UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Práctica 2: Programación en ensamblador direccionamiento indirecto

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

2 de marzo de 2023

FECHA DE ENTREGA

16 de marzo del 2023

1. Desarrollo

1. Escribir, comentar y ejecutar la simulación del siguiente programa:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
  ORG 0
  GOTO INICIO
  ORG 5
INICIO:
  BCF STATUS,RP1
  BSF STATUS,RP0
  MOVLW 0X20
  MOVWF FSR
LOOP:
  MOVLW 0X5F
  MOVWF INDF
  INCF FSR
  BTFSS FSR,6
  GOTO LOOP
  GOTO \$
  END
```

En este ejercicio se pidió comentar el funcionamiento de código. lo que hace este programa es recorrer las direcciones de memoria desde la 0x20 a la 0x3F colocando el valor 0x5F en todas las direcciones que visita.

Algoritmo

Primero se selecciona el banco de memoria que se quiere utilizar. En este caso, se está utilizando el bando de memoria 1 ya que el bit RP0 del registro STATUS está encendido y el registro RP1 está apagado.

Posteriormente, se carga el valor 0x20 en el registro FSR, el cual funciona como apuntador, por lo que empezaremos apuntando a la dirección 0x20.

Después se entra en un loop. En este loop se escribe el valor 0x5F en el registro INDF, que es el registro apuntado por FSR.

Posteriormente se incrementa el valor del registro FSR y finalmente se verifica si el bit 6 de FSR está encendido. Si este bit está encendido significa que se llegó a la dirección 64 o 0x40, por lo que se finaliza el programa; si el bit 6 no está encendido, se repite el loop.

Programa comentado

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
  ORG 0
  GOTO INICIO
```

```
  ORG 5
INICIO:
  BCF STATUS,RP1
  BSF STATUS,RP0; pone el banco en 01
  MOVLW 0X20 ; empieza en la direccion 20
  MOVWF FSR
LOOP:
  MOVLW 0X5F
  MOVWF INDF ; Coloca 0x5F en el apuntador FSR
  INCF FSR; Incrementa el apuntador FSR
  ;Mientras FSR no tenga el bit 6 encendido,
  ;haremos el loop
  BTFSS FSR,6
  GOTO LOOP
  GOTO \$
  END
```

2. Elaborar un programa que encuentre el número menor, de un conjunto de datos ubicados entre las localidades de memoria 0x20 a 0X3F; mostrar el valor en la dirección 40H.

   La solución de este problema fue similar a la proporcionada para el problema anterior. La idea básica fue recorrer las mismas direcciones y a través de restas se determinaba cual era el mayor.

Algoritmo

El algoritmo implementado fue una búsqueda lineal. Se comienza seleccionando el banco de memoria 1 y colocando el número más grande representable en el controlador en la dirección 0x40. También se coloca la dirección 0x1F en el registro FSR.

Después inicia un loop donde se realiza la resta del contenido del registro actual menos el actual mínimo. Aquí encontramos dos casos:

a) Cuando la resta es 0 o mayor a 0: En este caso el valor de los elementos es igual o el valor mínimo hasta el momento es menor que el valor en el registro actual, por lo que no se debe hacer nada.

b) Cuando la resta es menor a 0: En este caso el valor del registro actual es menor que el mínimo encontrado hasta el momento, por lo que debe actualizarse.

El loop se repite con la misma condición de paro que el ejercicio anterior.

Código comentado

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

MIN equ H'40'
  ORG 0
```

```
    GOTO INICIO
    ORG 5
INICIO:
  MOVLW H'FF'
  MOVWF MIN
   ; pone el banco en 00
  BCF STATUS,RP1
  BCF STATUS,RP0

