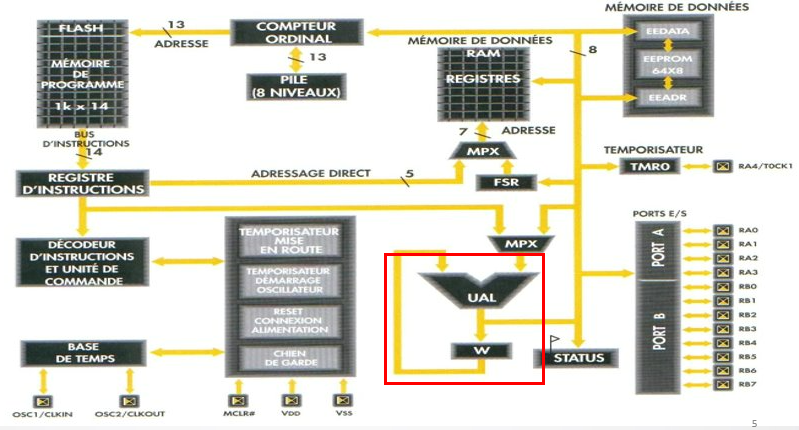
**Chapitre 3 : La mémoire de programme PIC 16F84A**

**Architecture interne du PIC16F84A**

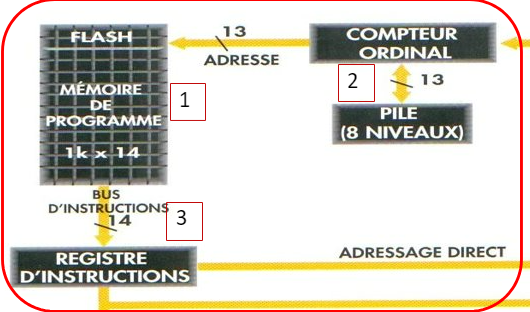
**1.L’Unité Arithmétique et Logique**



* Unité arithmétique et logique(UAL): Chargée d’ éxécuter les opérations arithmétiques  
  (addition, soustraction, division,…) et logiques ( ET, Non, AND,…. ).
* Unité de contrôle : émet les **signaux nécessaires** aux différentes entités (bus adresse, bus de données, UAL..) pour assurer l’exécution **des opérations**.
* Un ensemble de registre

|  |  |
| --- | --- |
| **Fonctionnement de L’Unité Arithmétique et Logique** | |
| **PIC16F84: Architecture à accumulateur :**  Toutes les opérations arithmétiques et logiques sont effectuées en utilisant un seul registre spécial appelé **"accumulateur"**.  Terminologie Microship: l’accumulateur = Registre de travail **W: Working register**  Après l’exécution d’une instruction, l’UAL modifie **le registre STATUS (Registre d’Etat**) indiquant l’issue de celle-ci.  (exemple: Bit Z: résultat null ou pas, Bit C: débordement ou pas…)  **Le registre STATUS se trouve dans la mémoire de données (RAM) à l’adresse mémoire 0x03.** |  |

