UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Práctica 1:Introducción a la programación del microcontrolador PIC16F877; "Direccionamiento Directo"

INTEGRANTES DE EQUIPO Y GRUPO DE TEORÍA

López Becerra Ricardo - 420053710 - GRUPO 3
Navarrete Zamora Aldo Yael - 317242409 - GRUPO 4

ASIGNATURA

Laboratorio de Microcomputadoras

GRUPO DE LABORATORIO

8

FECHA DE REALIZACIÓN

23 de Febrero del 2023

FECHA DE ENTREGA

13 de Marzo del 2023

1. Desarrollo

Para cada uno de los siguientes ejercicios, realizar los programas solicitados y simular el funcionamiento de ellos.

1.- Siguiendo las indicaciones previas, escribir el siguiente programa, ensamblar y simular el funcionamiento de este.

```
        PROCESSOR 16f877
        INCLUDE <p16f877.inc>

K       equ     H'26'
L       equ     H'27'

        ORG  0
        GOTO INICIO

        ORG 5
INICIO: MOVLW H'05'
        ADDWF K,0
        MOVWF L
        GOTO INICIO
        END
```

Algoritmo:

El programa previo es un programa simple, el cuál mueve un valor de H'05' al registro de trabajo, que posteriormente se sumará con el valor que tenga el registro K (en H'26') y lo guardará en el registro W.

Una vez que el resultado se almacene en W, moveremos dicho resultado al registro L (en este caso al registro H'27').

Código comentado:

```
        PROCESSOR 161877
        INCLUDE <p16f877.inc>
K equ H'26'; Registro para operando
L equ H'27'; Registro para resultado
        ORG 0
        GOTO INICIO
        ORG 5
INICIO: MOVLW H'05'; Moviendo un 5h a W
        ADDWF K,0; Suma W con K y guarda en W
        MOVWF L ;Mover W a L
        GOTO INICIO ; Repetir
        END
```

2.- Escribir, ensamblar y ejecutar el siguiente programa:

```
        PROCESSOR 16f877
        INCLUDE <p16f877.inc>

K       equ     H'26'
L       equ     H'27'
M       equ     H'28'

                ORG  0
                GOTO INICIO

                ORG 5
INICIO:         MOVF K,W
                ADDWF  L,0
                MOVWF  M
                GOTO INICIO
                END
```

a. Comentar e indicar que hace el programa

Este programa tiene un funcionamiento muy parecido al anterior, primero cargamos un registro en W, en este caso, el valor que tiene almacenado K (que vale H'26') lo moverémos a W. Una vez en W, haremos la suma con L (H'27'), y el resultado lo almacenademos en W. Finalmente, lo que almacena W se moverá al registro M, ahí estará el resultado de la suma con los valores que pongamos manualmente en los registros. b. Realizar la ejecución con diferentes valores en K y L

Como se menciona anteriormente, los valores en K y en L se sumarán y almacenarán en el registro M, cabe recalcar que cuando hay un desbordamiento de bits (esto es que el número sobrepasa los 8 bits que puede soportar un registro) entonces no se puede representar dicho número.

c. Revisar el valor que se genera en la bandera C

Cuando hacemos la suma entre dos valores que crean un 'Carry' o un Acarreo esta bandera del STATUS se prende, es decir, se pone en un valor binario de 1.

Código comentado:

```
    PROCESSOR 16f877
    INCLUDE <p16f877.inc>
K equ H'26'; Registro para primer operando
L equ H'27'; Registro para segundo operando
M equ H'28'; Registro de resultado
        ORG 0
        GOTO INICIO
        ORG 5
INICIO: MOVF K.W ; Movemos el valor del registro K a W
        ADDWF L,0; Operamos adición de L y W, guardando en W
        MOVWF M; Movemos lo que hay en W a M
        GOTO INICIO; Repetimos
        END
```

3.- Modificar el programa anterior, para que ahora los datos que operará se encuentren en las localidades J y K respectivamente, el resultado almacenarlo en otras direcciones, reservadas para C1 y R1; C1 representa el valor de la bandera de acarreo y R1 el resultado.

Algoritmo:

Como lo menciona el enunciado, debemos definir un registro extra llamado 'C1' y 'R1' que almacenarán el resultado de la operación y el carry.

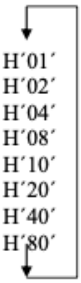
Para manejar este problema, al momento de hacer la operación de adición debemos checar de nuevo el registro STATUS, en su bandera C. Con la instrucción BTFSS mencionando el registro y número de bit a evaluar checamos si el carry está en alto, de no estarlo, ese registro se mantendrá en cero, en caso contrario, lo pondremos en 1.

