Université de Jijel.
Faculté des sciences et de la technologie
Département d'électrotechnique
1ière Année Master électrotechnique

T.P. N°1 microprocesseur

Opérations arithmétiques avec le PIC 16F877

I. Aperçu

Les instructions du PIC16F877 peuvent être classées en trois types :

1. Les instructions orientées octet (Byte-Oriented File Register Operation)
2. Les instructions orientées bit (Bit-Oriented File Register Operation)
3. Les instructions littérales (Literal operation)

1. Les instructions orientées octet

Dans ce type d'instruction, l'opération s'effectue toujours entre **le registre w** (accumulateur) et un **emplacement mémoire (f)** qu'on doit indiquer dans l'instruction. L'opération s'effectue donc entre le **contenu (représenté par […]) de l'emplacement mémoire** indiqué et le **contenu de l'accumulateur** (registre w). Le résultat peut être, suivant le cas, stocké soit dans l'emplacement mémoire indiqué dans l'instruction, soit stocké dans l'accumulateur. Ce choix est indiqué à travers le bit de destination d. si **d = 0, le résultat est stocké dans l'accumulateur**, si **d = 1 le résultat est stocké dans l'emplacement mémoire**.

Schéma général :

CODE f,d

CODE : indique le mnémonique de l'opération
f : indique un emplacement mémoire entre 0x00 et 0x7F
d : indique le bit de destination 0 ou 1
(dans certaines instructions, seul le code est présent)

Exemple

MOVF 0x70,0 ; Transférer le contenu de l'emplacement (l'adresse) mémoire 0x70 dans l'accumulateur (d =0)

Il faut noter que cette instruction ne permet pas de connaître le contenu de l'adresse 0x70

2. Les instructions orientées bit

Dans ce type d'instructions, l'opération s'effectue **toujours sur un bit** (indiqué par b) du contenu de l'emplacement mémoire indiqué dans l'instruction.

Schéma général :

CODE f,b

CODE : indique le mnémonique de l'opération
f : indique un emplacement mémoire entre 0x00 et 0x7F
b : indique le numéro du bit ciblé par l'opération, entre 0 et 7

Exemple 1

BCF 0x7F,3 ; met à zéro le bit numéro 3 du contenu de l'adresse mémoire 0x7F

Il faut noter qu'après l'exécution de cette instruction nous connaissons uniquement l'état du bit numéro 3 du contenu de l'adresse 0x7F.

Exemple 2

BTFSC 0x03,3 ; test le bit numéro 3 du contenu de l'adresse 0x03. Si le bit est à zéro, l'instruction suivante n'est pas exécutée.

Ce genre d'instructions sont appelées instructions de rupture de séquence conditionnelle car elles permettent de créer un branchement dépendant de l'état du bit testé.

3. Les instructions Littérales

Dans ce type d'instruction, l'opération s'effectue entre le **contenu du registre w** et le **contenu de 8 bit fourni par l'instruction**. Le résultat est toujours stocké dans le registre w. Dans ce genre d'instructions, on connaît **l'un des contenus** intervenant dans l'opération ; on le "**lis**" dans l'instruction.

Schéma général :

CODE K

CODE : indique le mnémonique de l'opération
K : indique un contenu de 8 bit (entre 0x00 et 0xFF)

Exemple 1 :

SUBLW 0X20 ; soustrait du contenu du registre w la valeur 0x20

Exemple 2 :

GOTO 0x200 ; permet de se brancher à l'instruction d'adresse 0x200

Ce genre d'instruction est appelée instruction de rupture de séquence inconditionnelle

II. les bits d'état

Le PIC16F877 possède **trois bits d'état** qui vont nous permettre d'identifier **la nature du résultat** d'une opération **arithmétique ou logique**

Ces trois bits sont implémenté dans le registre mémoire qui se trouve à l'adresse 0x03 et qui porte le nom de STATUS :

STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

a. Le bit Z (bit N° 2)

Ce bit permet d'indiquer un résultat nul. Ce bit **passe à 1** après une opération arithmétique ou logique qui fournit un **résultat nul.**

Exemple

```
MOVLW   0x8F   ; [w] = 0x8F
SUBLW   0x8F   ; [w] = 0x00, Z = 1
```

Cependant, certaines instructions n'affectent pas le bit Z

Exemple

DECFSZ, **MOVWF**

a. Le bit C (bit N° 0)

Ce bit permet d'indiquer si une **retenue est générée** après une opération **arithmétique** et Les deux opérations logiques **RLF** et **RRF.** Ce bit **passe à 1** en cas de **retenue** générée par le **huitième bit** d'une opération.

Exemple

```
MOVLW   0b1010 0101  ;   [w] = 0b1010 0101
ADDLW   0b1000 0011  ;   [w] = 0b0010 1000, C = 1
```

b. Le bit DC (bit N° 1)

Ce bit permet d'indiquer si une "***demi***" **retenue est générée** après une opération **arithmétique.** Ce bit **passe à 1** en cas de **retenue** générée par le **quatrième bit** d'une opération.

Exemple

```
MOVLW   0b0001 1011  ;   [w] = 0x0001 1011
ADDLW   0b1000 1000  ;   [w] = 0x1010 0011, DC = 1
```

II. Travail à faire

1. Créer un projet Mplab ou ouvrir un projet existant.
2. Créer un nouveau fichier (Menu **File/New**).
3. Enregistrer le fichier (Menu **File/Save As…**) dans le dossier c:\Mplab, lui donner le nom TP1.asm.
4. Saisir le programme ci-dessous, le compiler (Menu **Project/Build**), corriger les éventuelles erreurs puis l'exécuter pas à pas (Touche F7 du clavier).
5. Vérifier le résultat de chaque instruction en visualisant l'état de la mémoire-données (Menu **View/File Registers**). Ajouter un commentaire pour indiquer le rôle de chaque instruction.
6. Indiquer ce que réalise le programme écrit.

```
            MOVLW 0x13
            MOVWF 0x70
            MOVLW 0x05
            MOVWF 0x71

            MOVF  0x71,1
            BTFSC 0x03,2
            GOTO  FIN
            MOVF  0x70,1
            BTFSC 0x03,2
            GOTO  FIN

            CLRW
BOUCLE      ADDWF 0x70,0
            DECF  0x71,1
            BTFSS 0x03,2
            GOTO  BOUCLE
            GOTO  STORE
FIN         CLRW
STORE       MOVWF 0x72
            END
```

Le chargé de T.P.
B. MEDJAHED