

Microcontroladores

Semana 2

Semestre 2021-0

Profesor: Kalun José Lau Gan

1

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

2

Preguntas previas

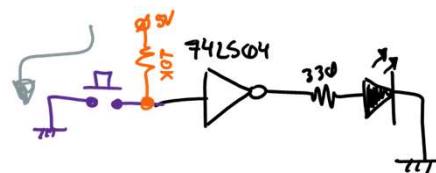
- ¿En dónde encuentro microcontroladores en el mundo real?

- Fuentes conmutadas (switching PSU)



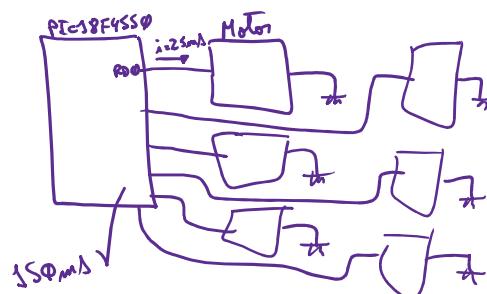
- Pandemia por COVID-19: Ventilador mecánico, termómetro infrarrojo, pulsioxímetro
- Y en muchos dispositivos electrónicos de uso cotidiano!

- ¿Qué son las resistencias de pull-up?



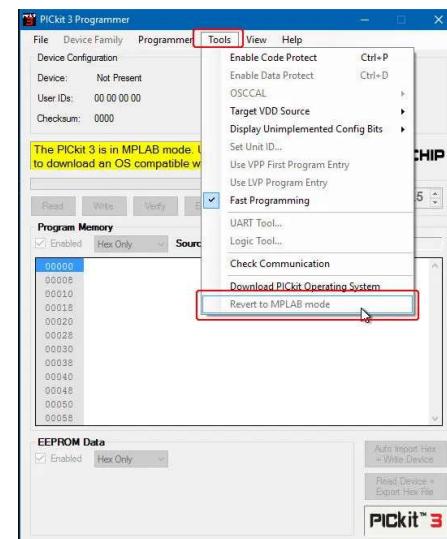
3

- Estoy usando el PICKIT2 y a veces funciona y a veces no funciona, qué puede ser?
 - Revisa continuidad de cables jumper
- Corriente máxima de pin I/O?
 - 25mA pero máximo 200mA en total



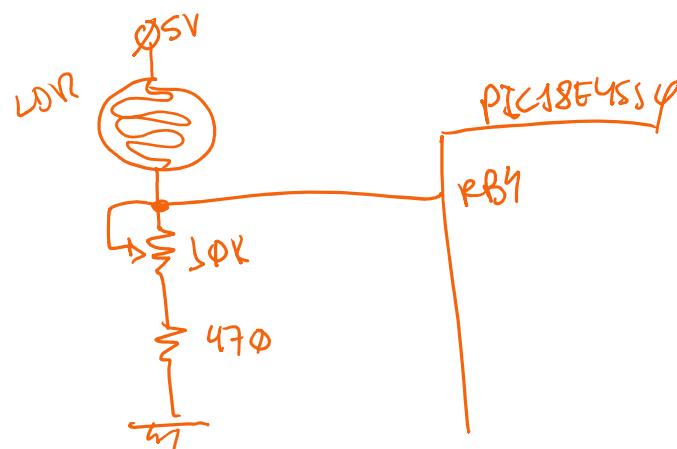
4

- 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



5

- Circuito de LDR para la vela electrónica



6

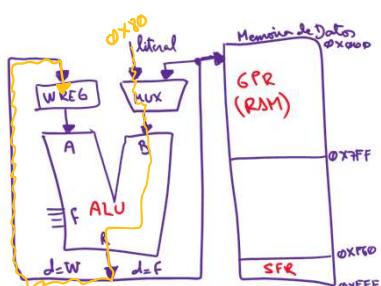
Instrucciones básicas en MPASM

- Instrucciones de movimiento de datos

movlw [literal]

