Lab02 Addressing Mode

(1) What is Addressing Mode?

Link:IS for PIC18F4520 (http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/)

(2) PIC18F4520 Adressing Mode

No Operation

- NOP

Inherent Addressing (Implied Addressing)

- SLEEP、RESET、DAW

Literal Addressing (Immediate Addressing)

- MOVLW、ADDLW、SUBLW、ANDLW、GOTO、CALL...

Direct Addressing (Absolute Addressing)

Indirect Addressing

Bit Addressing

Relative addressing

No Operation

- NOP

NOP	No Operation			
Syntax:	[*label*] NOP			
Operands:	None			
Operation:	No operation			
Status Affected:	None			
Encoding:	0000 1111	0000 xxxx	0000 xxxx	0000 xxxx
Description:	No operation.			
Words:	1			
Cycles:	1			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

Sample code:

```
#INCLUDE <p18f4520.inc>
        CONFIG OSC = INTIO67
        CONFIG WDT = OFF
        org 0x10 ;PC = 0x10
start:
        nop
        nop
        nop
        nop
        nop
end
```

- PC += 2
- “wasting” 1 clock cycle
- delay loop

Inherent Addressing

- SLEEP、RESET、DAW

- SLEEP

SLEEP	Enter SLEEP mode
Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT postscaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	0000 0000 0000 0011
Description:	The power-down status bit ($\overline{PD}$) is cleared. The time-out status bit ($\overline{TO}$) is set. Watchdog Timer and its postscaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to sleep

- RESET

RESET	Reset
Syntax:	[*label*] RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.
Status Affected:	All
Encoding:	0000 0000 1111 1111
Description:	This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset in software.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: `RESET`

After Instruction

Registers = Reset Value
Flags* = Reset Value

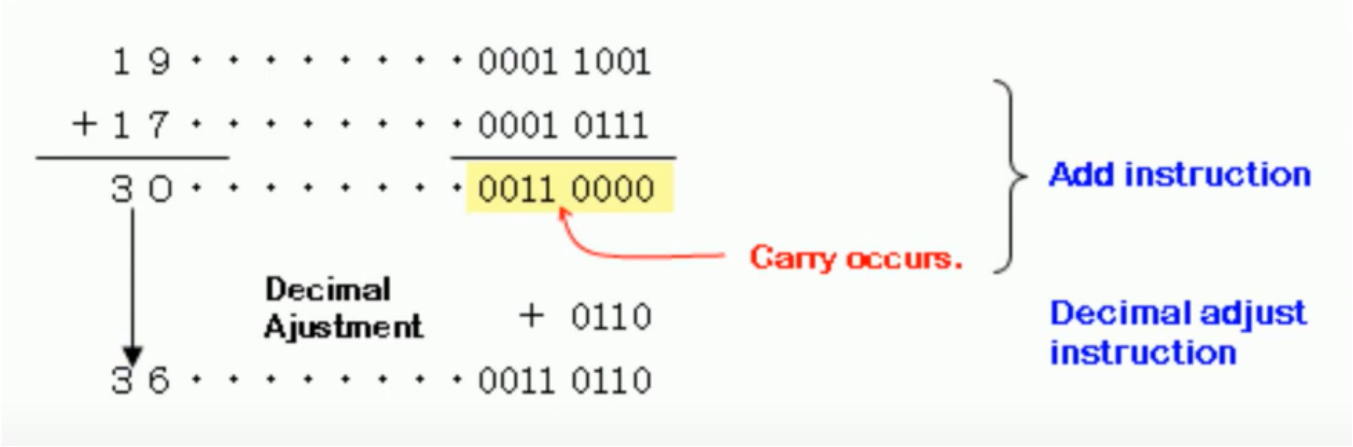
- DAW

DAW	Decimal Adjust W Register
Syntax:	[*label*] DAW
Operands:	None

Operation:	If [W<3:0> >9] or [DC = 1] then
	(W<3:0>) + 6 → W<3:0>;
	else
	(W<3:0>) → W<3:0>;
	
	If [W<7:4> >9] or [C = 1] then
	(W<7:4>) + 6 → W<7:4>;
	else
	(W<7:4>) → W<7:4>;

Status Affected:	C
Encoding:	0000 0000 0000 0111
Description:	DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.
Words:	1
Cycles:	1
Q Cycle Activity:	

- BCD addition adjustment



Literal Addressing

- 8-bits literal MOVLW、ADDLW、SUBLW、ANDLW
- 20-bits literal *GOTO...

