# Lab1-2: Introduction to Instruction Set

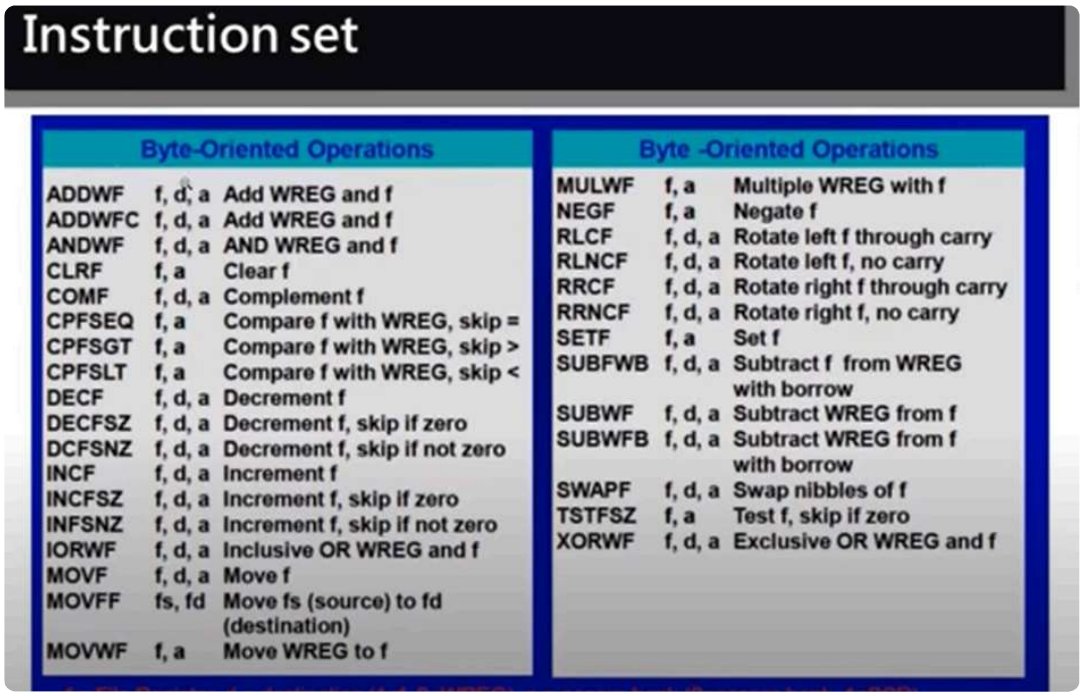
## 目錄



## 介紹

- PIC18: 本實驗所使用到的微處理器
- MPLAB: 編寫並執行程式的平台
- 實驗課規劃: 文件及影片都會提早公布，一定要自己動手操作!!!上機考才能輕鬆歐趴~

## 指令集

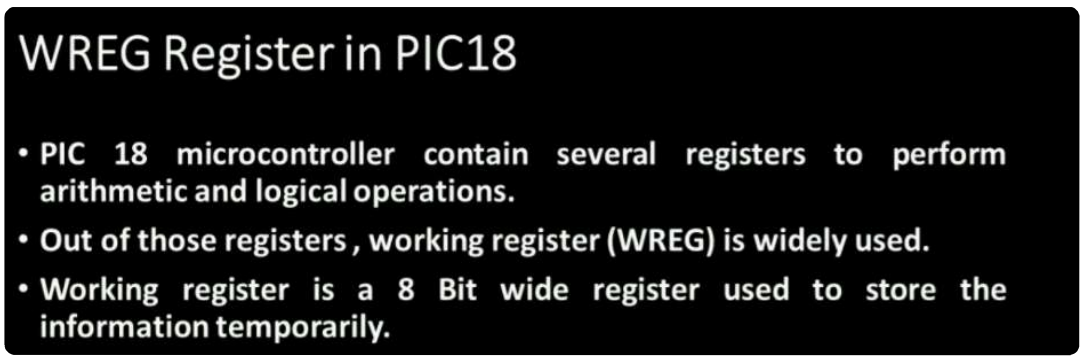



PIC18指令集介紹 (http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/)

- 補充說明: f, d, a代表的是這個指令所需要用到的參數
  - f: 記憶體位置
  - d: 計算後的數值存放在WREG(0/W)中或是指定記憶體位置(1/F,預設)

## WREG

What is WREG? (https://forum.microchip.com/s/topic/a5C3I000000Ly1xEAC/t222489?comment=P-1777506)




- The W register is a special register in the PIC architecture
- It used as one of the 2 operands for ALU operations

- It can be the destination for any ALU operation.