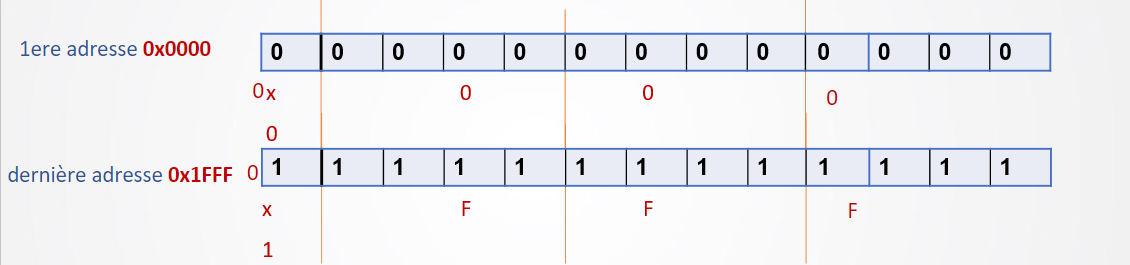
**2. La mémoire de programme (Flash):**



|  |  |
| --- | --- |
| **La mémoire programme** | |
| * Elle contient le **programme à exécuter** * Taille = **1K mots (1024 emplacements)** * Taille d’un mot = **14 bits (PIC16F84: Mid-Range)** * Un mot = instruction du programme de l’application à laquelle le microcontrôleur est dédié. * Type= **FLASH (non volatile, effaçable et reprogrammable).** * La technologie utilisée permet plus de **1000 cycles d’effacement et de programmation.** |  |
| * L'adresse 0000h contient le vecteur du reset.   Lorsque le microcontrôleur est alimenté ou qu'il subit une réinitialisation matérielle ou logicielle, le programme de l'utilisateur commence à s'exécuter à partir de l'adresse mémoire 0000h. En d'autres termes, le processeur saute à l'adresse 0000h pour démarrer l'exécution du programme après une réinitialisation.   * L'adresse 0004h réservée à l'unique vecteur d'interruption.   Lorsqu'une interruption se produit (par exemple, une interruption externe ou une minuterie), le processeur interrompt l'exécution du programme principal et saute à l'adresse mémoire 0004h pour exécuter une routine de traitement d'interruption spécifique définie par le programmeur |  |

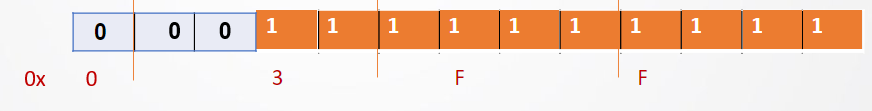
**Adresses d’une case mémoire programme**

* **1. Largeur du bus d’adresse instruction :** 13 bits   
  🡺 on peut adresser 213 mots (instructions)= 23 x 210 = 8 x 1K=8k mots (taille max du programme) en binaire : 1000000000000



* **2. Dans le PIC16F84:**

La mémoire programme FLASH implanté physiquement est de **1K = 210 = 1024 Instructions au max**



|  |  |
| --- | --- |
| **Adresse de début** | **Adresse de fin** |
| 0x0000  0000h | 0x03FF  03FFh |
| h veut dire en hexadécimal : 03FF h :  3×16^2+ 15×16^1+ 15×16^0 | |

L'OPCODE (code opération) est une partie du langage machine ou de l'instruction d'un processeur qui spécifie l'opération à effectuer.

Chaque instruction d'un processeur a un code opération unique qui lui est associé.

**MOV (Move - Déplacer) : (1)**

* OPCODE : 0001
* L'instruction MOV copie des données d'une source vers une destination.

**ADD (Addition - Additionner) : (2)**

* OPCODE : 0010)
* L'instruction ADD ajoute deux valeurs et stocke le résultat dans un registre ou un emplacement mémoire.

**JMP (Jump - Sauter) : (4)**

* OPCODE : 0100
* L'instruction JMP permet au programme de sauter à une autre adresse mémoire pour continuer l'exécution.

**CMP (Compare - Comparer) : (5)**

* OPCODE : 0101
* L'instruction CMP compare deux valeurs et met à jour les indicateurs de statut en fonction du résultat de la comparaison.

**AND (Bitwise AND - ET logique) : (6)**

* OPCODE : 0110
* L'instruction AND effectue un ET logique entre deux valeurs et stocke le résultat dans un registre ou un emplacement mémoire.

Une instruction est codée en **14 bits** et comporte :

* L’OPCODE: spécifie le type d’instruction
* Une ou deux opérandes selon l’opération

Les opérandes sont les données sur lesquelles l'opération doit être effectuée

Exemple: **Charger une valeur immédiate dans un registre**

**En langage C: REGISTRE = 0x8C ;**

MOVLW B'10001100' ; W = B'10001100’