- mover un literal hacia WREG

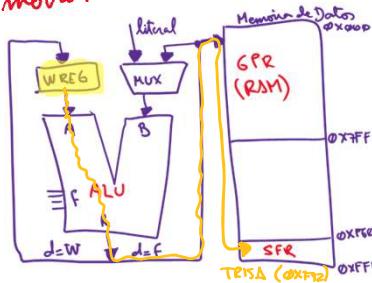
Ej: movlw 0x80



movwf [registro]

- mover el contenido de WREG hacia [registro]

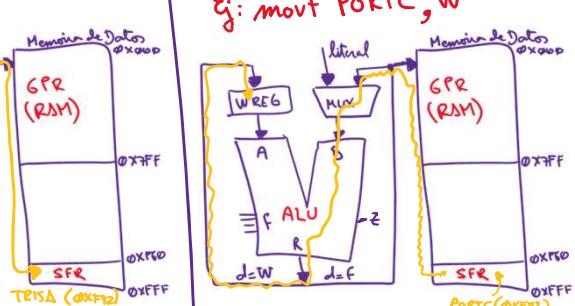
Ej: movwf TRISA



movf [registro], d

mover el contenido de [registro] y lo muerde según "d".

Ej: movf PORTC, W



Note: movf [registro], F sirve para act. flag

7

Instrucciones básicas en MPASM

- Instrucciones de manipulación de bits en un registro

bsf [registro], #bit
coloca a '1' el #bit de [registro]

Ej:

 bsf TRISB, 3

bcf [registro], #bit
coloca a '0' el #bit de [registro]

Ej:

 bcf TRISB, 7

btg [registro], #bit
aplica complemento al #bit de [registro]

Ej: btg TRISB, 6

8

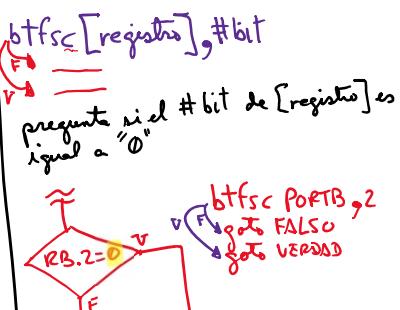
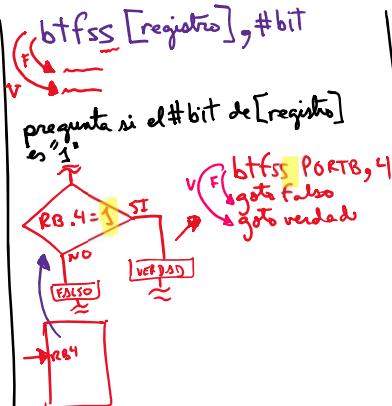
Instrucciones básicas en MPASM

goto [etiqueta]
salto incondicional

Ej: bucle: bsf LATD, 6
goto bucle

call [etiqueta]
salto a subroutine

Ej:
call previo
previo: return



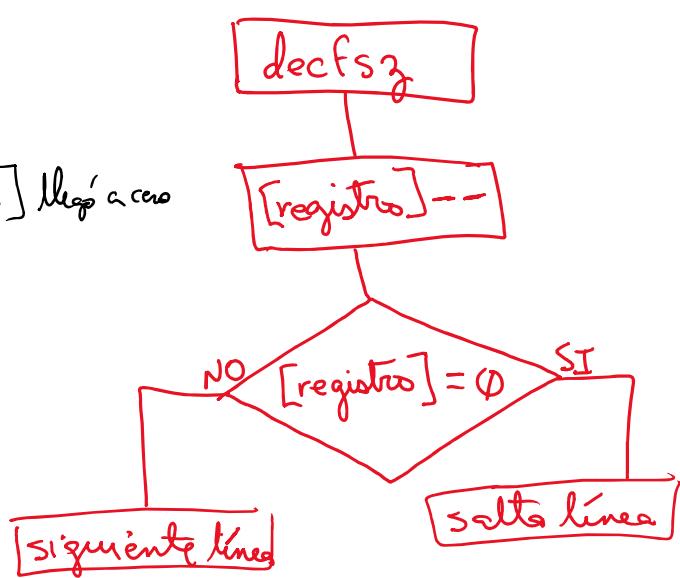
mop
instrucción de no operación
lo podemos considerar como un micro retraso