**WREG=working register**

- PIC18中,可以經常用來當作運算元的register
- 運算時常用來暫時存放data

## 初始化程式碼

```
List p=18f4520 ;設備是PIC18F4520
    ;初始化PIC18F
    #include<p18f4520.inc>
        CONFIG OSC = INTIO67
        CONFIG WDT = OFF
        org 0x00 ; 程式從0x00的位置開始執行

```

## 常用的指令集介紹

- MOVLW: 將指定數值放入WREG中

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    MOVLW 0x2B            ; 十六進制
    MOVLW D'15'           ; 十進制
    MOVLW b'00001111'     ; 二進制

    end                   ; 結束程式碼
```

- MOVWF: 將WREG的數值放入指定位置中
    - address一共有12bits，但大部分指令集只能控制後8bits位置(前4個bits要用access bank跟BSR才能控制，後面lab會學到)

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    MOVLW 0x2B
    MOVWF 0x00            ; 將0x2B寫入0x00位置

    MOVLW D'15'
    MOVWF 0x01            ; 將D'15'寫入0x01位置

    end                   ; 結束程式碼
```

- CLRF: 將指定位置清空，也可以將WREG的資料清空

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    ; 清空指定位置
    MOVLW 0x2B
    MOVWF 0x00            ;[0x00] = 0x2B
    CLRF 0x00             ;[0x00] = 0

    ; 清空WREG
    MOVLW 0x2B            ;[WREG] = 0x2B
    CLRF WREG             ;[WREG] = 0

    end                   ; 結束程式碼
```

- INCF/DECF: 將指定位置的數值加一/減一

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00
    ; 好習慣: 對欲處理的位置先進行清空避免殘留值影響結果
    CLRF 0x00              ;[0x00] = 0
    INCF 0x00              ;[0x00] = 1
    INCF 0x00              ;[0x00] = 2
    DECF 0x00              ;[0x00] = 1

    end                    ; 結束程式碼
```

- ADDWF: 將WREG跟指定位置數值相加
  - d = 0/W: 加完後數值存到WREG
  - d = 1/F: 加完後數值存到指定位置(預設)

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    ; d = 1/F
    MOVLW 0x12
    MOVWF 0x00             ; [0x00] = 0x12
    MOVLW 0x23             ; [WREG] = 0x23
    ADDWF 0x00             ; [0x00] = 0x12 + 0x23 = 0x35

    ; d = 0/W
    MOVLW 0x12
    MOVWF 0x00             ; [0x00] = 0x12
    MOVLW 0x23             ; [WREG] = 0x23
    ADDWF 0x00, W          ; [WREG] = 0x12 + 0x23 = 0x35

    end                    ; 結束程式碼
```

- 實作迴圈

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    initial:
        CLRF 0x00
    start:                  ; 創立一個迴圈的標籤
        INCF 0x00
        GOTO start          ; 程式碼會回到start的下一行(line 10)

    end                     ; 結束程式碼
```

- DECFSZ: 將指定位置數值減一，若減完後為0則跳過下一行
  - 可用於控制迴圈次數

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    initial:
        MOVLW 0x04          ; start會做4次
        MOVWF 0x00
    start:
        DECFSZ 0x00
            GOTO start

    end                     ; 結束程式碼
```

- CPFSEQ: 比較WREG跟指定位置數值大小，若一樣就跳過下一行
  - 可用來實作if/else或控制迴圈

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    initial:
        MOVLW 0x15
        MOVWF 0x00          ; [0x00] = 0x15
        MOVLW 0x10          ; [WREG] = 0x10
    start:
        CPFSEQ 0x00
            INCF 0x01
        NOP                 ; 觀察用
    end                     ; 結束程式碼
```

- RRNCF: 把指定位置數值向右搬一格，最右邊一位搬到最左邊

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    initial:
        MOVLW b'01100110'
        MOVWF 0x00          ; [0x00] = b'01100010'
    Loop:
        RRNCF 0x00          ; [0x00] = b'00110001'
        GOTO Loop
    end                     ; 結束程式碼
```

- BTFSS/BTFSC: 檢查指定位置的某一位是0還是1，若是1/0則跳過下一行