Sample code:

```
#INCLUDE <p18f4520.inc>
        CONFIG OSC = INTIO67
        CONFIG WDT = OFF
        org 0x10 ;PC = 0x10
start:
    MOVLW 0x05

end
```

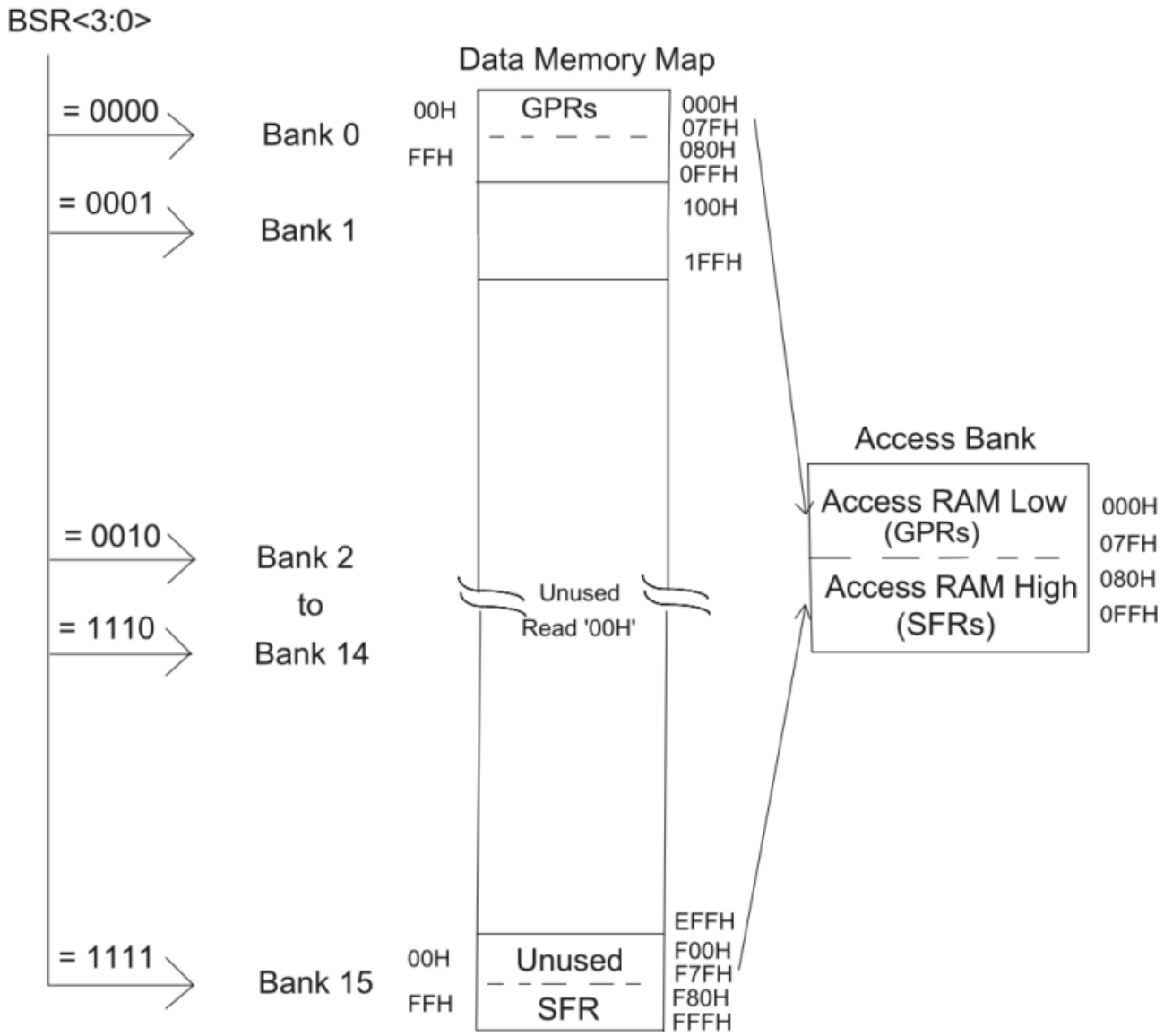
SFRs ×

Address	Name	Hex	Decimal	Binary	Char
FE0	BSR	0x00	0	00000000	'.'
FE1	FSR1	0x0000	0	00000000 00000000	'..'
FE1	FSR1L	0x00	0	00000000	'.'
FE2	FSR1H	0x00	0	00000000	'.'
FE3	PLUSW1	0x00	0	00000000	'.'
FE4	PREINC1	0x00	0	00000000	'.'
FE5	POSTDEC1	0x00	0	00000000	'.'
FE6	POSTINC1	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FE8	WREG	0x05	5	00000101	'.'
FE9	FSR0	0x0000	0	00000000 00000000	'..'
FE9	FSR0L	0x00	0	00000000	'.'
FEA	FSR0H	0x00	0	00000000	'.'
FEB	PLUSW0	0x00	0	00000000	'.'
FEC	PREINC0	0x00	0	00000000	'.'
FED	POSTDEC0	0x00	0	00000000	'.'
FEE	POSTINC0	0x00	0	00000000	'.'
FEF	INDF0	0x00	0	00000000	'.'
FF0	INTCON3	0xC0	192	11000000	'À'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF2	INTCON	0x00	0	00000000	'.'
FF3	PROD	0x0000	0	00000000 00000000	'..'
FF3	PRODL	0x00	0	00000000	'.'

Memory SFRs Format Individual

Direct Addressing

- Data Memory MAP



12-bits memory address, higher 4 bits for bank select.
==> 將整塊4096Bytes的記憶體區分成16個小區(bank0~bank15)