   ;Coloca el inicio del recorrido una direccion
   ; de memoria antes de la primera que realmente
   ;se quiere tomar en cuenta.
  MOVLW 0X1F
  MOVWF FSR
LOOP:
   ;Incrementa el apuntador FSR
   INCF FSR
   ;Carga el contenido del registro MIN en W y lo
   ;resta al contenido del registro actualmente
   ;apuntado por FSR
   MOVF MIN,W
   SUBWF INDF,0; F - W
   ;Si no hay acarreo, el numero puede ser negativo
   BTFSS STATUS, C
   GOTO CHECK_Z
CONTINUE:
   ;Mientras FSR no tenga el bit 6
   ;encendido, haremos el loop
   BTFSS FSR,6
   GOTO LOOP
   GOTO \$
CHECK_Z:
   ; Si z esta en 0, significa que el numero es negativo;
   ;si es 1, es cero, por lo que no se hace nada
   BTFSC STATUS, Z
   GOTO CONTINUE
   ; Como el resultado de la resta fue negativo,
   ;el valor del registo actual es menor que el
   ;minimo actual, por lo que se actualiza el valor
   ;minimo
   MOVFW INDF
   MOVWF MIN
   GOTO CONTINUE
   END
```

3. Desarrollar el algoritmo y el programa que ordene de manera ascendente un conjunto de datos ubicados en el banco 0 del registro 0X20 al 0X2F.

   Para este programa se tomo como base el programa anterior, pero se le agregaron

varios componentes. El algoritmo de ordenamiento utilizado fue Bubble Sort.

Algoritmo

El algoritmo utilizado fue Bubble Sort por su facilidad de implementación. Para este problema se utilizaron 4 registros que almacenan el valor del registro actual, el valor del registro pasado, el numero de elementos a ordenar y el número de elementos ya ordenados.

Primero se selecciona el banco de memoria 1. Después se carga la dirección de memoria del segundo elemento, la cantidad de elementos a ordenar y se coloca 0 en el número de elementos ordenados.

Luego comienzan dos loops anidados. El exterior se repetirá el número de elementos que tenga el arreglo de números y el interior el número de elementos menos 1. Esto se verificará realizando restas entre el número de veces que queremos repetir los loops y los que llevamos actualmente.

Dentro del segundo loop se verifica si el contenido del registro INDF es menor que el contenido del registro antes que él. Esto se verifica copiando los valores de los registros apuntados por FSR y FSR - 1 a otros registros y comparándolos.

La lógica para determinar cual es mayor que otro es igual a la del ejercicio anterior.

Si el el valor del registro apuntado por FSR es menor que el valor apuntado por FSR - 1, entonces estos se intercambian. Cuando este proceso se realiza para cada uno de los elementos del arreglo, se sale de este loop y se agrega uno al numero de elementos ya ordenados.

Código comentado

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
I equ H'30'; valor adelante
J equ H'31'; valor atras
N equ H'32'; Numero de elementos a ordenar
CNT equ H'38'; Numero de iteraciones completadas
  ORG 0
  GOTO INICIO
  ORG 5
INICIO:
;Selcciona el bando de moemoria 0
  BCF STATUS,RP1
  BCF STATUS,RP0; BANCO 0
  MOVLW 0X21
  MOVWF FSR ; Inicia en la direccion de memoria 21
  ; Carga la cantidad de numeros a ordenar
  MOVLW H'0f'
  MOVWF N
  ; Carga el nuero de pasadas realizadas
  MOVLW H'00'
```

```
  MOVWF CNT
LOOP0:
  ;Verifica si se llego al limite de pasadas
  MOVF CNT, W
  SUBWF N, W
  BTFSC STATUS, Z
  GOTO END_LOOP0

LOOP1:
  ;Verifica si no se ha llegado al final del arreglo
  MOVLW H'30'
  SUBWF FSR, W
  BTFSC STATUS, Z
  GOTO END_LOOP1
  ;Carga los valores de INDF e INDF - 1  a los registros
  ; I y J para compararlos
  MOVF INDF, W
  MOVWF I
  DECF FSR
  MOVF INDF, W
  MOVWF J
  INCF FSR
  ; Compara INDF e INDF - 1
  MOVF I, W
  SUBWF J, W  ; J - I
  BTFSC STATUS, C
  GOTO CHECK_Z
  ; Si llega el flujo a aqui, el resultado
  ; de la resta es negativo
CONTINUE:
  ;Se pasa a la siguiete localidad de memoria
  INCF FSR
  GOTO LOOP1

END_LOOP1:
  ;Se regresa el apuntador al inicio del arreglo
  MOVLW H'21'
  MOVWF FSR
  INCF CNT
  GOTO LOOP0

END_LOOP0:
  GOTO \$

CHECK_Z:
  BTFSC STATUS, Z
  GOTO CONTINUE ;El resultado de la resta es cero
```

```
;El resultado de la resta es positivo
;por lo que los elementos se deben intercambiar
MOVF J, W
MOVWF INDF
DECF FSR
MOVF I, W
MOVWF INDF
GOTO CONTINUE
END
```

2. Conclusiones

- López Becerra Ricardo: En esta práctica aprendimos sobre el direccionamiento indirecto en la PIC16. A diferencia de otros procesadores, no todos los registros pueden ser utilizados para realizar direccionamiento indirecto. En este caso se debe usar unicamente los registros FSR e INDF. Con la capacidad de realizar direccionamiento indirecto implementamos dos de los algoritmos más comunes y fáciles para aplicar los conocimientos teóricos, los cuales fueron una búsqueda lineal y Bubble Sort. Debido a que implementamos exitosamente estos algoritmos, considero cumplido el objetivo de la práctica.
- Navarrete Zamora Aldo Yael: En la práctica el direccionamiento indirecto fue muy importante, en el caso de este microcontrolador, tuvimos la capacidad de acceder a el registro FSR que funciona similar a un apuntador de memoria en cualquier otro programa de lenguaje medio-alto y a su contenido con el registro INDF. Implementamos con eficacia algoritmos de ordenamiento (BubbleSort) y una búsqueda lineal para encontrar el elemento mínimo en un conjunto dado de memoria RAM.