Pin 15 CONFIG LVP = OFF ; Single-Supply Operation 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 </pre>	<pre> 35 loop: 36     btfss PORTB, 0 ;Preguntamos si se presionó botón0 37     goto next1 38     goto accion1 39 40 next1: 41     btfss PORTB, 1 ;Preguntamos si se presionó botón1 42     goto next2 43     goto accion2 44 45 next2: 46     btfss PORTB, 2 ;Preguntamos si se presionó botón2 47     goto loop 48     goto accion3 49 50 accion1: 51     movlw 0x00 52     movwf TBLPTRL ;Para acceder a 0x0700 de la mem prog 53     TBLRD* 54     movff TABLAT, LATD 55     incf cuenta, f ;Incremento la cuenta y se almacena en el registro 56 57     btfsc PORTB, 0 58     goto otro1 59     goto loop 60 61     accion2: 62     movlw 0x01 63     movwf TBLPTRL ;Para acceder a 0x0700 de la mem prog 64     TBLRD* 65     movff TABLAT, LATD 66     incf cuenta, f ;Incremento la cuenta y se almacena en el registro 67 68     otro2: 69     btfsc PORTB, 1 70     goto otro3 71     goto loop 72 73     accion3: 74     lfsr 0, 0x030 ;Asignación una dirección de memoria de datos a FSR0 75     movff INDF0, LATD 76 77     otro3: 78     btfsc PORTB, 2 79     goto otro3 80     goto loop 81     end </pre>
--	--

28

## Simulación:



29

## Ejercicio:

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

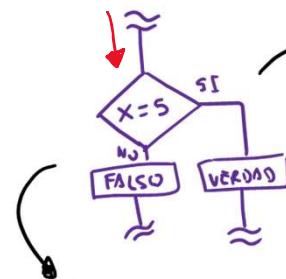
0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A
- Dichos números corresponden a letras en mayúscula según tabla de caracteres ASCII (ver tabla de la derecha)
- Desarrollar un programa (previamente 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

30

## 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
    
```

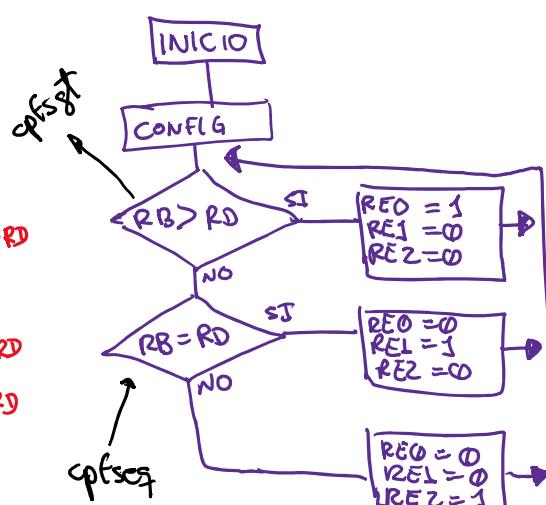
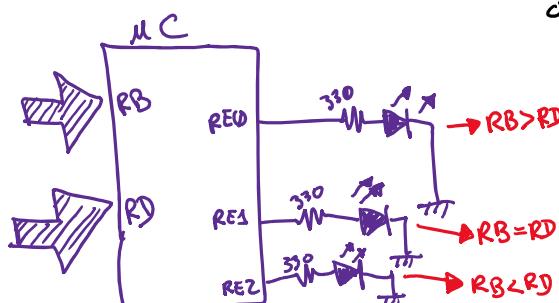
$$x = 5$$

$\text{goto Falso}$   
 $\text{goto Verdad}$

31

## Ejemplo:

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



32

## Código MPASM:

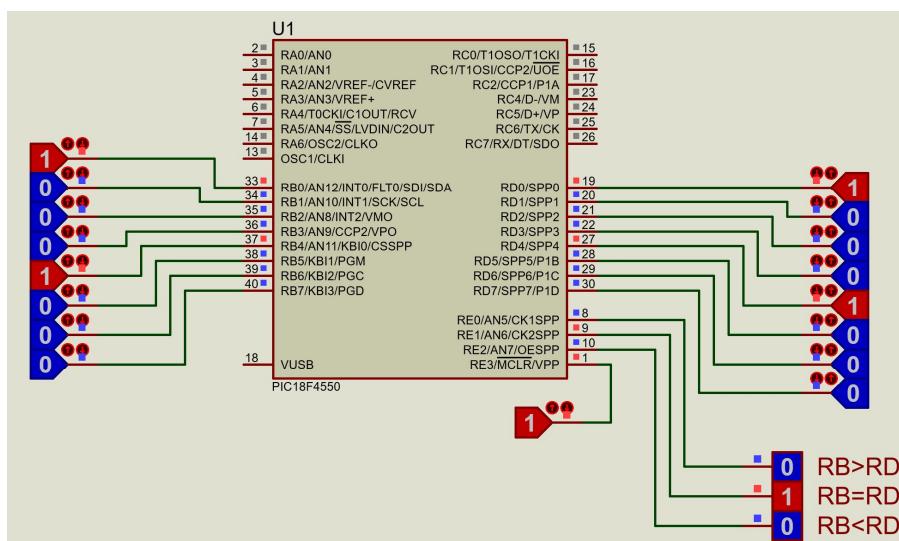
```