9

Instrucciones básicas en MPASM

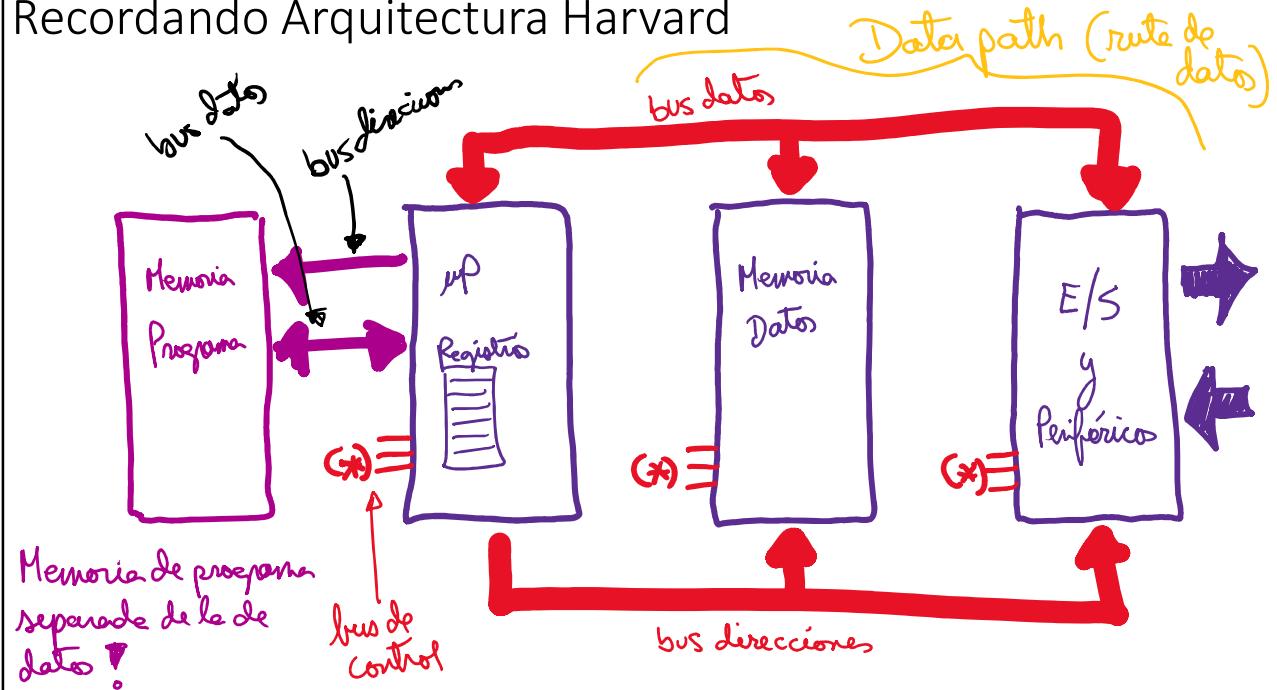
decfsz [registro], d

decrementa y pregunta si [registro] llegó a cero



10

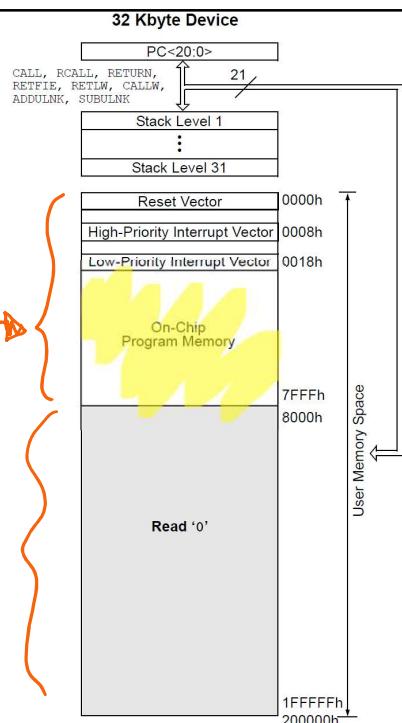
Recordando Arquitectura Harvard



11

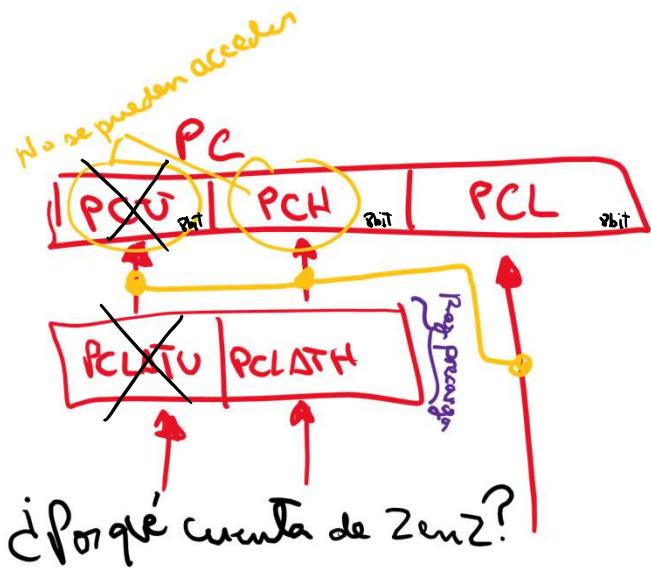
Memoria de programa del PIC18F4550

- 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 y luego se lee, se obtendrá 0.
