M16

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 2:

Programación en ensamblador direccionamiento indirecto

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Zepeda Baeza Jessica	317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 11 de marzo de 2023

Fecha de entrega: 14 de marzo de 2023

Desarrollo

Para esta práctica se buscó crear programas en ensamblador utilizando el modo de direccionamiento indirecto. En este método de direccionamiento, lo primero que se debe hacer es especificar el banco que se estará utilizando. Posteriormente, se puede hacer uso de FSR, una dirección especificada, y de INDF, el valor de lo que contiene esa dirección. Para esta práctica se desarrollaron los siguientes dos ejercicios:

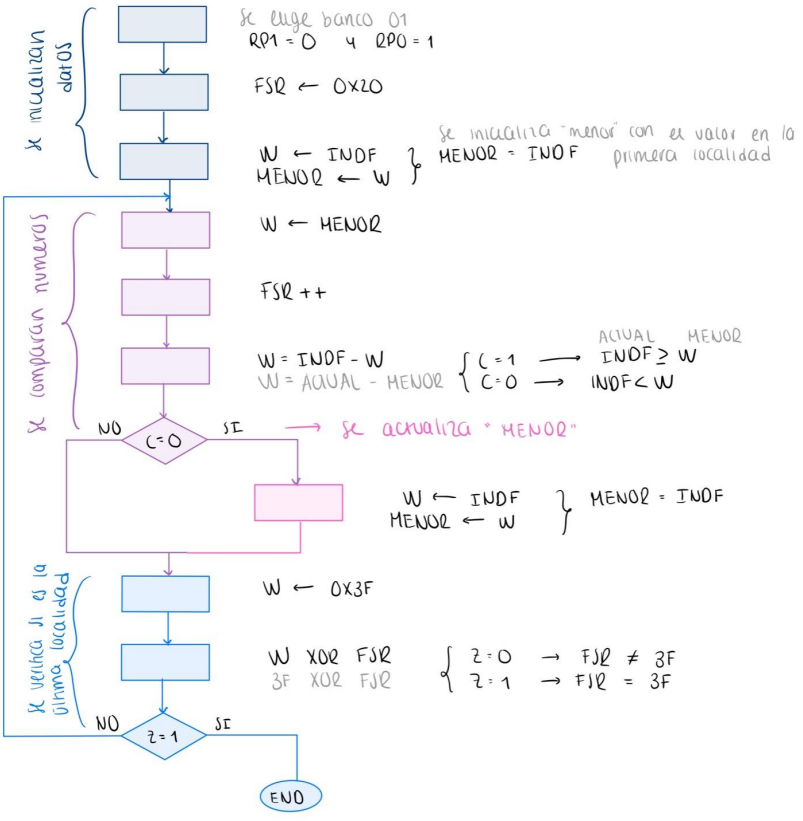
1. Un programa que encuentre el valor mínimo que se encuentre entre un conjunto de datos ubicado en las localidades 0x20 a 0x3F y mostrar ese valor mínimo en la dirección 40H.
2. Un programa que ordena de manera ascendente los números de un conjunto de datos que va desde 0x20 a 0x2F.

Ambos programas son más simples de resolver utilizando direccionamiento indirecto ya que de manera directa se tendría que especificar cada dirección que se quiere acceder. De esta manera se pueden realizar recorridos en un conjunto de datos.

Algoritmos

Ejercicio 2

Para el Ejercicio 2 se realizó la siguiente carta ASM:

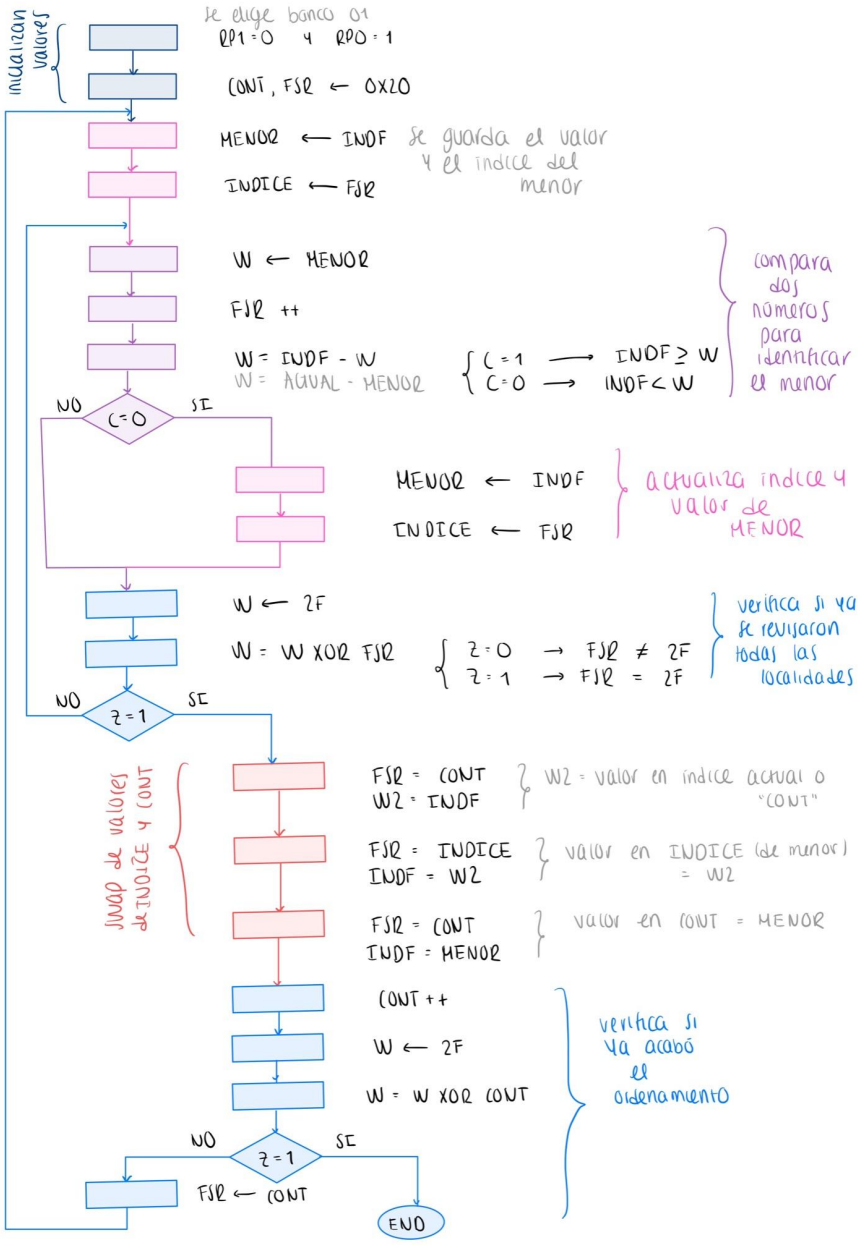


Lo primero que hace el programa es guardar el primer valor en la localidad “menor”. De ahí pasa al siguiente valor y lo compara con el guardado en “menor” mediante la resta de ambos valores. La bandera de Carry indica si el valor actual es menor cuando C=0 entonces en ese caso se actualiza el valor de “menor” asignándole el valor actual. Si C=1, no se hace ninguna modificación.

Por último, se verifica que la localidad actual (FSR) no sea última (3F). En caso de que lo sea, el programa termina. Y en caso contrario, regresa al bloque de comparación, donde ahora se verifica la siguiente localidad.

Ejercicio 3

Para el Ejercicio 3 se realizó la siguiente carta ASM:



Este ejercicio implementó un ordenamiento por selección donde se hacen n-1 iteraciones, n siendo el tamaño del arreglo a ordenar. En cada iteración se identifica el número menor y se intercambia con el primero. En la siguiente iteración se analiza un subarreglo que no incluye aquellos elementos ya ordenados.

Se realiza un proceso similar al ejercicio anterior donde el primer número (CONT) se establece como el menor y se guarda su valor y su índice. Después se va comparando con los demás mediante la resta y actualizando el valor e índice de menor cuando sea necesario. Una vez que termina la iteración (se compara con todos los números), se hace el intercambio entre la primera localidad del subarreglo (CONT) y la localidad con el menor valor.

Por último, se verifica si es la última iteración. En caso de que lo sea, el programa termina y en caso contrario, se define la primera localidad del nuevo subarreglo y se repite el proceso.

Programas comentados

Ejercicio 2

```
PROCESSOR p16f877
INCLUDE <p16f877.inc>

MENOR equ H'40'          ;Registro donde se almacena el mínimo
TF equ H'41'             ;Registro para 3F (último registro a checar)

    ORG 0
    GOTO INICIO
    ORG 5

INICIO:
    BCF STATUS,RP1       ;Se hace cero el bit 1 del banco
    BSF STATUS,RP0       ;Se hace uno el bit 0 del banco
    MOVLW 0X20
    MOVWF FSR            ;FSR = 0X20
    MOVF INDF,W          ;El valor de la localidad 0X20 se guarda en W
    MOVWF MENOR          ;Se guarda el valor de W en MENOR

COMP:                    ;Se realiza la comparación entre dos números utilizando la resta
    MOVF MENOR,W         ;El valor menor se almacena en W
    INCF FSR             ;Se incrementa FSR en uno
    SUBWF INDF,0         ;INDF-W:NUM2-NUM1
    BTFSS STATUS,C       ;Si el carry es cero NUM2 es menor
    GOTO ACTUALIZAR      ;Si carry es cero -> se actualiza el menor
    GOTO TRESF           ;Si carry es uno -> checar valor de FSR

ACTUALIZAR:              ;Actualizar el valor del menor
    MOVF INDF,W
    MOVWF MENOR          ;MENOR = INDF

TRESF:                   ;Checar que no se haya llegado a localidad 3F
    MOVF TF,W
    XORWF FSR,W          ;Se genera la comparación 3F == FSR
    BTFSC STATUS,Z
    GOTO $               ;Si son iguales -> final del programa
    GOTO COMP            ;Si no son iguales -> seguir comparando
    END
```