- Some data movement instructions on file register
  - *Access Bank
    - Access RAM (or GPRs) (0x000 ~ 0x07F and 0xF80 ~ 0xFFF)
  - Bank Select (higher address) (0x000 ~ 0xFFF)

(*) 因為 data movement 的 file register 欄位只有 8 個 bits，所以 Access Bank 方式只能存取 256 個 bytes 的記憶體空間(0x000 ~ 0x07F 和 0xF80 ~ 0xFFF)。
因此若要存取整個 4096-bytes 大小的記憶體空間，要使用 Bank Select 的方式存取。

- MOVWF

MOVWF	**Move W to f**
Syntax:	[*label*] MOVWF f [,a]
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$(W) \rightarrow f$
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'

- MOVLB (Use MOVLB to select bank)

MOVLB	**Move literal to low nibble in BSR**
Syntax:	[*label*] MOVLB k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow BSR$
Status Affected:	None
Encoding:	0000 0001 kkkk kkkk
Description:	The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLB 5`

Before Instruction
BSR register = 0x02
After Instruction
BSR register = 0x05

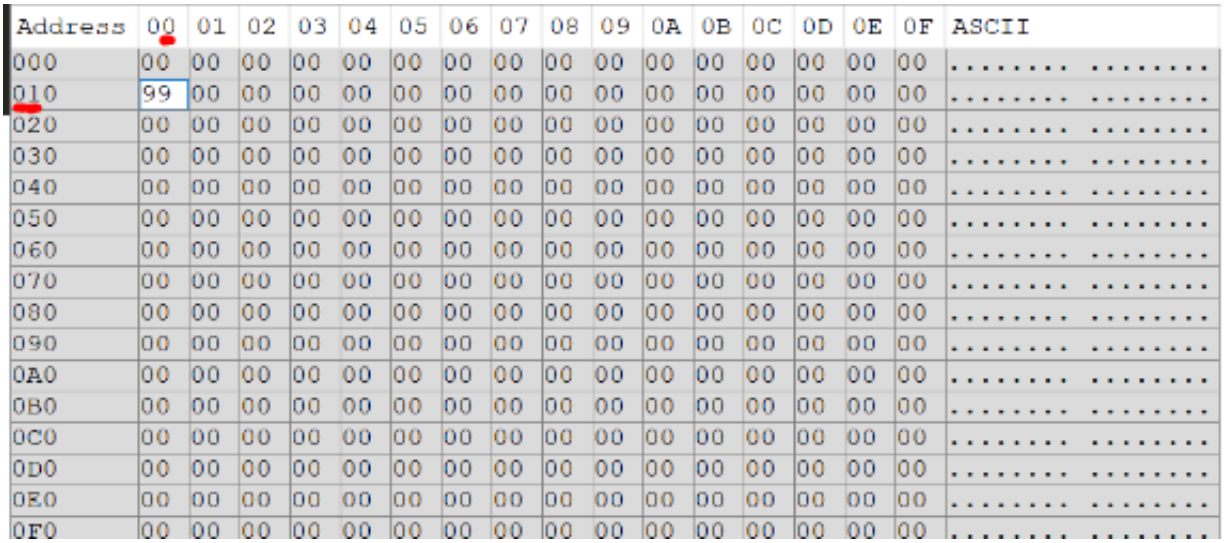
- MOVFF

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) 2nd word (destin.)	1100 1111	ffff ffff	ffff ffff	ffff$_s$ 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			

