Lab3-Instruction Set

Instruction set:**http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/** (http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/)
Datasheet:**https://pdf1.alldatasheet.com/datasheet-pdf/view/112665/MICROCHIP/PIC18F4520.html**

(https://pdf1.alldatasheet.com/datasheet-pdf/view/112665/MICROCHIP/PIC18F4520.html)

大綱

- 指令種類
- 指令的構成要素
- 指令的參數、行為介紹

指令種類

指令種類共分為5類

1. ***Byte-oriented File Register Operations***
   操作的基本單元為1個Byte，為針對一整個Byte的操作，包含算術運算、邏輯運算以及簡單的比較、算術後Skip等，並有部分指令會額外使用到BSR、Status register的內容

2. ***Bit-oriented File Register Operations***
   操作的基本單元為1個Bit，為單獨對一個Byte的特定Bit做操作，包含歸零、設1、檢查0(or 1)後Branch，以及Toggle

3. ***Control Operations***
   此類指令用於流程控制，主要為各式Branch指令

4. ***Literal Operations***
   此類指令參數完全由常數構成，用於直接以常數對Register操作(絕大部分對WREG操作，僅有兩個指令對

另外兩個Register操作)

5. ***Data Memory & Program Memory Operations***
   此類指令為Data Memory及Program Memory間之互動、映射等(此學期Lab用不太到，因此不會深入解說)

指令的構成要素

在PIC18F4520中指令大多由2個Byte構成，少數指令(如MOVFF、GOTO、LFSR等)為4Byte
其中包含OPCode及指令所需的Operand
較為常見的Byte-oriented類指令通常由
1、 6bit OPCode + 1bit參數 + 1bit參數 + 8bit address
2、 7bit OPCode + 1bit參數 + 8bit address
這兩類構成
Bit-oriented類指令則由
4bit OPCode + 3bit定位欲操作的bit + 1bit參數 +8bit address構成
Control Operations、Literal Operations這兩類指令則多由
8bit OPCode + 8bit address or literal構成
當然，也有一些構成特殊的指令如BRA為5bit OPCode + 11 bit address或者如前面提到的4Byte指令，但上四種構成占絕大多數

指令的參數、行為介紹

在正式介紹指令之前我們先來了解一下BSR與STATUS REGISTER

REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

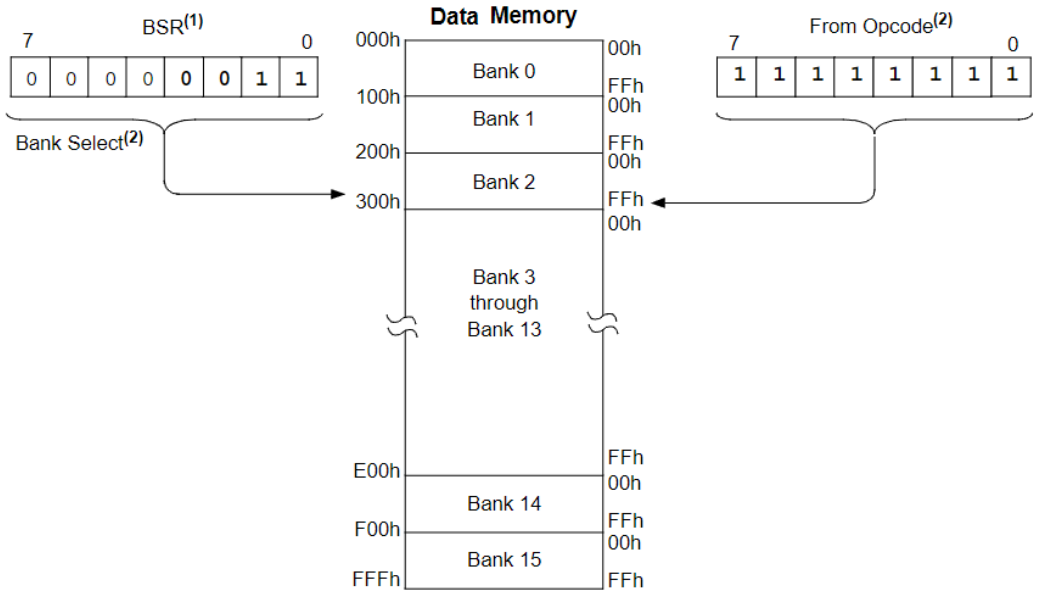
bit 1 **DC:** Digit Carry/$\overline{\text{borrow}}$ bit
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the source register.

STATUS REGISTER會儲存各式運算後的狀態，如運算後是否OVERFLOW、有沒有進位、結果是否為0、為負等



因為除MOVFF外所有牽涉到ADDRESS的指令都只能容納8 BITS，但DATA MEMORY卻有12 BITS，因此會需要BSR來儲存DATA MEMORY最左邊四個BIT，若指令中不使用BSR，則視為0

Byte-oriented File Register Operations

ADDWF — ADD W to f

Syntax: [*label*] ADDWF f [,d [,a]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01da	ffff	ffff

Description: Add W to register 'f'. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR is used.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWF REG, 0, 0`

Before Instruction
W = 0x17
REG = 0xC2

After Instruction
W = 0xD9
REG = 0xC2

ADDWFC — ADD W and Carry bit to f

Syntax: [*label*] ADDWFC f [,d [,a]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) + (C) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0010	00da	ffff	ffff

Description: Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWFC REG, 0, 1`

Before Instruction
Carry bit = 1
REG = 0x02
W = 0x4D

After Instruction
Carry bit = 0
REG = 0x02
W = 0x50

DECFSZ — Decrement f, skip if 0

Syntax: [*label*] DECFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0010	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). If the result is 0, the next instruction, which is already fetched, is discarded, and a `NOP` is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

