

Microcontroladores Laboratorio

Semana 3

Semestre 2021-2

Profesor: Kalun José Lau Gan

1

Preguntas previas:

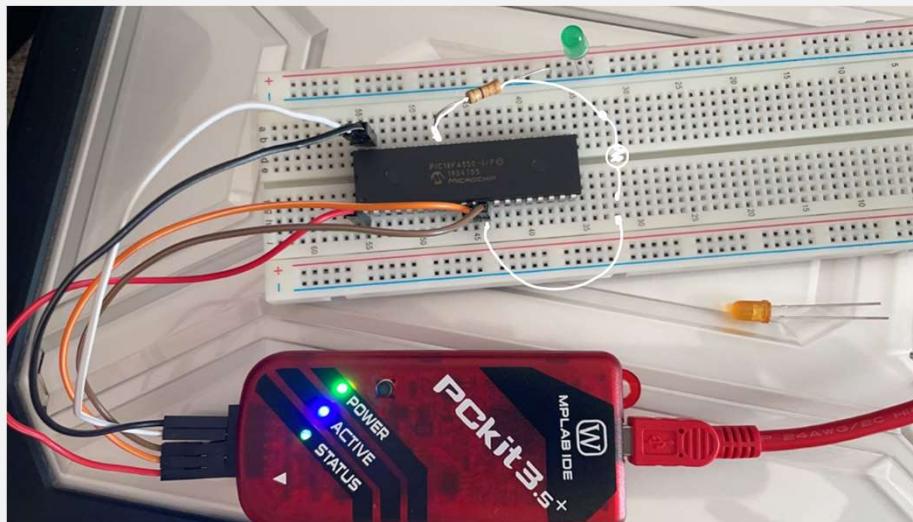
- ¿Cuáles eran los punteros?
 - Para la memoria de datos: FSR0, FSR1, FSR2
 - Para la memoria de programa: TBLPTR
- Ayer se mencionó que faltaba una configuración para que funcionaran las salidas en RE. ¿Qué era lo que faltaba?
 - El puerto E (RE2:RE0) por defecto son entradas analógicas por lo que hay que configurarlas como digital haciendo lo siguiente:

*movlw 0X0F
movwf ADCON1*

- Las evaluaciones serán grupales o individuales?
 - Eso se decidirá al inicio de cada evaluación

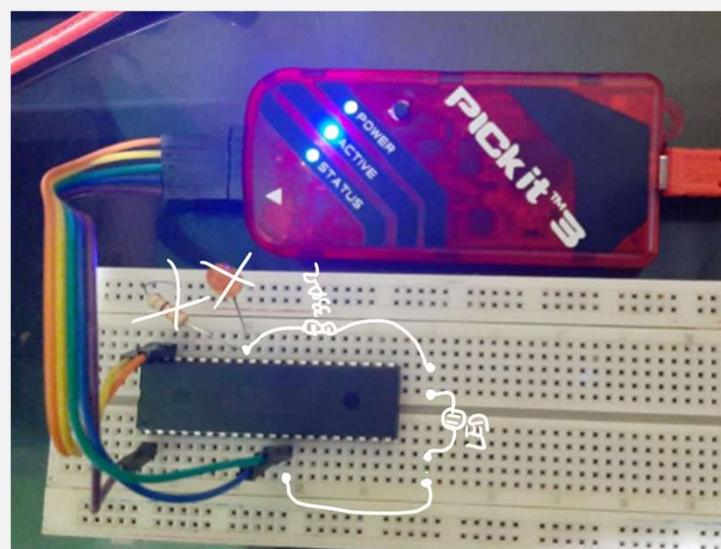
2

Errores cometidos en la implementación de circuitos en el breadboard:



3

Errores cometidos en la implementación de circuitos en el breadboard:



4

Agenda:

- Instrucciones CPFSEQ, CPFSLT, CPFSGT en XC8 PIC ASM
- Interface a display de siete segmentos
- Ejemplos relacionados con LUT en PC y TBLPTR
- Multiplexación en displays de siete segmentos

5

Registro STATUS

- Ref. página 73 de la hoja técnica del microcontrolador PIC18F4550
- 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							

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

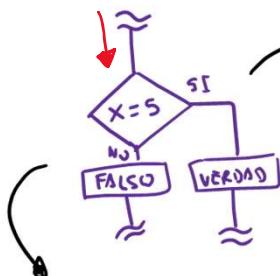
- Note 1:** For Borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.
- 2:** For Borrow, the polarity is reversed. A subtraction is executed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

6

Instrucciones de comparación numérica (CPFSEQ, CPFSLT, CPFSGT)

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

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



En lenguaje de alto nivel:

```

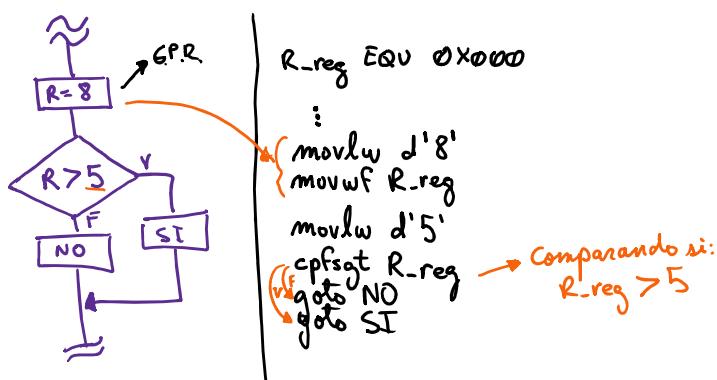
if (x = 5) {
    [VERDAD]
} else {
    [Falso]
}
  