MOVFF 是一個很特別的指令，他的指令格式是給你兩個 12-bits 的 file register 欄位去填，所以在整個 4096-bytes 大小的記憶體裡，想去哪就去哪，不需要在乎bank這件事，也就是這個指令可以隨意在 4096-bytes 大小的記憶體空間裡存取。

Sample code: Access Bank

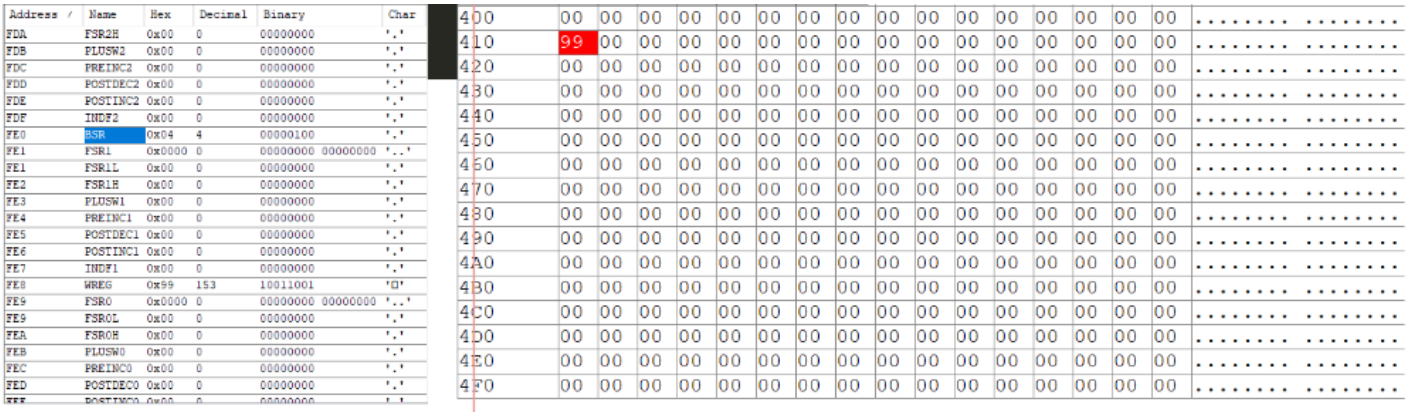
```
#INCLUDE <p18f4520.inc>
        CONFIG OSC = INTIO67
        CONFIG WDT = OFF
        org 0x10 ;PC = 0x10
start:
        MOVLW 0x99 ; WREG = 0x99
        MOVWF 0x10 ; [0x010] = 0x99
end
```



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
010	99	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Sample code: Bank Select

```
#INCLUDE <p18f4520.inc>
        CONFIG OSC = INTIO67
        CONFIG WDT = OFF
        org 0x10 ;PC = 0x10
start:
        MOVLW 0x99 ; WREG = 0x99
        MOVLB 0x4 ; BSR = 4
        MOVWF 0x10, 1 ; use BSR select bank ; [0x410] = 0x99
end
```



Address	Name	Hex	Decimal	Binary	Char
FDA	FSR2H	0x00	0	00000000	'.'
FDB	PLUSW2	0x00	0	00000000	'.'
FDC	PREINC2	0x00	0	00000000	'.'
FDD	POSTDEC2	0x00	0	00000000	'.'
FDE	POSTINC2	0x00	0	00000000	'.'
FDF	INDF2	0x00	0	00000000	'.'
FE0	BSR	0x04	4	00000100	'.'
FE1	FSR1	0x0000	0	00000000 00000000	'..'
FE1	FSR1L	0x00	0	00000000	'.'
FE2	FSR1H	0x00	0	00000000	'.'
FE3	PLUSW1	0x00	0	00000000	'.'
FE4	PREINC1	0x00	0	00000000	'.'
FE5	POSTDEC1	0x00	0	00000000	'.'
FE6	POSTINC1	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FE8	WREG	0x99	153	10011001	'Ö'
FE9	FSR0	0x0000	0	00000000 00000000	'..'
FE9	FSR0L	0x00	0	00000000	'.'
FEA	FSR0H	0x00	0	00000000	'.'
FEB	PLUSW0	0x00	0	00000000	'.'
FEC	PREINC0	0x00	0	00000000	'.'
FED	POSTDEC0	0x00	0	00000000	'.'

