



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO



FACULTAD DE INGENIERÍA

MATERIA

- Laboratorio de Microcomputadoras
 - Grupo:04

PRÁCTICA 02

PROFESOR

- M.I. Ruben Anaya García

ALUMNOS

- Carreón Guzmán Mariana Ivette
 - Núm. Cta.: 312103914
 - Gpo. Teoría: 04
 - Rojas Méndez Gabriel
 - Núm. Cta.: 314141712

SEMESTRE 2022-1

Objetivo

Analizar la programación en lenguaje ensamblador. Realizar algoritmos en lenguaje ensamblador empleando direccionamiento indirecto.

Introducción

Este procesador dispone de dos modos de direccionamiento, estos son direccionamiento directo e indirecto.

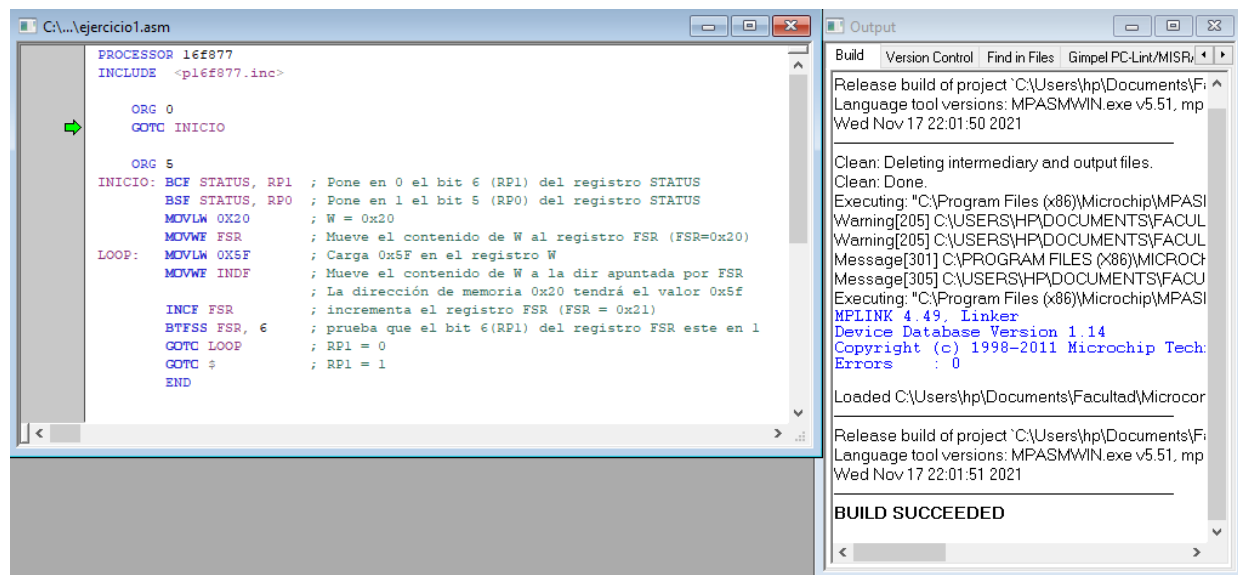
Direccionamiento indirecto

El banco de memoria RAM es seleccionado por la codificación de los bits más significativos de los registros STATUS (IRP) y FSR. La dirección dentro del banco será especificada por los bits restantes del registro FSR. Para acceder a la dirección especificada por FSR, deberá ser indicando como parámetro de la instrucción al registro INDF.

Desarrollo

Ejercicio 1

Iniciamos poniendo RP1 en 0 y RP0 en 1, después vamos a apuntar a la dirección 20, esta será la dirección de W, moveremos el contenido de W al registro FSR para después cargar el valor de 5F en W. De esta forma tendremos en la dirección 0x20 el valor de 5F, incrementamos el registro de FSR, por lo que ahora estamos en la dirección 0x21. En caso de que RP1 se encuentre en 0 se entra en un loop y en caso de estar en 1 se saldrá de dicho loop.