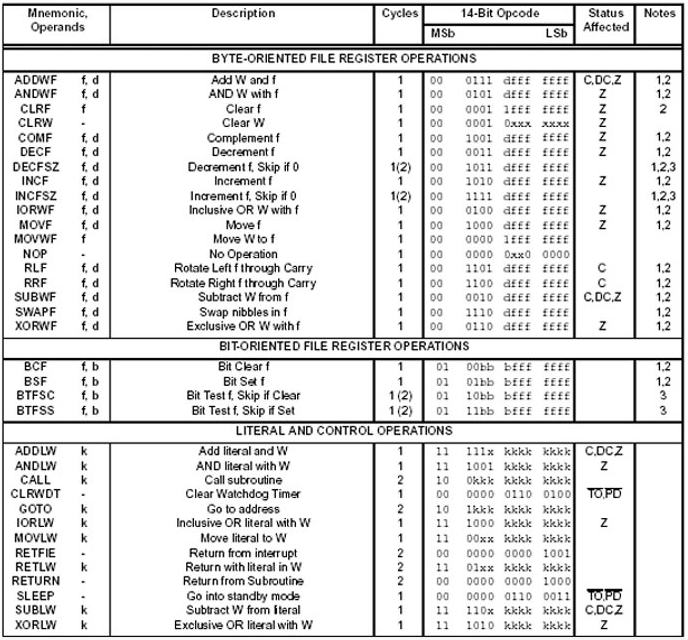
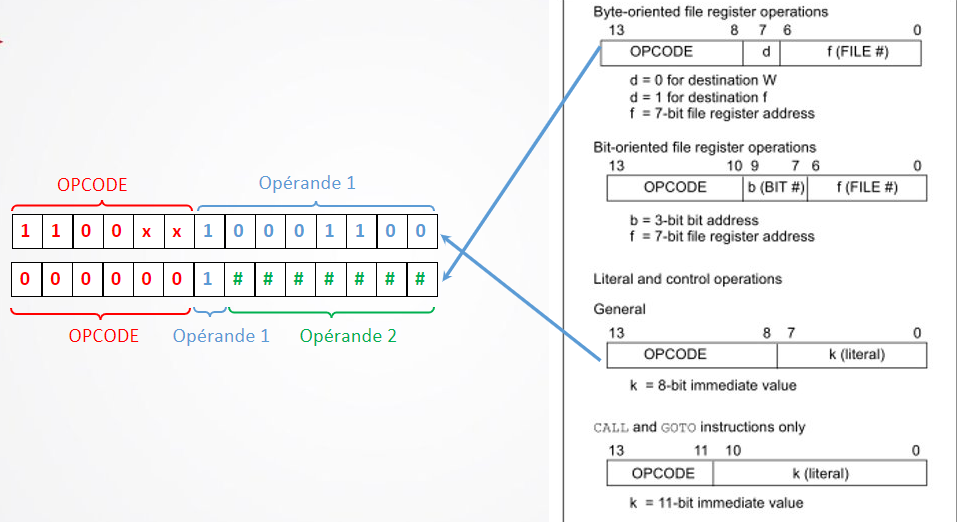
MOVWF REGISTRE ; (REGISTRE) = B'10001100' = 0x8C = D'140'

**MOVLW (Move Literal to W) :**

* L'instruction **MOVLW** est utilisée pour charger une valeur littérale (une constante) dans le registre W (Working register) du microcontrôleur.

**MOVWF (Move W to File register) :**

* L'instruction **MOVWF** est utilisée pour déplacer le contenu du registre W vers un emplacement mémoire, un registre spécial appelé un "registre de fichier" (File Register)