CPFSEQ — Compare f with W, skip if f = W

Syntax: [*label*] CPFSEQ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
```

Example: HERE DECFSZ CNT, 1, 1
GOTO LOOP
CONTINUE

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE+2)

EQUAL :

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

MOVF Move f

Syntax: [*label*] MOVF f [,d [,a]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: f → dest

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W

Example: MOVF REG, 0, 0

Before Instruction
REG = 0x22
W = 0xFF
After Instruction
REG = 0x22
W = 0x22

MOVWF Move W to f

Syntax: [*label*] MOVWF f [,a]

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: (W) → f

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG, 0

Before Instruction
W = 0x4F
REG = 0xFF
After Instruction
W = 0x4F
REG = 0x4F

注意:MOVF預設為存回原REG，因此d參數需給0才能存入WREG

MOVFF — Move f to f

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'f_d' can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Note: The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled. See Section 8.0 for more information.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 0x33
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MULWF — Multiply W with f

Syntax: [*label*] MULWF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

0000	001a	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the status flags are affected.
Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: MULWF REG, 1

Before Instruction
W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

After Instruction
W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

RLCF — Rotate Left f through Carry

Syntax: [*label*] RLCF f [,d [,a]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

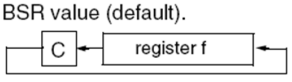
Operation: (f<n>) → dest<n+1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

0011	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

RLNCF — Rotate Left f (no carry)

Syntax: [*label*] RLNCF f [,d [,a]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

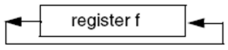
Operation: (f<n>) → dest<n+1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLCF REG, 0, 0`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

Decode	Read register 'f'	Process Data	Write to destination

Example: `RLNCF REG, 1, 0`

Before Instruction
REG = 1010 1011

After Instruction
REG = 0101 0111

Bit-oriented File Register Operations

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 0 → f<b>

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47

BTFSC **Bit Test File, Skip if Clear**

Syntax: [*label*] BTFSC f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

1011	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is 0, then the next instruction is skipped.
If bit 'b' is 0, then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BSF

BTFSS

BTG

注意:此處bit的編號為由左到右7~0，因此Left most bit為第7個bit，Right most bit為第0個bit

Control Operations

1. Conditional Branch

   此部分Branch指令須滿足特定條件(紀錄於Status Register)

BZ **Branch if Zero**

Syntax:	[*label*] BZ n
Operands:	-128 ≤ n ≤ 127
Operation:	if Zero bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0000 | nnnn | nnnn
Description:	If the Zero bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example:

```
HERE    BZ   Jump
```

Before Instruction

```
PC          =   address (HERE)
```

After Instruction

```
If Zero     =   1;
    PC      =   address (Jump)
If Zero     =   0;
    PC      =   address (HERE+2)
```

BC

BN

BNC

BNN

BNOV

BNZ

BOV

2. Unconditional Branch

此部分Branch指令無須滿足任何特定條件，直接Branch

BRA

GOTO

注意:BRA為2 Byte指令，而GOTO為4 Byte指令

Literal Operations

MOVLW **Move literal to W**

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	0000 1110 kkkk kkkk
Description:	The eight-bit literal 'k' is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `MOVLW 0x5A`

After Instruction

W = 0x5A

LFSR **Load FSR**

Syntax:	[*label*] LFSR f,k
Operands:	$0 \le f \le 2$ $0 \le k \le 4095$
Operation:	$k \rightarrow FSRf$
Status Affected:	None
Encoding:	1110 0000 00ff k_{11}kkk 1111 0000 $k_7$kkk kkkk
Description:	The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR 2, 0x3AB`

After Instruction

FSR2H = 0x03

FSR2L = 0xAB

此外還有...

MOVLB

ANDLW

IORLW

ADDLW

MULLW

RETLW

SUBLW

XORLW

下面附上解說影片程式碼

```
List p=18f4520
#include<p18f4520.inc>
    CONFIG OSC = INTIO67
    CONFIG WDT = OFF
org 0x00
start:
    MOVLW 0x02
    MOVWF 0x00
    ADDWF 0x00,0    ;d = 0,放回WREG
    ADDWF 0x00,1    ;d = 0,放回原F
    ADDWF 0x00,W
    ADDWF 0x00,F
    BSF STATUS,0
    BCF STATUS,0
    BSF STATUS,C
    ADDWFC 0x01       ;default,放回原F
    MOVLW 0x02
DECFSZ_TEST:
    DECFSZ WREG
    GOTO DECFSZ_TEST
    MOVLW 0x04
    MOVWF 0x02
    CLRF WREG
    MOVF 0x00
    MOVF 0x00,W
    MOVFF 0x02,0x03

    BSF STATUS,C
    MOVLW 0x81
    RLNCF WREG
    MOVLW 0x01
    RLNCF WREG
    MOVLW 0x81
    RLCF WREG
    MOVLW 0x01
    RLCF WREG

    CLRF WREG
BTFSS_TEST:
    BTFSS WREG,0
    GOTO func1
    GOTO func2
func1:
    BSF WREG,0
    GOTO BTFSS_TEST
func2:
    MOVLW 0x01
    SUBLW 0x02
    BZ Zero
Nzero:
    SUBLW 0x01
    BZ Zero
    GOTO BSR_TEST
Zero:
    GOTO BSR_TEST
BSR_TEST:
    MOVLW 0x10
    MOVWF 0x100
    MOVLB 0x01
    MOVWF 0x00,1
    MOVFF 0x100,0x101
    CLRF WREG
    MOVFF 0x000,0xFE8
Done:
end
```