- Tamaño máximo de la memoria de programa de la familia PIC18: 2Mbyte (21 bits en direcciones)



12

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).

13

Ejemplo

- Cargar dirección 0x00366A en PC:

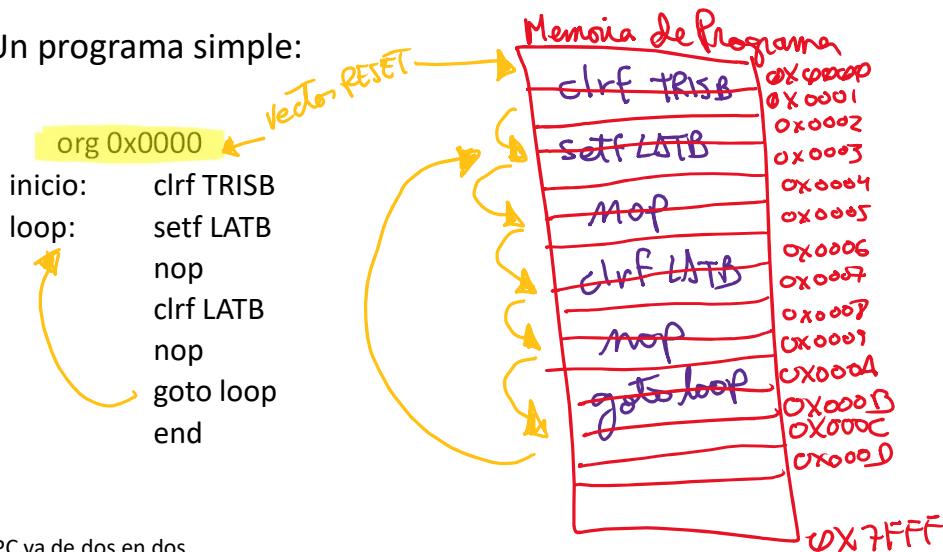
```

    clrf PCLATU
    movlw 0X36
    mouwf PCLATH
    movlw 0X6A
    mouwf PCL
  
```

14

¿Cómo funciona el PC?

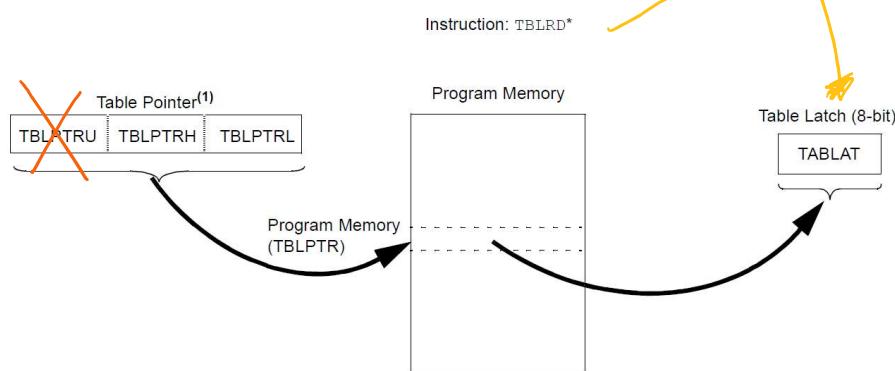
- Un programa simple:



Nota: PC va de dos en dos

15

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) su contenido.
 - 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

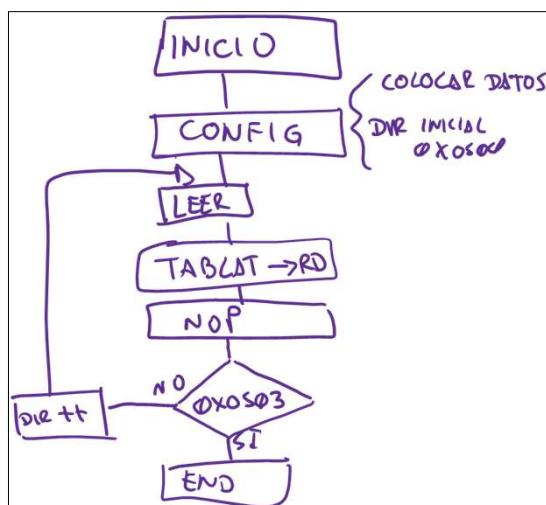
16

Ejemplo:

- Desarrollar un programa donde se tenga almacenado los siguientes datos en la memoria de programa:
 - 0x0500: 0x04
 - 0x0501: 0xAF
 - 0x0502: 0xBE
 - 0x0503: 0x89
- Desarrollar un algoritmo que permita leer los datos anteriores y arrojarlas de manera secuencial a través de RD con periodo de NOP

17

Diagrama de flujo y código del ejemplo:



```

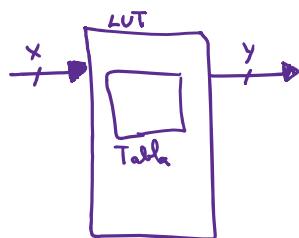
2      list p=18f4550
3      #include <p18f4550.inc> ;libreria de nombre de los registros sfr
4
5      CONFIG FOSC = XT_XT ; Oscillator Selection bits (XT oscillator (XT))
6      CONFIG PWRT = ON ; Power-up Timer Enable bit (PWRT enabled)
7      CONFIG BOR = OFF ; Brown-out Reset Enable bit (Brown-out Reset)
8      CONFIG WDT = OFF ; Watchdog Timer Enable bit (WDT disabled (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         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         movwf 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

```

18

Tablas de búsqueda (lookup tables)

- Decodificadores implementados en software

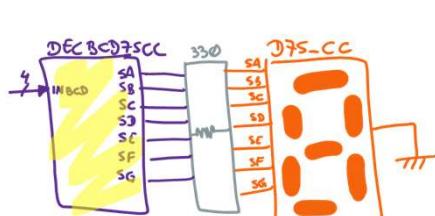


- Dos maneras de implementar LUT en MPASM-PIC18
 - Utilizando la funcionalidad del PC
 - Utilizando el TBLPTR

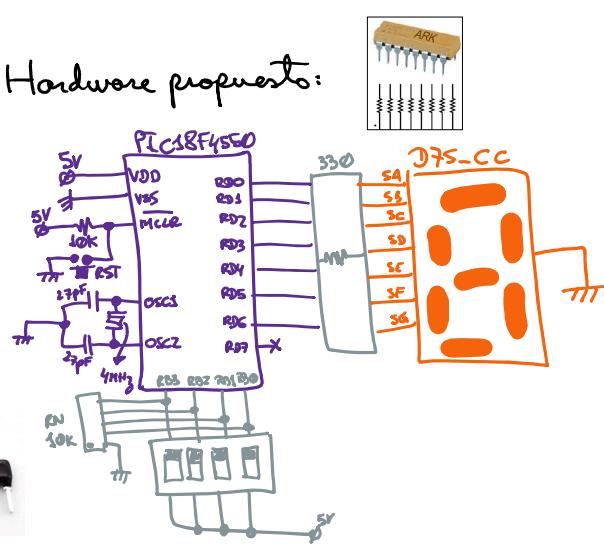
19

Ejemplo de decodificador BCD a 7 segmentos cátodo común con PC

Idea:



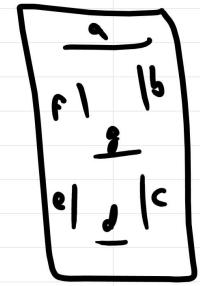
Hardware propuesto:



20

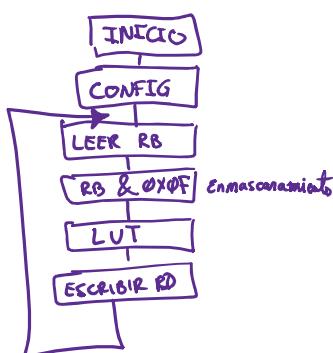
Desarrollo de la tabla de decodificación

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

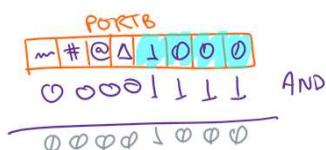


21

Diagrama de flujo



Enmascaramiento



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. prog

loop:
 movf val... ,W
 x+0
 andlw 0x0F
 x+1
 movwf valor_entrada
 x+2
 call tabla_pc
 x+3
 movwf LATD
 x+4
 goto loop
 x+5
 x+6
 x+7
 x+8
 x+9
 :

tabla_pc:
 movf valor_entrada, W
 addwf PCL, F
 (0)retlw 0x3F
 (1)retlw 0x06
 (2)retlw 0x5B
 (3)retlw 0x4F
 retlw 0x66
 retlw 0x6D
 retlw 0x7D
 retlw 0x07
 retlw 0x7F
 retlw 0x67