400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
410	99	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
420	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Sample code: MOVFF

```
1  #INCLUDE <p18f4520.inc>
2          CONFIG OSC = INTIO67
3          CONFIG WDT = OFF
4          org 0x10 ;PC = 0x10
5  start:
6          MOVLW 0x99 ; WREG = 0x99
7          MOVLB 0x4 ; BSR = 4
8          MOVWF 0x10, 1 ; use BSR select bank ; [0x410] = 0x99
9          MOVFF 0x410, 0x420 ; [0x420] = 0x99
10 end
```

400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
410	99	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
420	99	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
500	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Indirect Addressing

Three SFRs call FSRx (x for 0~2)

- FSR0、FSR1、FSR2

- 16bits (FSRxH : FSRxL) to cover all data memory address
- they are pointer
- LFSR (let FSRx point to memory address k)

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 1111	1110 0000	00ff k_7kkk	$k_{11}kkk$ 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

Some SFRs related to pointer FSRx (x for 0~2)

- INDFx：指針不變，對指向的記憶體位置進行操作
- POSTINCx：對指向的記憶體位置進行操作後，指針 + 1
- POSTDECx：對指向的記憶體位置進行操作後，指針 - 1
- PREINCx：指針 + 1 後，對指向的記憶體位置進行操作
- PLUSWx：指針 + WREG = 新的記憶體位置後，對指向的記憶體位置進行操作

Sample code:

```
#INCLUDE <p18f4520.inc>
 CONFIG OSC = INTIO67
 CONFIG WDT = OFF
 org 0x00 ;PC = 0x00
setup1:
 LFSR 0, 0x000 ; FSR0 point to 0x000
 LFSR 1, 0x010 ; FSR1 point to 0x010
 LFSR 2, 0x020 ; FSR2 point to 0x020
 MOVLW 0x10 ; WREG = 0x10
start:
 INCF POSTINC0
 ; [0x000] += 1; FSR0 point to 0x001

 INCF PREINC1
 ; FSR1 point to 0x011 ;[0x011] += 1

 INCF POSTDEC2
 ; [0x020] += 1 ; FSR2 point to 0x01F

 INCF INDF2
 ; [0x01F] += 1 ;
 ; FSR2 point to 0x01F(unchanged)

 INCF PLUSW2
 ; [0x01F+0x10] += 1
 ; FSR2 point to 0x01F(unchanged)
end
```

Bit Addressing

- BSF、BCF、BTFSC、BTFSS
  - Set or clear specific bit file register
  - BSF：Bit set f
  - BCF：Bit clear f
  - BTFSC：Bit test f skip if clear
  - BTFSS：Bit test f skip if set

Sample code:

```
1  #INCLUDE <p18f4520.inc>
2          CONFIG OSC = INTIO67
3          CONFIG WDT = OFF
4          org 0x10 ;PC = 0x10
5  start:
6      BSF 0x000, 1 ; [0x000] = b'00000010
7      BTFSC 0x000, 0 ; test bit 0 of [0x000], skip if bit 0 is 0
8      MOVFF 0x000, 0x001
9      BTFSC 0x000, 1 ; test bit 1 of [0x000], skip if bit 1 is 0
10     MOVFF 0x000, 0x001
11 end
```

Address	Symbol	Hex	Decimal	Binary	Char
000		0x02	2	00000010	'.'
001		0x00	0	00000000	'.'
002		0x00	0	00000000	'.'
003		0x00	0	00000000	'.'
004		0x00	0	00000000	'.'

Address	Symbol	Hex	Decimal	Binary	Char
000		0x02	2	00000010	'.'
001		0x02	2	00000010	'.'
002		0x00	0	00000000	'.'
003		0x00	0	00000000	'.'
004		0x00	0	00000000	'.'

Relative Addressing

- BC、BN、BNC、BNN、BNZ(branch if not zero) ...
- for all the “Branch” instruction to addressing
- relative address to PC value (branch的下一行)
- offset is word address(for PIC18F4520 1 word = 2Bytes)
- if branch: PC = PC + 2 + n * 2 (2’s complement)

if not branch: PC = PC + 2

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	Process	Write to PC

- Example:

```
Memory
address  Op code
                                  1:          #INCLUDE<P18F4321.INC>
                                  2:          ORG       0x00
0000     0E02   MOVLW   0x2     3:  BACK    MOVLW     0x02
0002     0802   SUBLW   0x2     4:          SUBLW     0x02
0004     E001   BZ      0x8     5:          BZ        DOWN
0006     0E04   MOVLW   0x4     6:          MOVLW     0x04
0008     0804   SUBLW   0x4     7:  DOWN    SUBLW     0x04
000A     E0FA   BZ  0           8:          BZ        BACK
000C     0003   SLEEP           9:          SLEEP
```