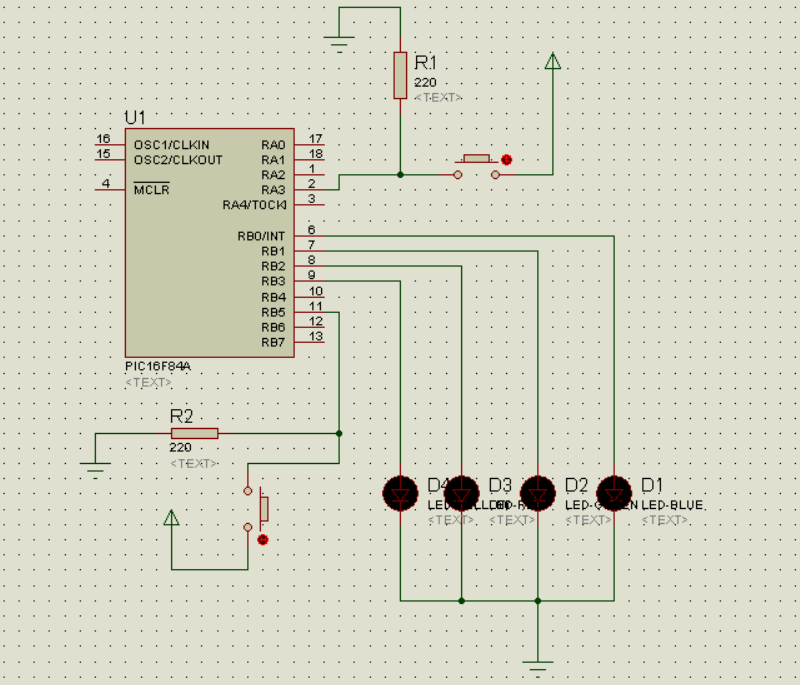


|  |  |
| --- | --- |
| **La compteur ordinal & le registre d’instruction** | |
|  | Le **compteur ordinal ou Program Counter PC**  \* Rôle : contient toujours **l’adresse de la prochaine instruction à exécuter.**  Après l'exécution de chaque instruction, le compteur ordinal est automatiquement incrémenté pour pointer vers l'instruction suivante dans la mémoire du programme.  **\*** Taille = **13 bits** ,  => donc il permet d’adresser 213 mots= 8K mots  => 213 \* 14bits= 23 \* 210 \* 14bits = 8K \* 14 bits  Or seulement 1k (1024 instructions) mots est implémentable avec PIC 16F84  La **pile** contient 8 valeurs: Ce sont des zones  réservées par le système (pas d’adresses).  Remarque : Même si le compteur ordinal a la capacité d'adresser 8192 mots, le microcontrôleur ne dispose que de 1024 mots de mémoire programmable. |
| **Registre d’instruction** | |
|  | Une fois l’instruction à exécuter reconnue via son adresse dictée par le compteur ordinal (PC), son **code** est transféré sur le **Registre d’instruction** ( de taille =14 bits= taille d’une instruction)  A partir du code de l’instruction , le **décodeur** va analyser et dégager les informations nécessaires (types de l’opération, l’emplacement des opérandes et du résultat) à l’exécution de celle-ci. |

**Remarque :** La Pile : Le PIC16F84 dispose également d'une pile (stack) qui est une zone réservée par le système.

Ces emplacements mémoire ne sont pas accessibles directement par le programmeur et sont utilisés pour stocker temporairement des données ou des adresses de retour lors de l'exécution des sous-routines (appel de fonctions). La pile du PIC16F84 a une capacité de 8 valeurs, ce qui signifie qu'elle peut stocker 8 adresses ou données temporaires. L'accès à la pile se fait à l'aide d'instructions spécifiques de pile, telles que **les instructions CALL et RETLW.**

**Exercice Problème 3 :**



Écrire et tester le programme en C qui illustre ce fonctionnement :

les quatre LED sont initialement éteintes.

Les LED RB0 et RB2 contrôlées via le bouton RA3, l'appui sur ce bouton poussoir (RA3) allume ces deux LED, sinon et le bouton est relâché, ces deux LED sont éteintes.

Les LED RB1 et RB3 contrôlées via le bouton RB5, l'appui sur ce bouton poussoir (RB5) allume ces deux LED, sinon et le bouton est relâché, ces deux LED sont éteintes.

|  |  |
| --- | --- |
| **Organigramme** | **Algorithme** |
|  | #include <htc.h>  void main()  {  //Configuration  RB0=0  TRISA = 0x08;  TRISB = 0x20;  //Initilisation  PORTB = 0;  while(1)  {  if(RA3==1)  {  PORTB = 0x05;  }  else {  PORTB = 0x00;  }  if (RB5==1)  {  PORTB = 0x0A;  }  else {  PORTB = 0x00;  }  }  } |