```

XC8 ASM:
 En MPASM:
 Usaremos CPFSEQ:
 CPFSEQ [reg]

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

7

Ejemplo: Pasar a XC8 ASM el siguiente diagrama de flujo:



Nota: En las instrucciones cpfs-- el segundo parámetro de la comparación debe de estar en Wreg

8

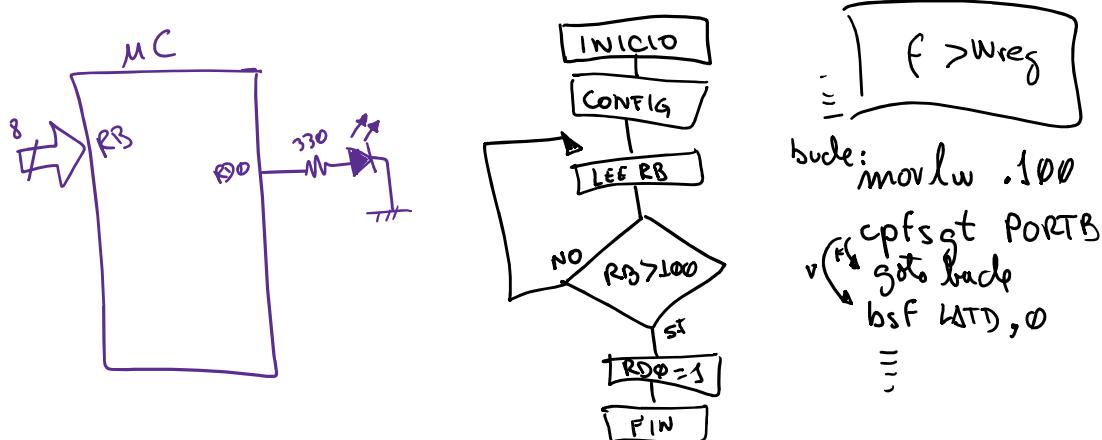
Instrucciones con toma de decisión:



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

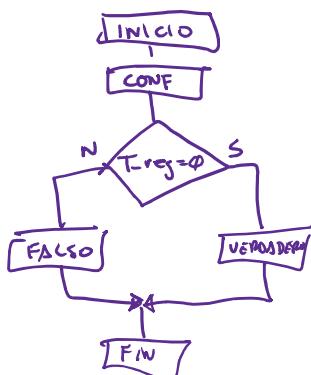
9

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



10

Ejemplo: Pregunta si T_reg (GPR) es igual a cero, si es cierto va a etiqueta verdadero, si no es cierto va a etiqueta falso



Opción 1:

inicio: movlw .0
cpfseq T-reg
vF goto FALSO
vF goto VERDADERO

Opción 2:

inicio: tstfsz T-reg
vF goto FALSO
vF goto VERDADERO

Opción 3:

inicio: movf T-reg, W
sublw .0
btfsz STATUS, Z
vF goto FALSO
vF goto VERDADERO

Opción 4:

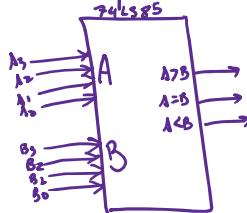
inicio: movf T-reg, W
sublw .0
b3 VERDADERO
FALSO: =
VERDADERO: =

11

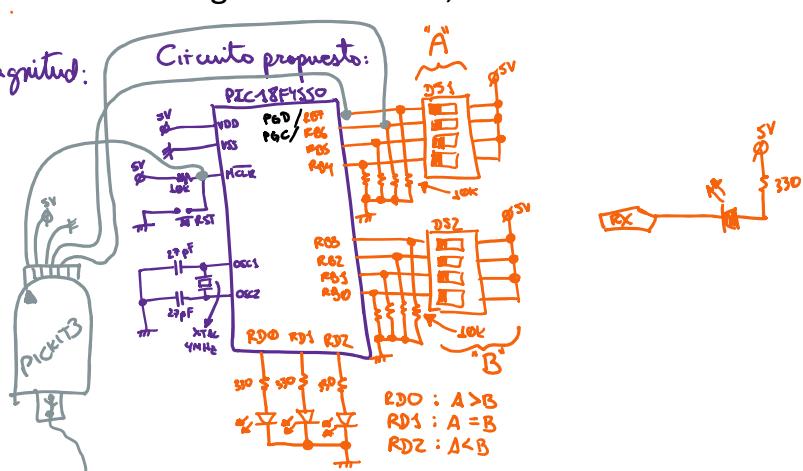
Ejemplo

- Desarrollar un comparador de magnitud de 4 bits, de funcionamiento similar al 74LS85

Un comparador de magnitud:



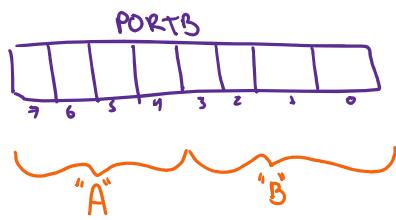
Nota: El número A y el número B ambos estan ingresando por el Puerto B:
-Número A: RB4:RB7
-Número B: RB0:RB3