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
```

PORTB (3)

↓
enmascaramiento

valor_entrada (3)

W ← 3

W + valor_entrada

3 + 3 → valor_entrada

W ← 6

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
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

Registro STATUS

- Contiene las banderas del CPU, esas banderas serán actualizadas luego de operar

REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC ⁽¹⁾	C ⁽²⁾
bit 7							bit 0

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

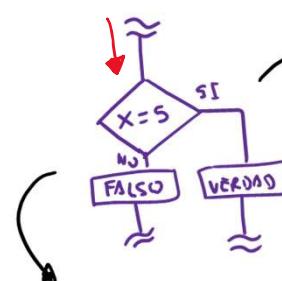
bit 7-5	Unimplemented: Read as '0'
bit 4	N: Negative bit This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1). 1 = Result was negative 0 = Result was positive
bit 3	OV: Overflow bit This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state. 1 = Overflow occurred for signed arithmetic (in this arithmetic operation) 0 = No overflow occurred
bit 2	Z: Zero bit 1 = The result of an arithmetic or logic operation is zero 0 = The result of an arithmetic or logic operation is not zero
bit 1	DC: Digit Carry/Borrow bit ⁽¹⁾ For ADDWF, ADDLW, SUBLW and SUBWF instructions: 1 = A carry-out from the 4th low-order bit of the result occurred 0 = No carry-out from the 4th low-order bit of the result
bit 0	C: Carry/Borrow bit ⁽²⁾ For ADDWF, ADDLW, SUBLW and SUBWF instructions: 1 = A carry-out from the Most Significant bit of the result occurred 0 = No carry-out from the Most Significant bit of the result occurred

- Note 1:** For Borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRE, RLF) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.
- 2:** For Borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRE, RLF) instructions, this bit is loaded with either the high or low-order bit of the source register.

25

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

- CPFSEQ → f = Wreg
 CPFS LT → f < Wreg
 CPFS GT → f > Wreg



En MPASM:
 Usaremos CPFSEQ:
 CPFSEQ [reg]

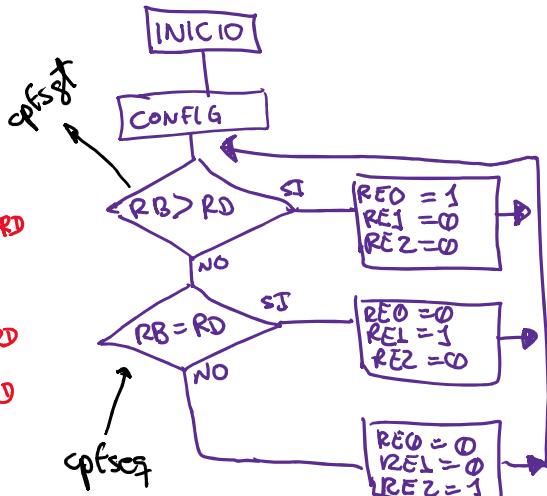
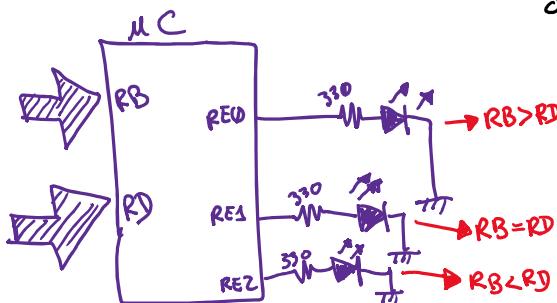
En lenguaje de alto nivel:
 if (x=5){
 [VERDAD]}
 else {
 [Falso]}

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

26

Ejemplo:

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



27

Código MPASM:

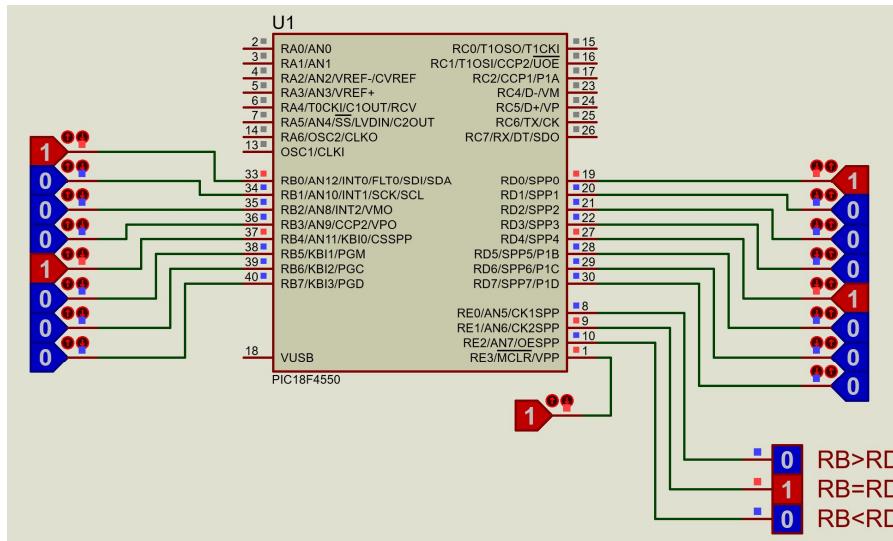
```