The screenshot shows the MPLINK 4.49 linker window. The main window displays assembly code for 'ejercicio1.asm'. The code includes a processor declaration, an include file, and a loop that increments the FSR register until RP1 is 1. The output window on the right shows the build process, including cleaning files, executing the linker, and a final 'BUILD SUCCEEDED' message.

```
PROCESSOR 16f877
INCLUDE <pl6f877.inc>

ORG 0
GOTO INICIO

ORG 5
INICIO: BCF STATUS, RP1 ; Pone en 0 el bit 6 (RP1) del registro STATUS
        BSF STATUS, RP0 ; Pone en 1 el bit 5 (RP0) del registro STATUS
        MOVLW 0x20      ; W = 0x20
        MOVWF FSR       ; Mueve el contenido de W al registro FSR (FSR=0x20)
LOOP:   MOVWF 0x5F       ; Carga 0x5F en el registro W
        MOVWF INDF       ; Mueve el contenido de W a la dir apuntada por FSR
        ; La dirección de memoria 0x20 tendrá el valor 0x5f
        ; incrementa el registro FSR (FSR = 0x21)
        INCF FSR
        BTFSF FSR, 6     ; prueba que el bit 6(RP1) del registro FSR este en 1
        GOTO LOOP       ; RP1 = 0
        GOTO $           ; RP1 = 1
END
```

Output window content:

```
Build Version Control Find in Files Gimpel PC-Lint/MISR/
Release build of project 'C:\Users\hpl\Documents\Facultad\Microch...
Language tool versions: MPASMWIN.exe v5.51, mp
Wed Nov 17 22:01:50 2021

Clean: Deleting intermediary and output files.
Clean: Done.
Executing: "C:\Program Files (x86)\Microchip\MPAS...
Warning[205] C:\USERS\HPL\DOCUMENTS\FACUL...
Warning[205] C:\USERS\HPL\DOCUMENTS\FACUL...
Message[301] C:\PROGRAM FILES (X86)\MICROCH...
Message[305] C:\USERS\HPL\DOCUMENTS\FACU...
Executing: "C:\Program Files (x86)\Microchip\MPAS...
MPLINK 4.49, Linker
Device Database Version 1.14
Copyright (c) 1998-2011 Microchip Tech...
Errors : 0

Loaded C:\Users\hpl\Documents\Facultad\Microcor...
Release build of project 'C:\Users\hpl\Documents\F...
Language tool versions: MPASMWIN.exe v5.51, mp
Wed Nov 17 22:01:51 2021

BUILD SUCCEEDED
```

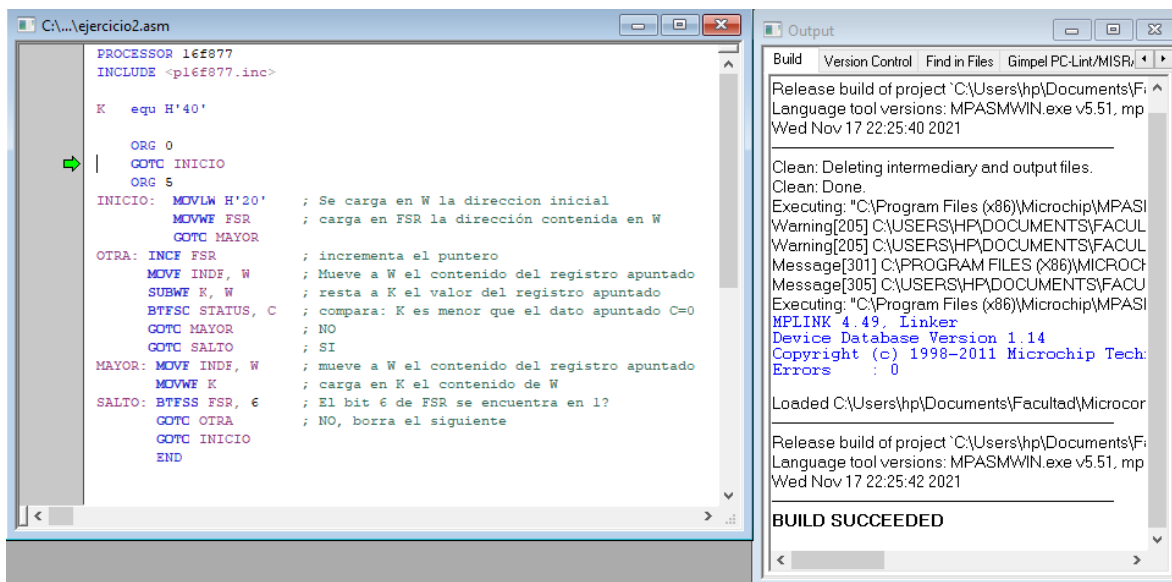
The image displays two screenshots of a 'File Registers' window, likely from a debugger or memory viewer. Both windows show a table of memory addresses (00 to 0F) and their corresponding values in hexadecimal. The top screenshot shows values such as 0D 3B 24 at address 002, while the bottom screenshot shows 0E 3B 40 at the same address. The windows also include a 'Hex' tab and a 'Symbolic' tab.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	A:
000	--	00	0D	3B	24	00	00	00	00	00	00	00	00	00	00	00	---
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
020	5F	5F	5F	5F	00	00	00	01	10	00	00	00	00	00	00	00	---
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
080	--	FF	0D	3B	24	3F	FF	FF	FF	07	00	00	00	00	00	--	---
090	--	00	FF	00	00	--	--	--	02	00	--	--	07	00	00	00	---
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
100	--	00	0D	3B	24	--	--	--	--	00	00	00	00	00	00	00	---
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	---