Código comentado:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
J equ H'26' ;Registro para segundo operando
K equ H'27'; Registro para primer operando
R1 equ H'29'; Registro para resultado
C1 equ H'28'; Registro para el carry
STATUS equ H'03' ;Bandera STATUS
        ORG 0
        GOTO INICIO
        ORG 5
INICIO: MOVF K,W ;Mover valor de K a W
        ADDWF J,0 ;Sumar W con J y guarda en W
        MOVWF R1; Mover lo que hay en W a R1 (resultado)
        BTFSC STATUS, 0 ;Checar bit 0 de STATUS, (carry)
        GOTO C_1 ;Si es 1, entonces va a C1
        GOTO C_0 ;En caso contrario, entonces va a C0
    C_0:
        MOVLW H'00' ;Se mueve la literal H'00' a W
        MOVWF C1; Mueve W a C1
        GOTO RESUME; Va a RESUME
    C_1:
        MOVLW H'01'; Se mueve la literal H'01' a W
        MOVWF C1; Mueve W a C1
        GOTO RESUME; Va a RESUME
RESUME:
        GOTO INICIO ; Repite
        END
```

4.- Realice un programa que ejecute la siguiente secuencia, misma que deberá ver en la dirección de memoria (registro) de su elección.

Donde H ’01’ indica que el dato esta dado en hexadecimal. En caso de seleccionar el registro cuya dirección es 0X20



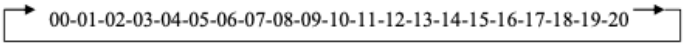
Algoritmo:

Para este programa, observamos que cada vez el número se suma por si mismo, esto equivale a multiplicar por 2 todas las veces, por lo que se nos ocurrió hacer un shifteo hacia la izquierda del número que tengamos almacenado en el registro J, esto lo hacemos hasta que el bit 7 de nuestro número en J esté encendido, esto finalmente nos indicará si el número ha llegado a 80h.

Código Comentado:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
J equ H'26' ;Registro para guardar el resultado.
STATUS equ H'03' ;Registro STATUS
        ORG 0
        GOTO INICIO
        ORG 5
INICIO: MOVLW H'01'; Movemos la literal H'01' a W
        MOVWF J; Movemos W a J
        BCF STATUS, 0 ;Limpiamos a STATUS
LOOP:   RLF J ; Left Shift (Corrimiento a la izquierda de J)
        BTFSC J, 7; Checamos si hemos llegado a 80
        GOTO INICIO; Si es verdadero, regresamos a inicio
        GOTO LOOP; Caso contrario, seguimos iterando.
        END
```

5.- Desarrollar un programa que presente la cuenta en numeración decimal en la localidad de memoria de su elección, como se indica a continuación.



Algoritmo:

La realización de este programa emplea un algoritmo bastante sencillo, definimos una constante que es el número H'07' que será el 'Offset' o salto que daremos cada vez que lleguemos a una novena unidad (09, 19, etc.) Ademas, en un loop (ciclo) se añaden una serie de condicionales, estas condicionales van a checar dentro del programa si:

- El número actual en nuestro contador R1 ha llegado a H'09'
- El número actual en nuestro contador R1 ha llegado a H'19'
- El número actual en nuestro contador R1 ha llegado a H'20'

Y si alguna de las primeras dos se cumple, entonces le sumaremos la constante mencionada en el parrafo de arriba. En caso de que se cumpla la tercera, entonces regresaremos el contador a cero y repetimos.

Código Comentado:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
R1 equ H'26'; CONTADOR
CONST equ H'27'; REGISTRO PARA CONSTANTE
STATUS equ H'03'; REGISTRO STATUS
ORG 0
 GOTO INICIO
 ORG 5
INICIO: MOVLW  H'00' ;MOVEMOS LITERAL H'00' A W
        MOVWF R1 ;MOVEMOS W A R1
        MOVLW  H'07'; MOVEMOS LITERAL H'07' A W
        MOVWF CONST; MOVEMOS W A CONST
LOOP:
        MOVF R1, w
        ; Condicional para checar si el contador es H'09'
        ; En caso verdadero se le suma la constante
        ; En caso falso continuamos
        SUBLW H'09'
        BTFSC STATUS, 2
        GOTO SUMA_CONST
        MOVF R1, w
        ; Condicional para checar si el contador es H'19'
        ; En caso verdadero se le suma la constante
        ; En caso falso continuamos
        SUBLW H'19'
        BTFSC STATUS, 2
        GOTO SUMA_CONST
        MOVF R1, w
        ; Condicional para checar si el contador es H'20'
        ; En caso verdadero vamos a SET_ZERO
        ; En caso falso incrementamos el contador y repetimos el loop.
        SUBLW H'20'
        BTFSC STATUS, 2
        GOTO SET_ZERO
        INCF R1
        GOTO LOOP
SUMA_CONST:
        MOVF CONST, w ;Movemos constante a W
        ADDWF R1; Sumamos W con R1
        GOTO LOOP; Regresamos al LOOP
SET_ZERO:
        MOVLW H'00' ;Movemos constante H'00' a W
        MOVWF R1; Movemos W a R1
        GOTO LOOP ; Continuamos LOOP
 END
```

2. Conclusiones

López Becerra Ricardo

Es esta práctica aprendimos a utilizar las herramientas necesarias para programar el pic 16 y los conceptos relacionadas, como la arquitectura de una computadora, el set de instrucciones y el uso de los registros de propósito general y específico. en esta práctica aprendimos lo básico sobre direccionamiento directo, de tal manera que pudimos usar las direcciones de los registros especificándolas explícitamente. Finalmente, aprendimos algunas de las instrucciones fundamentales del set de instrucciones de la pic 16, como el corrimiento a la izquierda.

Navarrete Zamora Aldo Yael

La primera práctica de microcomputadoras fue una experiencia muy emocionante para nosotros, ya que el objetivo principal era familiarizarnos con el lenguaje ensamblador, los registros de uso general y específico, los bancos de trabajo, el simulador y el conjunto de instrucciones de un microcontrolador. Junto con nuestro grupo, nos asignaron la tarea de escribir y ejecutar programas simples en el MPLAB. La práctica permitió adquirir una comprensión más profunda de los fundamentos de la programación en el lenguaje de nivel más bajo que existe y nos dio la confianza necesaria para trabajar prácticas más complejas en el futuro.

3. Referencias

- Anónimo. (s.f). PIC16F877A Instruction Set. Recuperado el 10 de Marzo del 2023 de https://ww1.microchip.com/downloads/en/DeviceDoc/31029a.pdf