```
List p=18f4520
    #include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
    org 0x00

    initial:
        MOVLW b'01100110'
        MOVWF 0x00        ; [0x00] = b'01100010'
    Loop:
        RRNCF 0x00
        BTFSS 0x00, 0     ; 檢查第0位是否是1(由右向左算，所以是最右邊那一
        GOTO Loop
    end                   ; 結束程式碼
```

## 補充

### 前言

指令中會看到有幾個參數，分別叫access bank與BSR， 這些會在之後的實驗詳細介紹，目前不會用到。 有興趣的同學可以先參考後面的資料學習。

### 描述PIC18的memory架構:

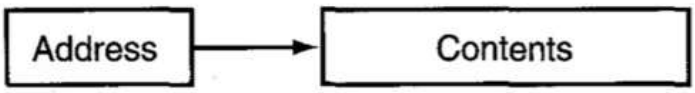
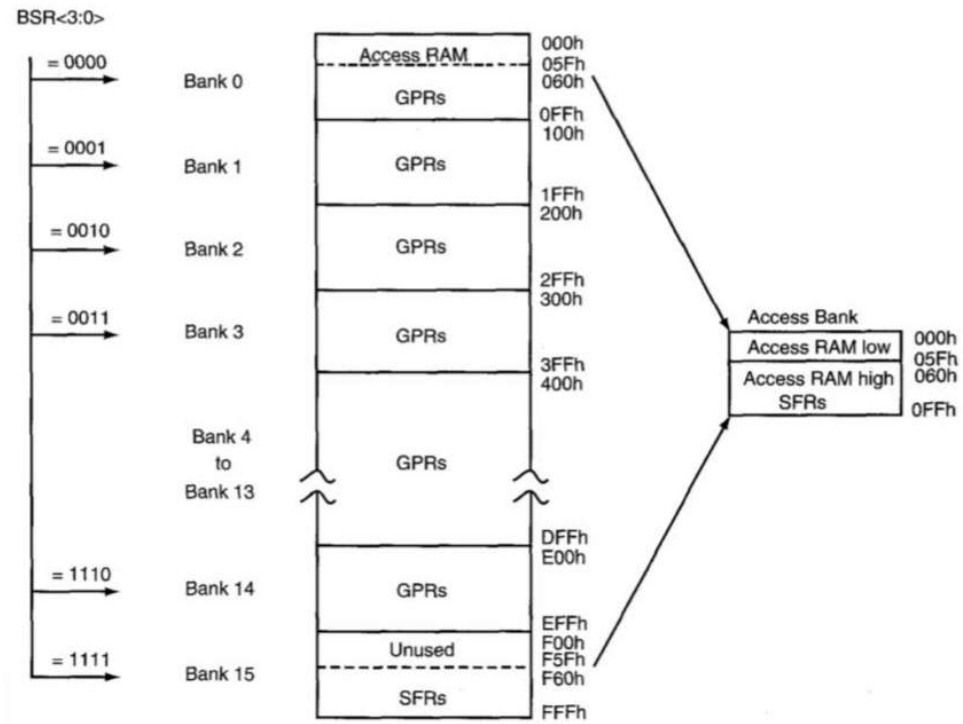
- The PIC18 Memory Organization
  - A memory location is referred to as an information unit.
  - A memory location in the PIC18 holds eight bits of information.
  - An information unit has two components: its address and its contents
  - 

    **Figure 1.2** ■ The components of a memory location

- Separation of Data Memory and Program Memory
  - The PIC18 MCU assigns data and program to different memory spaces

- PIC18 Data Memory
  - Each location in the data memory is also referred to as a register or file register
  - Supports 4096 bytes(8 bits) of data memory. It requires 12 bits of address to select one of the data registers. (要用12bits才能分辨現在用的是哪個register)
  - Because the limited length of the PIC instruction,only eight bits of the PIC18 instruction are used to specify the file register.
  - As a result, the PIC designers divided the 4096 file registers into 16 banks. Only one bank of 256 file registers is active at any time.
  - An additional four bits are placed in a special register called bank select register (BSR) to select the bank to be active.
  - 如果沒有指定BSR,通常就是預設access bank的register
  - 

    Note. 1. **BSR** is the 4-bit bank select register.

    **Figure 1.4** ■ Data memory map for PIC18 devices (redraw with permission of Microchip)

- Registers可以分成兩個種類:

- General-purpose registers (GPRs) hold dynamic data when the CPU is executing a prog. (運算的時候可以用來存放值、讀值...等等)

- Specialfunction registers (SFRs) control the desired operation of the MCU (就是可以有一些特殊用途,往後lab會慢慢去用到這些比較特別的register)