En las imágenes podemos apreciar como entra en el loop.

Ejercicio 2

Elaborar un programa que encuentre el número menor de un conjunto de datos ubicados entre las localidades de memoria 0x20 a 0x3F y que se muestre dicho valor en la dirección 40H



En este programa iremos haciendo la comparación de los números y recorriendo las direcciones. Lo primero que hacemos es llenar las localidades de memoria correspondientes como se muestra en la siguiente imagen.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	As
000	--	00	00	1B	40	00	00	00	00	00	00	00	00	00	00	00	----
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	5F	2F	4F	08	05	04	03	01	00	7F	5F	1F	FF	5F	02	03	_/O.....
030	5F	09	07	FF	02	5F	5F	6F	8F	9F	5F	5F	5F	5F	5F	5F	-----c
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	--	FF	00	1B	40	3F	FF	FF	FF	07	00	00	00	00	00	--	----@?..
090	--	00	FF	00	00	--	--	--	02	00	--	--	07	00	00	00	-----

Como podemos observar el valor más pequeño es el 00 y este es colocado en la dirección 40H

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	As
000	--	00	09	18	34	00	00	00	00	00	00	00	00	00	00	00	----4...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	5F	2F	4F	08	05	04	03	01	00	7F	5F	1F	FF	5F	02	03	_/O.....
030	5F	09	07	FF	02	5F	5F	6F	8F	9F	5F	5F	5F	5F	5F	5F	-----c
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	--	FF	09	18	34	3F	FF	FF	FF	07	00	00	00	00	00	--	----4?..

Ejercicio 3

Desarrollar el algoritmo y el programa que ordene de manera ascendente el conjunto de datos ubicados en los registros 0x20 al 0x2F.