Nota: Al momento de programar deben de liberar RB6 y RB7

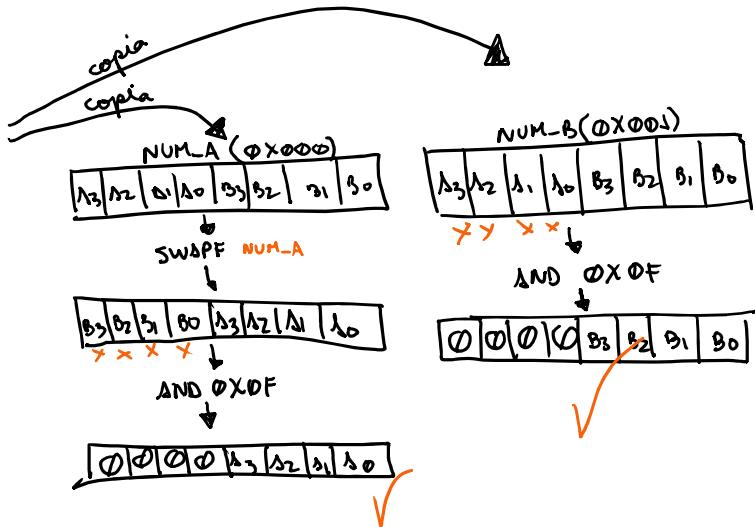
12

Algoritmo:



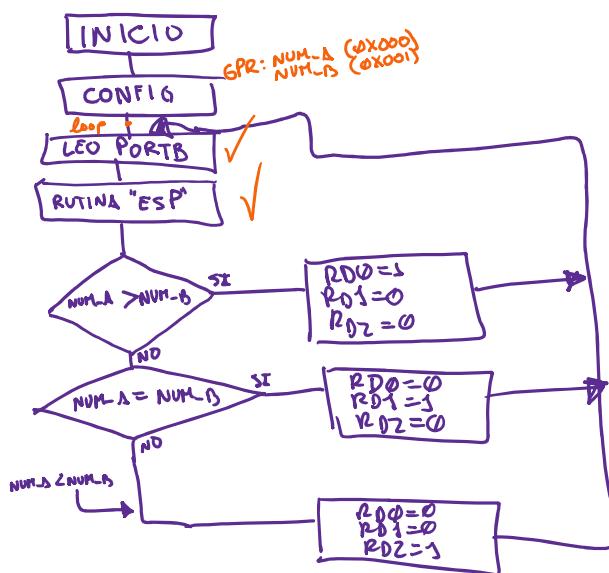
Rutina
"ESP"

GPRs:
NUM_A
NUM_B



13

Algoritmo (cont.)

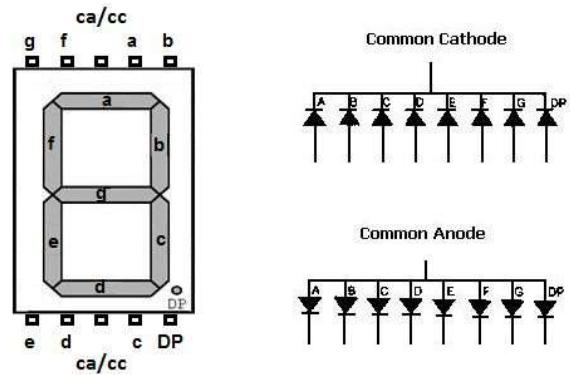


- Asignación:
Desarrollar el programa en XC8 ASM a partir del diagrama de flujo mostrado

14

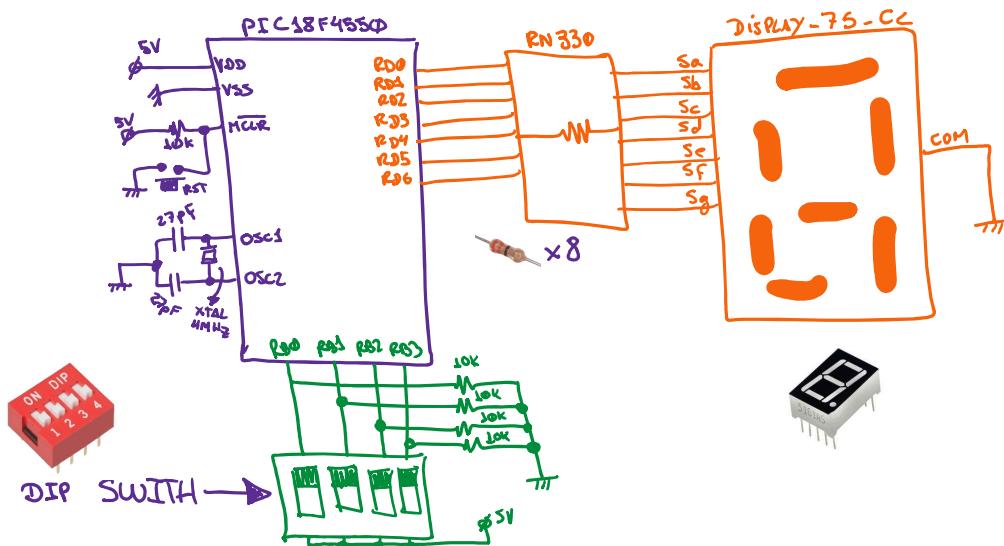
El display de siete segmentos

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



15

Ejercicio: Circuito base para los ejemplos de la sesión.