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     ;Aqui falta algo que no me acuerdo
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     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

```

28

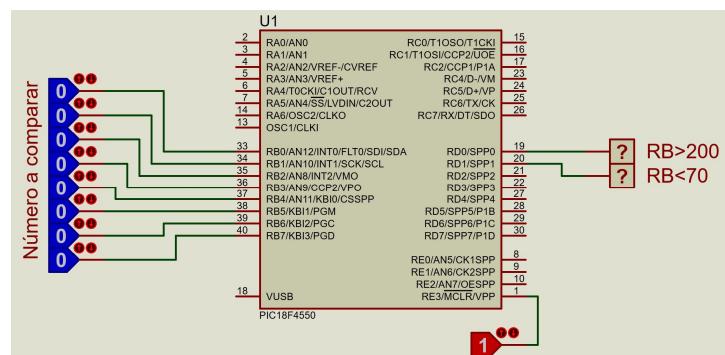
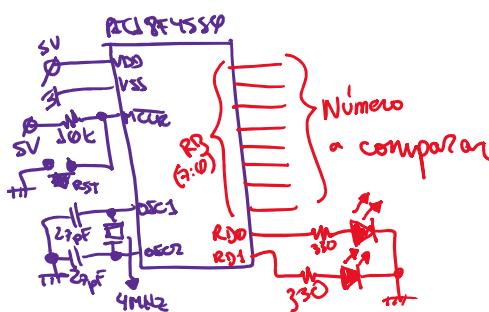
Simulación



29

Ejemplo:

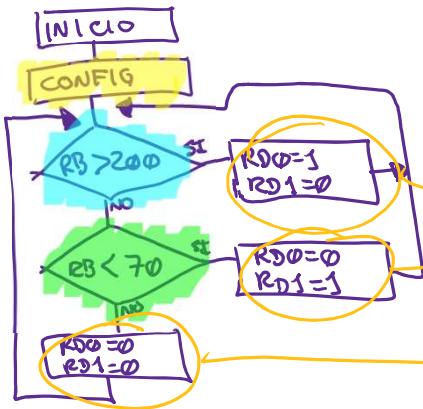
- Desarrollar un programa para que compare lo que se esta ingresando en RB y arroje lo siguiente: RD0=1 cuando RB>200 y RD1=1 cuando RB<70, cuando no se cumplan las dos condiciones las dos salidas permanecerán en cero.



30

Cont.

Diagrama de flujo:



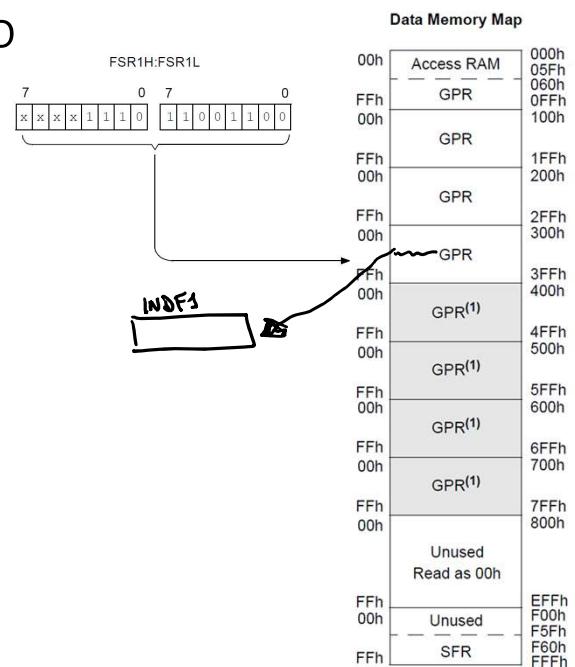
```

1 ;Este es un comentario, se le asigna una linea de numero
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     cpfsqt PORTB
28     goto next1
29     bcf LATD, 0
30     bcf LATD, 1
31     goto loop
32 next1:
33     movlw .70
34     cpfsqt 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
  
```

31

Memoria de datos: Acceso con punteros FSRx/INDFx

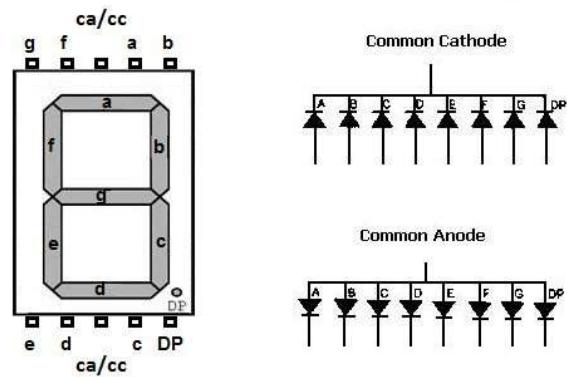
- En la memoria de datos se encuentra mapeado los 2Kbyte de RAM (0x000 – 0x7FF) y los S.F.R. (0xF60 – 0xFFFF)
- Se tienen tres punteros:
 - FSR0 / INDF0
 - FSR1 / INDF1
 - FSR2 / INDF2
- Al igual que TBLPTR, en FSRx se coloca la dirección de la celda a apuntar y en IDFx se opera su contenido (lectura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



32

El display de siete segmentos

- Dos tipos: ánodo común y cátodo común
- Tablas de decodificación diferentes entre ellos.



33

Fin de la sesión

34