1 ;Este es un comentario, se le asigna
2 list p=18f4550           ;Modelo
3 #include <p18f4550.inc>
4
5 ;Directivas de preprocesado:
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             cpfsgt 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             bcf LATE, 1
40             bcf LATE, 2
41             goto loop
42 next2:
43             bcf LATE, 0
44             bcf LATE, 1
45             bcf LATE, 2
46             goto loop
47         end

```

33

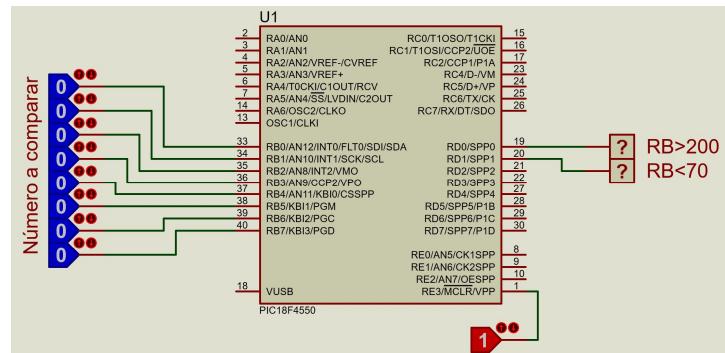
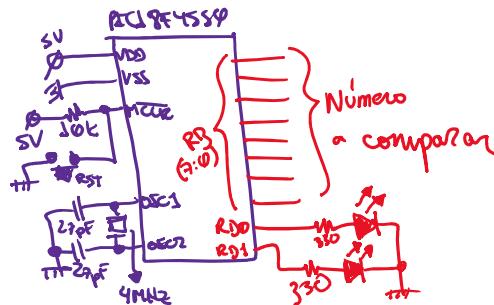
## Simulación



34

## Ejemplo:

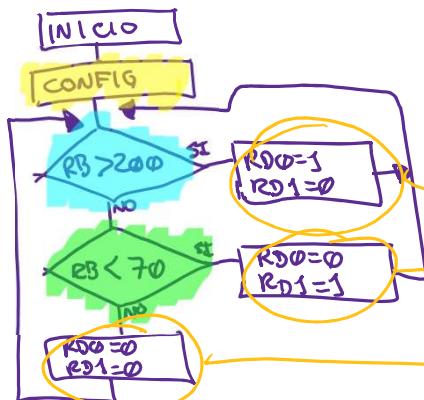
- Desarrollar un programa para que compare lo que se está ingresando en RB y arroje lo siguiente: RD0=1 cuando RB>200 y RD1=1 cuando RB<70, cuando no se cumplan las dos condiciones las dos salidas permanecerán en cero.



35

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

```

36

Fin de la sesión