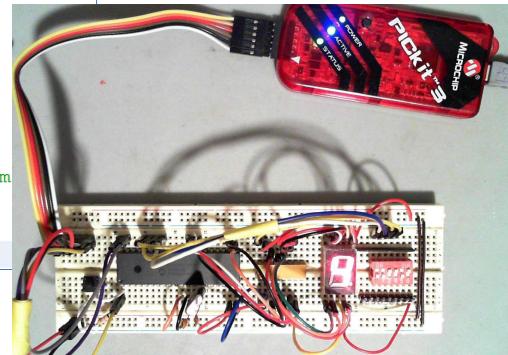
16

Ejemplo: Visualizar el dígito “4” en el display:

```

1 ;Programa para visualizar el digito 4 en el display de 7 segmentos
2      PROCESSOR 18F4550
3      #include "cabecera.inc"
4
5      PSECT rstVect,class=CODE,reloc=2
6      ORG 0000H
7      rstVect:    goto configuracion
8
9      ORG 0020H
10     configuracion: movlw 80H
11          movwf TRISD      ;RD6:RD0 como salidas
12     inicio:       movlw 01100110B ;dígito 4 en el disp 7 segm
13          movwf LATD
14
15      end rstVect

```



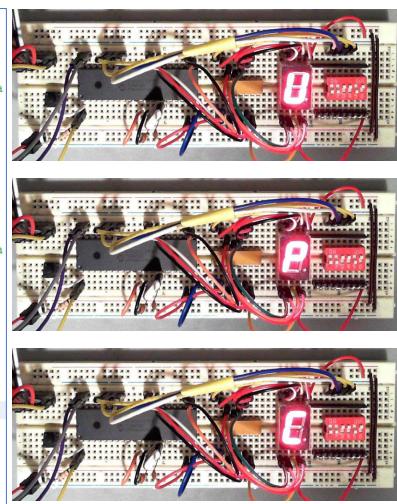
17

Ejemplo: Visualización de “UPC” a razón de una letra cada un periodo de retardo:

```

1 ;Programa para visualizar "UPC" en el display de 7 segmentos
2      PROCESSOR 18F4550
3      #include "cabecera.inc"
4
5      PSECT rstVect,class=CODE,reloc=2
6
7      ;Declaración de nombre en GPR
8      var_i EQU 0x000
9      var_j EQU 0x001
10     var_k EQU 0x002
11
12     ORG 0000H
13     rstVect:    goto configuracion
14
15     ORG 0020H
16     configuracion: movlw 80H
17          movwf TRISD      ;RD6:RD0 como salidas
18     inicio:       movlw 0011110B ;letra U en el disp 7 segm
19          movwf LATD
20          call repeticua
21          movlw 0111001B ;letra P en el disp 7 segm
22          movwf LATD
23          call repeticua
24          movlw 00111001B ;letra C en el disp 7 segm
25          movwf LATD
26          call repeticua
27          goto inicio
28
29      repeticua:
30          movlw 80
31          movwf var_i, b
32          otro1:
33              call bucle1      ;Salto a subrutina
34              decfsz var_i, 1, 0
35              goto otro1
36              return
37          bucle1:
38              movlw 80
39              movwf var_j, b
40          otro2:
41              call bucle2      ;Salto a subrutina
42              decfsz var_j, 1, 0
43              goto otro2
44              return
45          bucle2:
46              movlw 20
47              movwf var_k, b
48          otro3:
49              nop
50              decfsz var_k, 1, 0
51              goto otro3
52              return
53
54      end rstVect

```



18

Ejemplo: Visualizar “ingeniería” a razón de una letra a la vez en el display de siete segmentos:

```

1  ;Programa para visualizar "Ingenieria" en el display de 7 segmentos con TBLPTR
2  PROCESSOR 18F4550
3  #include "cabecera.inc"
4
5  PSECT rstVect,class=CODE,x reloc=2
6
7  PSECT data
8  ORG 0500H
9  cadena: DB 30H, 54H, 6FH, 7BH, 54H, 10H, 7BH, 50H, 10H, 5FH
10
11 ;Declaración de nombre en GPR (para la subrutina repeticua)
12 var_i EQU 000H
13 var_j EQU 001H
14 var_k EQU 002H
15 cuenta EQU 003H
16
17 PSECT code
18 ORG 0000H
19 rstVect: goto configuracion
20
21 ORG 0020H
22 configuracion: movlw 80H
23         movwf TRISD ;RD6:RD0 como salidas
24 inicio:   movlw HIGH cadena
25         movwf TBLPTRH
26         movlw LOW cadena
27         movwf TBLPTRL ;Asignamos dirección a TBLPTR (500H)
28         clrf cuenta ;Cuenta a cero
29 loop:    movlw 10 ;cuenta es la cantidad de caracteres a visualizar
30         cpfseq cuenta ;Preguntamos si cuenta = 3
31         goto aunno ;Falso (cuenta diferente de 3)
32         goto inicio ;Verdadero (cuenta llego a 3)
33 aunno:   TBLRD+
34         movff TABLAT, LATD ;Mueve contenido de TABLAT hacia RD
35         call repeticua ;Llamada a subrutina de retardo
36         incf cuenta, f ;Incremento de cuenta
37         goto loop

```

Nota: En este ejemplo se está empleando un sector de la memoria de programa para almacenar constantes y el TBLPTR para acceder a ellas.

