Lab Sheet 2

PIC16F84A Instructions

1. Overview

The instructions set of the PIC16F84A is divided into:

1. Byte-oriented instructions that operate on bytes of data and are further divided into:
    a. Data transfer instructions,
    b. Mathematical and logical instructions.
2. Bit-oriented instructions that operate on a single bit.
3. Literal and control instructions. Some instructions were tested last week in the lab (MOVLW and ADDLW) and in the report (SUBLW, IORLW, and XORLW).

2. Testing Byte-oriented Instructions

Program 1: First of all, create a project for this lab as was shown in the first week. Write the following program in the source code file:

```
MOVLW   0x20
MOVWF   0x3A
ADDWF   0x3A
NOP
```

Code 1 - Program 1

In order to view the contents of the file registers, select "Window → Target Memory Views → File Registers". The file registers window will appear as shown in Fig. 1.

Build the project and only execute the "goto main" instruction. After inspecting the file registers window, please answer the following (include this in the questions section of the report):

a. Why does the memory run from 01 till 4F and then from 81 till CF? Note that all other memory locations are grey dashed.
b. Why are memory locations 02 and 82 both equal to 05?

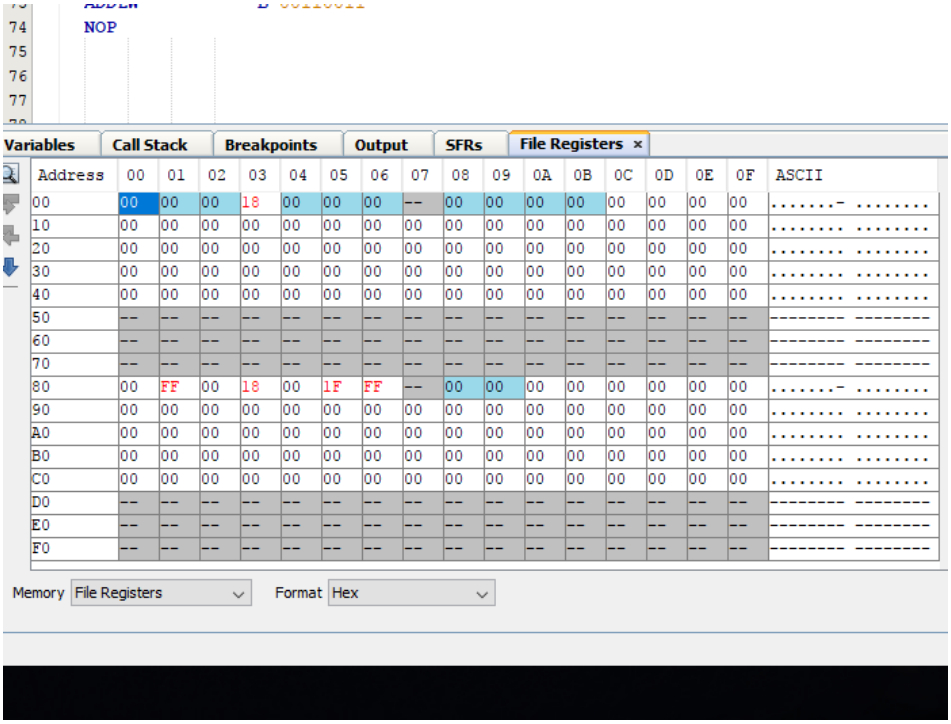

Fig. 1 File registers window.

Continue executing the rest of the program and fill in the table below after executing each instruction:

Instruction	W	Mem[0x3A]
MOVLW 0x20		
MOVWF 0x3A		
ADDWF 0x3A		

Table 1 Results of the first byte-oriented instructions program.

After inspecting the results:

a. Where was the result of the last operation placed? And why?
b. Modify the program such that W becomes the destination of the result of the last instruction. Rebuild your program and test it. (**Demonstrate to the instructor**)
c. Modify the program to define memory location 0x3A as a variable 'X' (using the "EQU" directive) and use the variable name instead of the address in the program. Rebuild your program and test it. You should add the line (X EQU 0x3A) before the first (org 0x00).

Program 2: In order to move a file register into W, we can use the MOVF instruction. Write the following small code into the source file (comment the old code).

```
MOVF    X, 0
ADDWF   X, 0
NOP
```

Code 2 –Program 2

You can start the program with a specific value in X by double clicking on the Hex value of the variable X in the file register window and inserting the value 0x42 (you can also double click and for the Decimal value insert 66). Now, simulate your program, fill in the table below.

Instruction	W	X
MOVF X,0		
ADDWF X,0		

Table 2 Results of the second byte-oriented instructions program.

Exercise: Write a program that:

- Defines three variables X, Y and K at addresses 0x1C, 0x1D, and 0x1E using EQU before org 0x00.
- Moves X into W,
- Adds Y to W placing the result in W,
- Moves the addition result into K,
- Moves the binary value '01010101' into X **(How many instructions do you need here?)**
- Clears the variable Y,
- Moves K into W,
- ANDs X and W placing the result in W,
- Moves the AND result into Y,
- Complements the value of Y
- Before running your program set the starting values of X and Y to 0xE4 and 123, respectively, in the file register window. You can switch the view from hex to symbol if you wish to enter values in decimal.

Simulate your program, fill in the table below and **verify** the results.

Instruction	File Registers				Flags		
	W	**X**	**Y**	**K**	**Z**	**DC**	**C**

Table 3 Results of the exercise program.