Ejercicio 3

```
PROCESSOR p16f877
INCLUDE <p16f877.inc>

MENOR equ H'40'          ;Registro donde se almacena el mínimo
ULTIMO equ 0x2F          ;Registro donde se alm
INDICE equ H'41'         ;Índice del registro con valor mínimo
CONT equ H'42'           ;Conteo del número de registros a comparar
W2 equ H'43'             ;Registro de variable temporal
ULTIMOCOMP equ 0X30      ;Dirección del último registro a ordenar

    ORG 0
    GOTO INICIO
    ORG 5

INICIO:
    BCF STATUS,RP1       ;Se hace cero el bit 1 del banco
    BSF STATUS,RP0       ;Se hace uno el bit 0 del banco
    MOVLW 0X20
    MOVWF FSR            ;FSR = 0X20
    MOVWF CONT           ;Se mueve a contador el valor inicial
ANTES:
    MOVF INDF,W          ;El valor de la localidad FSR se guarda en W
    MOVWF MENOR          ;MENOR = INDF
    MOVF FSR,W           ;La dirección de FSR se guarda en W
    MOVWF INDICE         ;INDICE = FSR
COMP:
    MOVF MENOR,W         ;Se realiza la comparación entre dos números utilizando la resta
    INCF FSR             ;El valor menor se almacena en W
    SUBWF INDF,0         ;INDF-W:NUM2-NUM1
    BTFSS STATUS,C       ;Si el carry es cero NUM2 es menor
    GOTO ACTUALIZAR      ;Si carry es cero -> se actualiza el menor
    GOTO FINAL           ;Si carry es uno -> checar valor de FSR
ACTUALIZAR:              ;Actualizar valor del menor y su índice
    MOVF INDF,W
    MOVWF MENOR          ;MENOR = INDF
    MOVF FSR,W
    MOVWF INDICE         ;INDICE = FSR

FINAL:                    ;Verificar que no sea el último elemento
    MOVLW ULTIMO
    XORWF FSR,W          ;Haciendo comparación: FSR == ULTIMO
    BTFSC STATUS,Z
    GOTO SWAP            ;Si son iguales realizar el swap
    GOTO COMP            ;Si no son iguales seguir comparando

SWAP:                    ;Realiza el swap del mínimo con el primer valor
    MOVF CONT,W
    MOVWF FSR            ;FSR = CONT
    MOVF INDF,W
    MOVWF W2             ;W2 = INDF
    MOVF INDICE,W
    MOVWF FSR            ;FSR = INDICE
    MOVF W2,W
    MOVWF INDF           ;INDF = W2

    MOVF CONT,W
    MOVWF FSR            ;FSR = CONT
    MOVF MENOR,W
    MOVWF INDF           ;INDF = MENOR

    INCF CONT            ;Incrementar el contador, se tiene un elemento menos
    MOVLW ULTIMO
    XORWF CONT,W         ;Realizar comparación CONT == ULTIMO
    BTFSC STATUS,Z       ;Se realiza para saber si ya se ordenó todo
    GOTO $               ;Si son iguales se acaba el programa
                         ;De lo contrario se realiza lo siguiente
    MOVF CONT,W
    MOVWF FSR            ;FSR = CONT
    GOTO ANTES           ;Volver a buscar el mínimo para acomodarlo
END
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

Con los ejercicios realizados en esta práctica, se trabajó el uso del direccionamiento indirecto para el que primero se debe indicar un cierto banco utilizando las banderas RP0 y RP1. Además se utilizan los registros FSR e INDF, los cuales simulan el uso de apuntadores ya que mientras FSR se refiere a una localidad, INDF se refiere al valor en esa localidad apuntada. En los ejercicios se trabajó tanto la búsqueda de un número como el ordenamiento de un conjunto de números. En ambos, fue necesario hacer varias comparaciones entre los valores en dos localidades y esto fue mucho más sencillo utilizando el direccionamiento indirecto ya que de esta manera solo es necesario incrementar o decrementar FSR para analizar una localidad contigua y no especificar la dirección en hexadecimal. También se aprendió a identificar el mayor entre dos números mediante la resta, ya que la bandera de Carry es afectada.

Barreiro Valdez Alejandro:

En esta práctica se pudo utilizar el direccionamiento indirecto en ensamblador para la resolución de dos problemas comunes de la computación. Los dos problemas que se resolvieron fueron la búsqueda de un valor mínimo en un conjunto de datos y el ordenamiento de un conjunto de datos. Para ambos problemas se debe recorrer un conjunto de datos y hacer las mismas operaciones sobre él. Para encontrar el mínimo se debe comparar cada uno de los valores con uno que se llama mínimo y actualizarlo cuando este cambie. Para el ordenamiento se realiza algo similar solo que al final se genera un swap del primer elemento con el mínimo. Ambos problemas necesitan que se opere el conjunto de datos en su totalidad y para ello se utilizó el direccionamiento indirecto. Sin esta herramienta se debería especificar cada una de las localidades que se desea operar. El direccionamiento indirecto es una gran herramienta para operar datos que se encuentran contiguos.