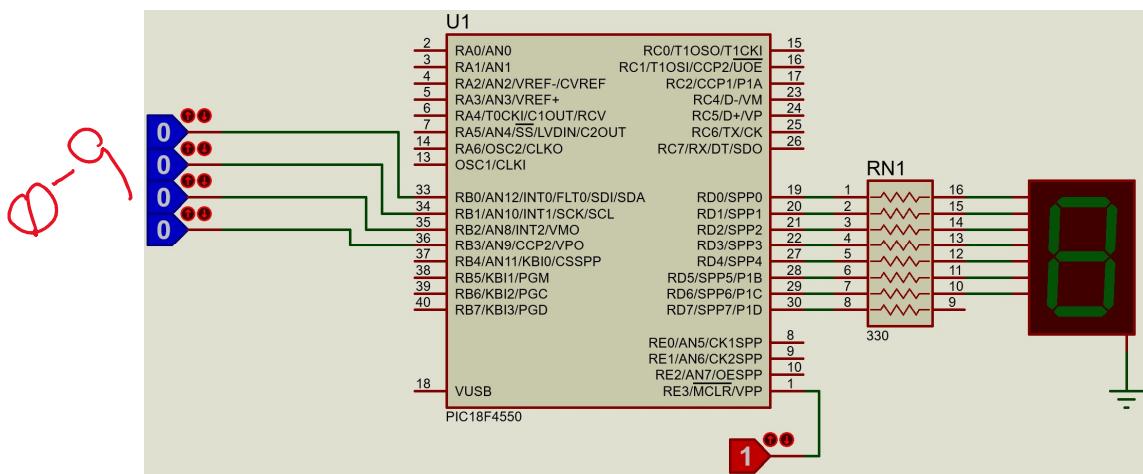
```

39  repeticua:
40      movlw 80
41      movwf var_i, b
42  otro1:
43      call bucle1 ;Salto a subrutina
44      decfsz var_i, l, 0
45      goto otro1
46      return
47
48  bucle1:
49      movlw 80
50      movwf var_j, b
51  otro2:
52      call bucle2 ;Salto a subrutina
53      decfsz var_j, l, 0
54      goto otro2
55      return
56
57  bucle2:
58      movlw 20
59      movwf var_k, b
60  otro3:
61      nop
62      decfsz var_k, l, 0
63      goto otro3
64      return
65
66 end rstVect

```

19

Ejemplo: Decodificador BCD-7Segmentos empleando tablas con PC



20

Código en MPASM

Diagrama de flujo:



```

1 ;Este es un comentario, se le ant
2     list p=18f4550      ;Modelo d
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 temp EQU 0x000          ;posición
18 org 0x0000
19 goto init_conf
20
21 org 0x0020
22 init_conf:
23     clrf TRISD
24
25 loop:
26     movf PORTB, W
27     andlw 0x0F           ;En
28     mullw .2             ;Mu
29     movf PRODL, W
30     call tabla_PC
31     movwf LATD
32     goto loop
33
34 tabla_PC:
35     addwf PCL, f
36     retlw 0xBF            ;0
37     retlw 0x86            ;1
38     retlw 0xDB            ;2
39     retlw 0xCF            ;3
40     retlw 0xE6            ;4
41     retlw 0xED            ;5
42     retlw 0xFD            ;6
43     retlw 0x87            ;7
44     retlw 0xFF            ;8
45     retlw 0xE7            ;9
46
47 end

```

21

Código en XC8 PIC Assembler

- “entrada” es una etiqueta mapeada en la RAM pero no sabemos en qué lugar lo alojara el ensamblador, funciona de manera similar que lo siguiente: “entrada EQU 0x000”

```

1  PROCESSOR 18F4550
2  #include "cabecera.inc"
3
4  PSECT udata_acs
5  entrada:
6      DS 1
7
8  PSECT VectRST, class=CODE, reloc=2, abs
9      ORG 00000H          ;Vector de reset
10 VectRST:
11     goto configuracion
12     ORG 00020H          ;Zona de programa
13 configuracion: movlw 80H
14         movwf TRISD        ;RD0 al RD6 como salidas
15 inicio:   movf PORTB, w       ;Leemos RB y almacenamos el contenido en Wreg
16         andlw 0FH           ;Enmascaramiento de Wreg con 0FH
17         movwf entrada        ;GPR
18         addwf entrada, w     ;Sumamos entrada+entrada y lo alojamos en Wreg
19         call alternativo_PC  ;Llamada a la LUT basado en retlw
20         movwf LATD          ;Mueve contenido de Wreg hacia RD
21         goto inicio         ;Salta a inicio

```

```

23 alternativo_PC: addwf PCL, f      ;LUT empleando PC 1
24     retlw 3FH             ;que contiene los datos del
25     retlw 06H              ;;decodificador BCD-7Seg
26     retlw 5BH
27     retlw 4FH
28     retlw 66H
29     retlw 6DH
30     retlw 7DH
31     retlw 07H
32     retlw 7FH
33     retlw 67H
34     retlw 79H
35     retlw 79H
36     retlw 79H
37     retlw 79H
38     retlw 79H
39     retlw 79H
40
41 end VectRST

```

- Se ha utilizado el flag “abs” en el PSECT para que funcione el direccionamiento absoluto y pueda ejecutarse sin problema las instrucciones retlw de la LUT