El programa va realizando el acomodo de los números mediante comparaciones, se apunta a un número y si este es menor al que se tiene en el registro se va a realizar el cambio.

```
C:\...\ejercicio3.asm

PROCESSOR 16f877
INCLUDE <p16f877.inc>

K equ H'40' ;Registro que guarda el primer numero a comparar
L equ H'41' ;Registro que guarda el segundo numero a comparar
LIM equ H'42' ;registro que guarda la direccion limite de la lista
BAND equ H'43' ;bandera para verificar si la lista esta ordenada
ORG 0
GOTO INICIO
ORG 5
INICIO: MOVLW H'2F'
        MOVWF LIM ; Ce carga a LIM la direccion limite de la lista (LIM = 2F)
        MOVLW H'20' ; Se carga en W la direccion inicial
        MOVWF FSR ; carga en FSR la dirección contenida en W (W = 20)
        MOVLW H'00'
        MOVWF BAND ; Inicializa BAND=0 indicando que no hay cambios en la lista

NUM1:   MOVF INDF, W ; se carga a W el contenido de lo que apunta FSR
        MOVWF K ; se carga en K el contenido de W (K = W)

NUM2:   INCF FSR ; incrementa el puntero
        MOVF INDF, W ; Se carga a W el contenido del registro apuntado

COMPARA: SUBWF K, W ; resta a K el valor del registro apuntado (W = K - W)
         BTFSC STATUS, C ; compara: ¿ NUM1 > NUM2 ?
         GOTO CAMBIO ; C=0 : SI
         GOTO LISTAF ; c=1 : NO

CAMBIO: MOVF INDF, W ; carga en W el contenido del registro apuntado
        MOVWF K ; carga en K el contenido de W
        DECF FSR ; regresamos al numero anterior
        MOVF INDF, W ; carga en W el numero anterior (el que apunta FSR)
        MOVWF L ; guarda el numero anterior en L (L = W)
        MOVF K, W ; se carga en W el numero menor. Se mueve a W el contenido de K
        MOVWF INDF ; se carga a donde apunta FSR el contenido de W
        ; (se coloca el numero menor en una posicion anterior)

        INCF FSR ; Incrementamos el puntero (volvemos a la posicion original)
        MOVF L, W ; se carga a W el contenido de L (W = L)
        MOVWF INDF ; se encarga en la pos. que apunta FSR, el contenido de W
        ; (se coloca el numero mayor en una posicion ascendente)

        MOVF INDF, W ; cargamos en W el contenido del del registro apuntado
        MOVWF K ; cargamos en K el contenido de W; se actualiza el registro de
        ; para en una comparacion futura

        MOVLW H'01'
        MOVWF BAND ; se carga a BAND=01, esto indica que hubo almenos un cambio
        ; al revisar la lista de nums.
        ; Hasta que este CONT se mantenga en CONT=00 habiendo recorrido
        ; la lista completa podemos estar seguros que esta ordenada.
```

```

C:\...\ejercicio3.asm

LISTAF:      ; Fin de la lista?
MOVW LIM, W  ; Se carga a W el contenido de LIM (W = LIM)
SUBWF FSR, W ; resta a FSR el valor del registro LIM (W = FSR - W)
BTFS STATUS, C ; compara: el apuntador (FSR) es menor que LIM (¿FSR < LIM?)
GOTC NUM1    ; c=1 : SI ; Continúa con los siguientes numeros

MOVW H'01'   ; C=0 : NO ; Fin de la lista; se limpia el registro W
SUBWF BAND, W ; resta para saber si esta ordenado (W = BAND - W)
BTFS STATUS, C ; ¿Esta ordenado?
GOTC $       ; c=1 : SI ; Lista Ordenada
GOTC INICIO  ; C: NO ; Se revisa de nuevo la lista desde el principio

END

```

Como podemos observar en esta captura los valores se encuentran desordenados y en la siguiente imagen después de correr el programa se ordenan de menor a mayor.

File Registers																	As ^
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	--	00	23	18	23	00	00	00	00	00	00	00	00	00	00	00	-.#.#...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	2F	4F	08	5F	05	04	03	01	00	7F	5F	1F	FF	5F	02	03	/0.
030	5F	09	07	FF	02	5F	5F	6F	8F	9F	06	01	A1	A1	0A	06c
040	5F	5F	2F	01	00	00	00	00	00	00	00	00	00	00	00	00	.../.....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	--	FF	23	18	23	3F	FF	FF	FF	07	00	00	00	00	00	--	-.#.#?..
090	--	00	FF	00	00	--	--	--	02	00	--	--	07	00	00	00	-...--
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

File Registers																	As ^
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	--	00	0D	08	21	00	00	00	00	00	00	00	00	00	00	00	-...!...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	00	01	02	03	03	04	05	08	1F	2F	4F	5F	5F	5F	7F	FF
030	5F	09	07	FF	02	5F	5F	6F	8F	9F	06	01	A1	A1	0A	06c
040	01	5F	2F	00	00	00	00	00	00	00	00	00	00	00	00	00	.../.....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	--	FF	0D	08	21	3F	FF	FF	FF	07	00	00	00	00	00	--	-...!?
090	--	00	FF	00	00	--	--	--	02	00	--	--	07	00	00	00	-...--
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Conclusiones:

Mariana Carreón Guzmán

En esta práctica pudimos elaborar 3 programas que requerían el uso de algoritmos, así como del direccionamiento indirecto. En cada uno de los ejercicios pudimos comprender poco a poco el

funcionamiento del direccionamiento ya que se requería hacer comparaciones e ir moviéndonos en los registros para poder ir realizando los cambios al momento de ir ordenando los valores.

Gabriel Rojas Méndez

En esta práctica pudimos ir comprendiendo a mayor profundidad el funcionamiento del lenguaje ensamblador, fuimos comprendiendo cómo hacer el direccionamiento indirecto al ir realizando los cambios de los valores a distintas direcciones de memoria, de igual forma fuimos viendo la forma de aplicar de manera adecuada los algoritmos y ciclos para poder ir iterando de manera secuencial al realizar el ordenamiento.