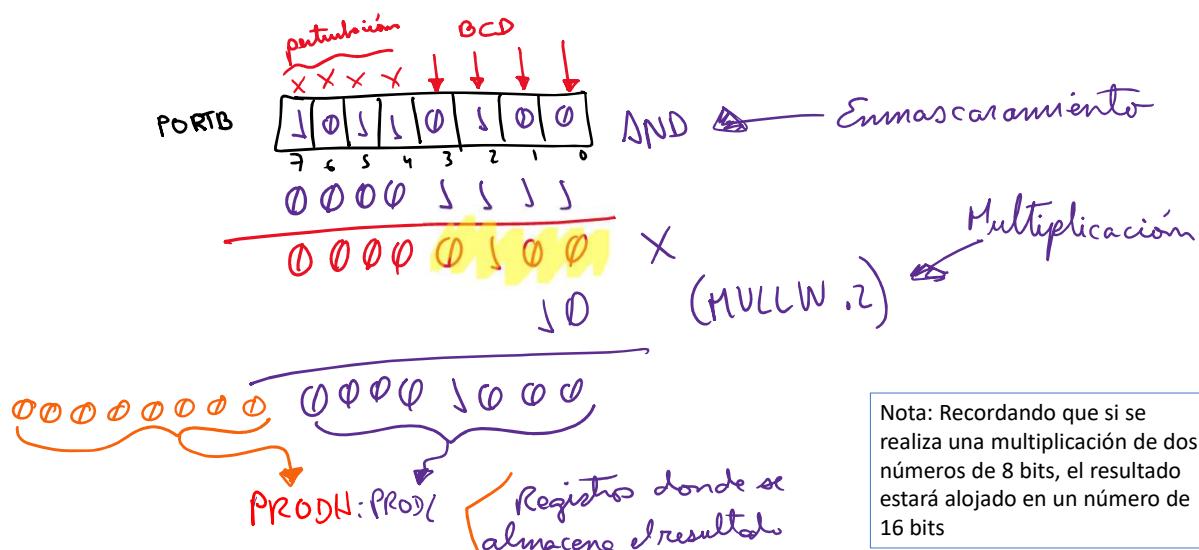
22

Cuestionario referente al ejemplo anterior

- ¿Cuál es la función de PC en este ejemplo?
- ¿Cuál es la diferencia entre RETURN y RETLW?
- ¿Por qué se tuvo que multiplicar x2 el valor recibido en el PORTB para que funcione correctamente la función del decodificador?

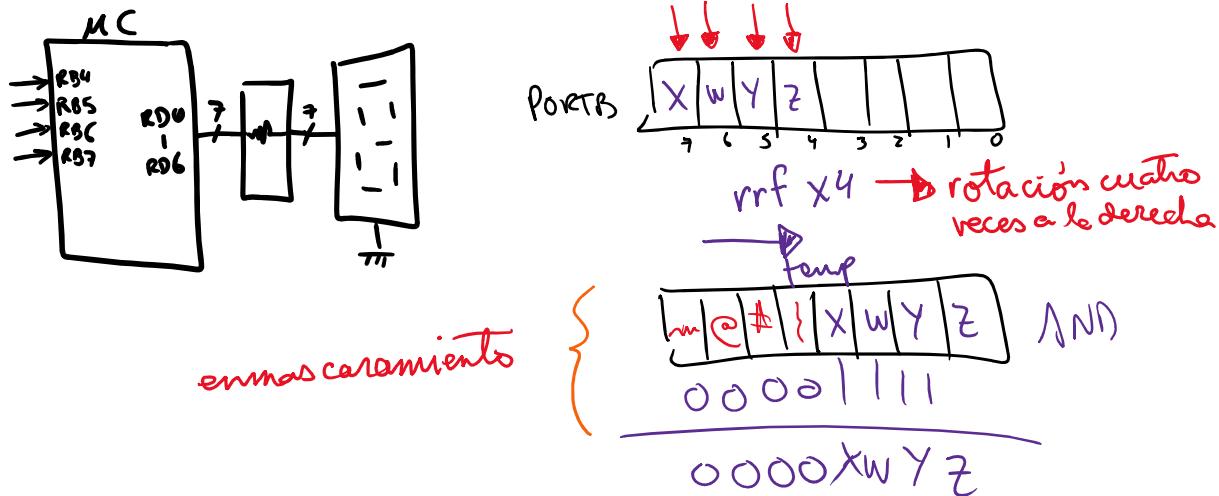
23

Operación de enmascaramiento de bits y operación de multiplicación empleando “mullw”



24

¿Qué pasa si uso RB7:RB4 como entradas al decodificador BCD-7S?



25

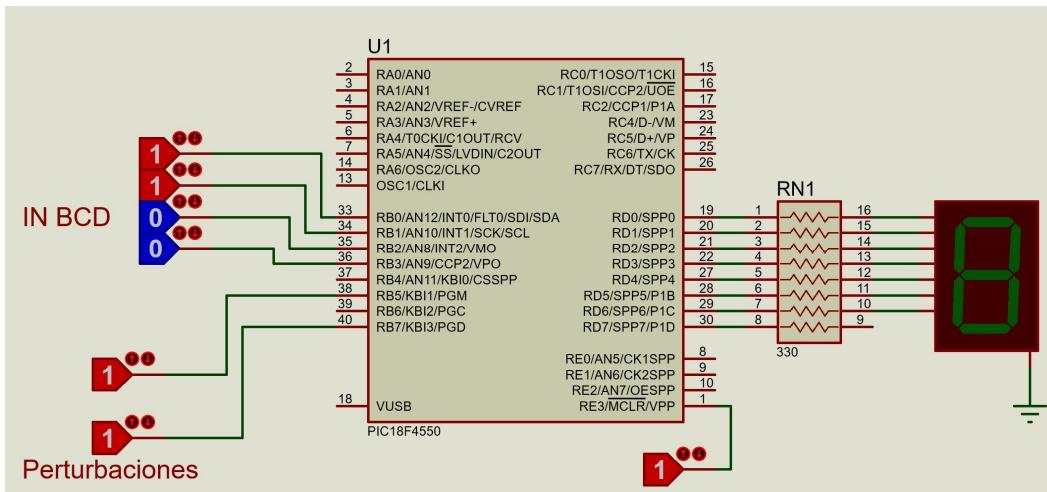
Cómo se están grabando las instrucciones en la memoria de programa:

Memoria de Programa	
35	addwf PCL, f
36	retlw 0xBF ;0
37	retlw 0x86 ;1
38	retlw 0xDB ;2
39	retlw 0xCF ;3
40	retlw 0xE6 ;4
41	retlw 0xED ;5
42	retlw 0xFD ;6
43	retlw 0x87 ;7
44	retlw 0xFF ;8
45	retlw 0xE7 ;9

Nota: Hay que recordar que las instrucciones ocupan dos bytes en la memoria de programa

26

Ejemplo: Decodificador BCD-7Segmentos empleando tablas con TBLPTR



27

Código en MPASM

```

1 ;Este es un comentario, se le antecede un punto y
2     list p=18f4550      ;Modelo del microcontrolador
3     #include <p18f4550.inc>    ;Llamada a la librería
4
5 ;Directivas de preprocesador o bits de configuración
6 CONFIG PLLEDIV = 1           ; PLL Prescaler
7 CONFIG CPUDIV = OSC1_PLL2   ; System Clock Prescaler
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 bit (On)
13 CONFIG PBADEN = OFF         ; PORTB A/D Enable
14 CONFIG MCILRE = ON          ; MCLR Pin Enable
15 CONFIG LVP = OFF            ; Single-Supply ICSP Enable bit (Single-supply)
16
17 org 0x0000
18 goto init_conf
19
20 org 0x0300
21 tabla_7s db 0xBF, 0x86, 0xDB, 0xCF, 0xE6, 0xED, 0xFD, 0x87, 0xFF, 0xE7
22
23     org 0x0020
24     init_conf:
25         clrf TRISD
26         movlw HIGH tabla_7s
27         movwf TBLPTRH
28         movlw LOW tabla_7s
29         movwf TBLPTRL
30
31     loop:
32         movf PORTB, W
33         andlw 0x0F           ;Enmascara los cuatro primeros bits
34         movwf TBLPTRL
35         TBLRD*
36         movff TABLAT, LATD
37         goto loop
38
39     end

```

28

Código en XC8 PIC Assembler:

```

1  PROCESSOR 18F4550
2  #include "cabecera.inc"
3
4  PSECT rstVect, class=CODE, reloc=2, abs
5
6  ORG 0500H
7  cadena: DB 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x79, 0x79, 0x79, 0x79, 0x79
8
9  ORG 0000H
10 rstVect: goto configuracion
11
12 ORG 0020H
13 configuracion: movlw 80H
14     movwf TRISD      ;RD6:RD0 como salidas
15     clrf TBLPTRU
16     movlw HIGH cadena
17     movwf TBLPTRH
18     movlw LOW cadena
19     movwf TBLPTRL    ;Asignamos dirección a TBLPTR (500H)
20 inicio:   movf PORTE, w
21     andlw 0FH
22     movwf TBLPTRL
23     TBLRD*
24     movff TABLAT, LATD
25     goto inicio
26
27 end rstVect

```

Nota: Para evitar crear muchos PSECTs pudiendo ocasionar conflictos entre ellos es que se está colocando el flag "abs" en el PSECT único del código

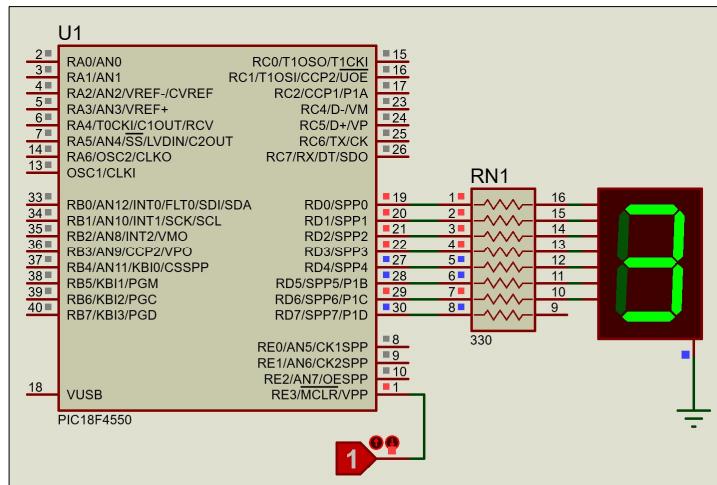
29

Cuestionario referente al ejemplo anterior:

- ¿Cuál es la función de la instrucción andlw 0x0F?
- ¿Qué es lo que hace TBLPTR en este ejemplo?
- ¿De qué trata la instrucción movff y cuántos ciclos demora?
- ¿Qué pasaría si se coloca un número superior al formato BCD?
- Modificar el decodificador para que de entrada sea binario en lugar de BCD y se muestre en el display de siete segmentos hasta el valor F
- Si en lugar de usar display de cátodo común se usa del tipo ánodo común. ¿Qué es lo que se modificaría en el código y hardware?

30

Ejemplo: Contador autoincremental 0-9



31

Ejemplo: Contador autoincremental 0-9 en MPASM

```

1 ;Este es un comentario, se le antecede un punto y coma
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 (# MHz oscillator input divide)
7 CONFIG CPUDIV = OSC1_PLL2 ; System Clock Postscaler Selection bits ([Primary Oscillator Selection bits (XT oscillator (XT))
8 CONFIG FOSC = XT_XT    ; Oscillator Selection bits (XT oscillator (XT))
9 CONFIG FWRT = ON        ; Power-up Timer Enable bit (FWRT enabled)
10 CONFIG BOR = OFF       ; Brown-out Reset Enable bits (Brown-out Reset disabled in software)
11 CONFIG WDT = OFF       ; Watchdog Timer Enable bit (WDT disabled (control is placed on the user timer instead))
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 inputs)
14 CONFIG MCLE = ON       ; MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)
15 CONFIG IVP = OFF       ; Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
16
17 cblock 0x000
18 var_i
19 var_j
20 var_k
21 endc
22
23 org 0x0000
24 goto init_conf
25
26 ;Lo de líneas se almacena de manera secuencial desde la 0x0300 en la mem prog
27 org 0x0500
28 tabla_7s db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x79, 0x79, 0x79, 0x79, 0x79
29
30 org 0x0020
31 init_conf:
32    movlw 0x80
33    movwf TRISD          ;RD6-RD0 como salidas
34    movlw HIGH tabla_7s
35    movwf TBLPTRH
36    movlw LOW tabla_7s
37    movwf TBLPTRL         ;TBLPTR apunta a 0x0300 de la mem de prog.

```

<pre> 39 loop: 40 TBLRD* 41 movff TABLAT, LATD 42 movlw .9 43 cpfseq TBLPTRL 44 goto falso 45 clrf TBLPTRL 46 goto otro 47 48 incf TBLPTRL, f 49 50 call delay_long 51 goto loop </pre>	<pre> 53 ;subrutina de retardo 54 delay_long: 55 movlw .50 56 movwf var_i 57 otrol: 58 call bucle1 59 decfsz var_i,f 60 goto otrol 61 return 62 bucle1: 63 movlw .55 64 movwf var_j 65 otro2: 66 nop 67 nop 68 call bucle2 69 decfsz var_j,f 70 goto otro2 71 return 72 bucle2: 73 movlw .20 74 movwf var_k 75 otro3: 76 nop 77 decfsz var_k,f 78 goto otro3 79 return 80 end </pre>
--	--

32

Ejemplo: Contador autoincremental de 0 al 9 en el display de 7 segmentos con XC8 ASM:

```

2      PROCESSOR 18F4550
3      #include "cabecera.inc"
4
5      PSECT rstVect,class=CODE,reloc=2
6
7      PSECT data
8      ORG 0500H
9      cadena: DB 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x79, 0x79, 0x79, 0x79, 0x79, 0x79
10
11     ;Declaración de nombre en GPR (para la subrutina repetitiva)
12     var_i EQU 000H
13     var_j EQU 001H
14     var_k EQU 002H
15     cuenta EQU 003H
16
17     PSECT code
18     ORG 0000H
19     rstVect: goto configuracion
20
21     ORG 0020H
22     configuracion: movlw 80H
23             movwf TRI5D ;RD6:RD0 como salidas
24             movlw HIGH cadena
25             movwf TBLPTRH
26             movlw LOW cadena
27             movwf TBLPTRL ;Asignamos dirección a TBLPTR (500H)
28             clrf cuenta ;Cuenta a cero
29             movlw 10 ;cuenta es la cantidad de caracteres a visualizar
30             cpfsq cuenta ;Preguntamos si cuenta = 3
31             goto aunno ;Falso (cuenta diferente de 3)
32             goto inicio ;Verdadero (cuenta llegó a 3)
33             TBLRD*+
34             movff TABLAT, LATD ;Lectura de contenido apuntado por TBLPTR y posterior incremento de la
35             call repetitiva ;Mueve contenido de TABLAT hacia RD
36             incf cuenta, f ;Llamada a subrutina de retardo
37             incf cuenta ;Incremento de cuenta
38             goto loop

```

Nota: Al igual que el ejemplo hecho para visualizar la palabra “ingeniería”, se ha cambiado solamente el contenido de los datos en el sector 500H de la memoria de programa

```

39     repetitua:
40             movlw 80
41             mowwf var_i, b
42             otro1:
43             call bucle1 ;Salto a subrutina
44             decfsz var_i, 1, 0
45             goto otro1
46             return
47
48             bucle1:
49             movlw 80
50             mowwf var_j, b
51             otro2:
52             call bucle2 ;Salto a subrutina
53             decfsz var_j, 1, 0
54             goto otro2
55             return
56
57             bucle2:
58             movlw 20
59             mowwf var_k, b
60             otro3:
61             nop
62             decfsz var_k, 1, 0
63             goto otro3
64             return
65
66     end rstVect

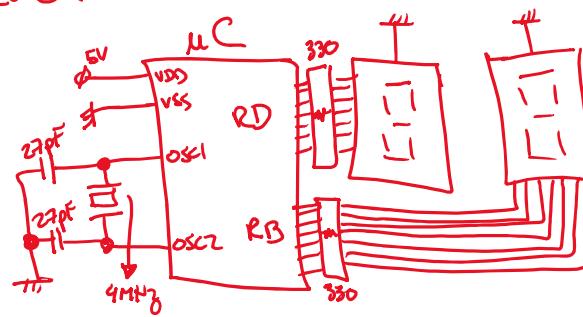
```

33

Asignación:

- Realizar un contador 00-99 autoincremental con periodo de incremento de 500ms aproximadamente, el dígito unidad será emitido por el puerto RB y el dígito decena será emitido por el puerto RD

Idea:



34

Fin de la sesión