PIC18F4520
Assembly language programming (III)

Hong-hen Lin
林弘恒

Networked Embedded Applications and Technologies Lab

Department of Computer Science and Information Engineering
National Cheng Kung University, TAIWAN

Outline

- ❑ Multiplier
- ❑Stack
- ❑Subroutine
- ❑Macro
- ❑Lab

- ❑ Multiplier
- ❑ Stack
- ❑ Subroutine
- ❑ Macro
- ❑ Lab

Hardware Multiplier

- All PIC18 devices include an 8 x 8 hardware multiplier

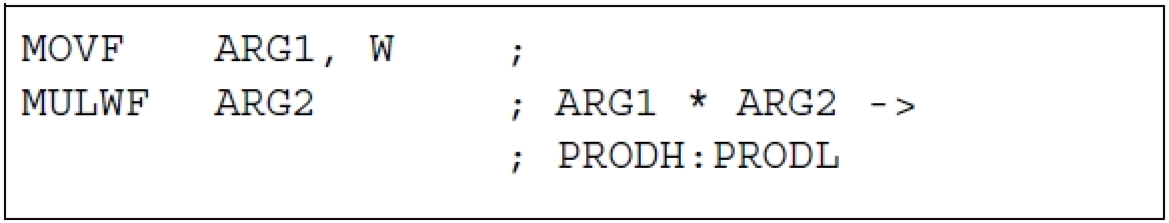
- Yields a 16-bit result that is stored in the product register pair, PRODH:PRODL

[The multiplier’s operation does not affect any flags in the Status register]

- Making multiplication a hardware operation allows it to be completed in a single instruction cycle.

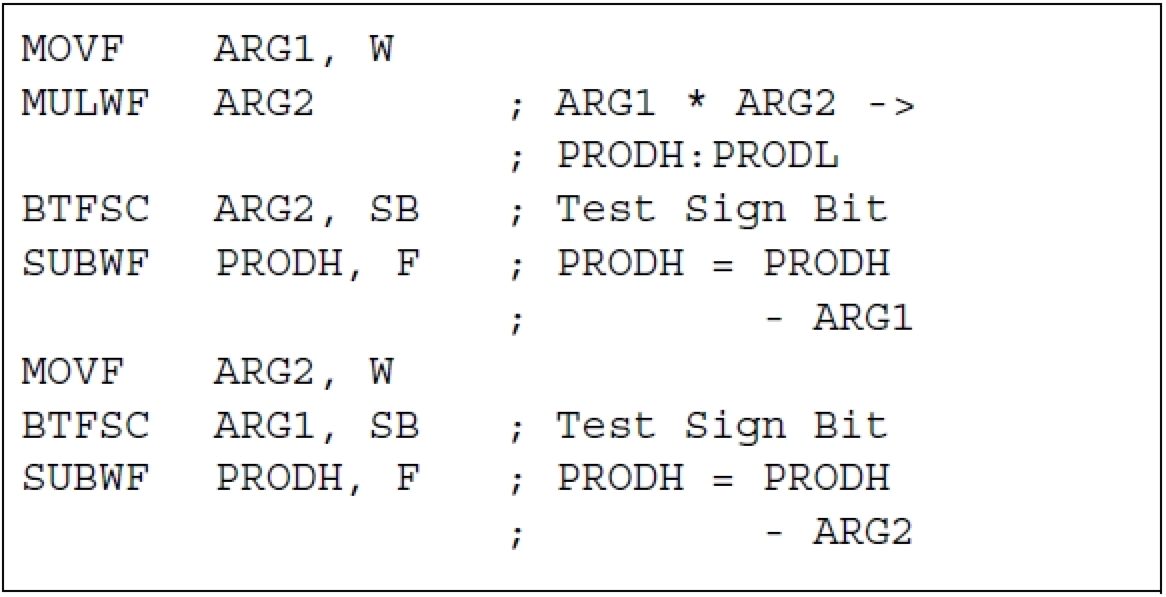
8x8 MLTIPLY ROUTINE

❑ UNSIGENED



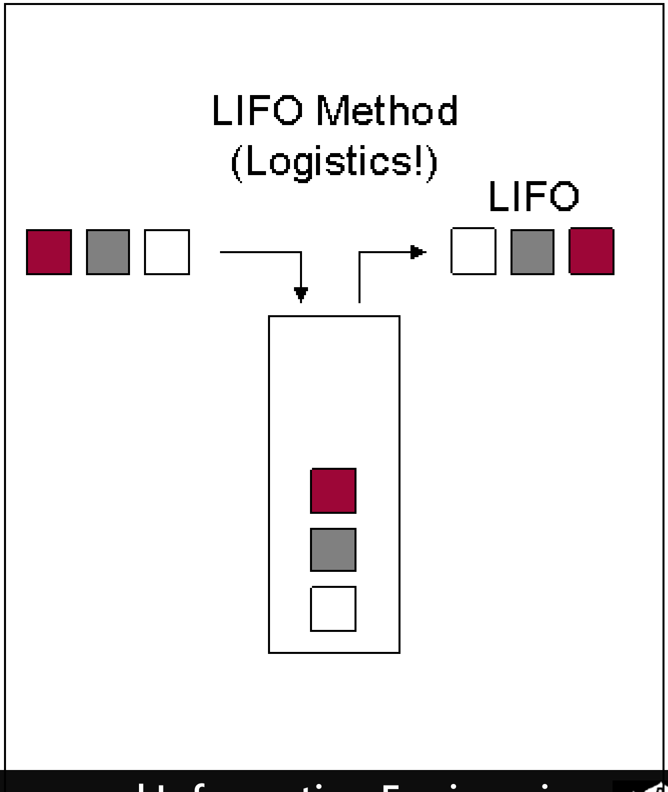
```
MOVF    ARG1, W      ;
MULWF   ARG2         ; ARG1 * ARG2 ->
                     ; PRODH:PRODL
```

❑ SIGENED

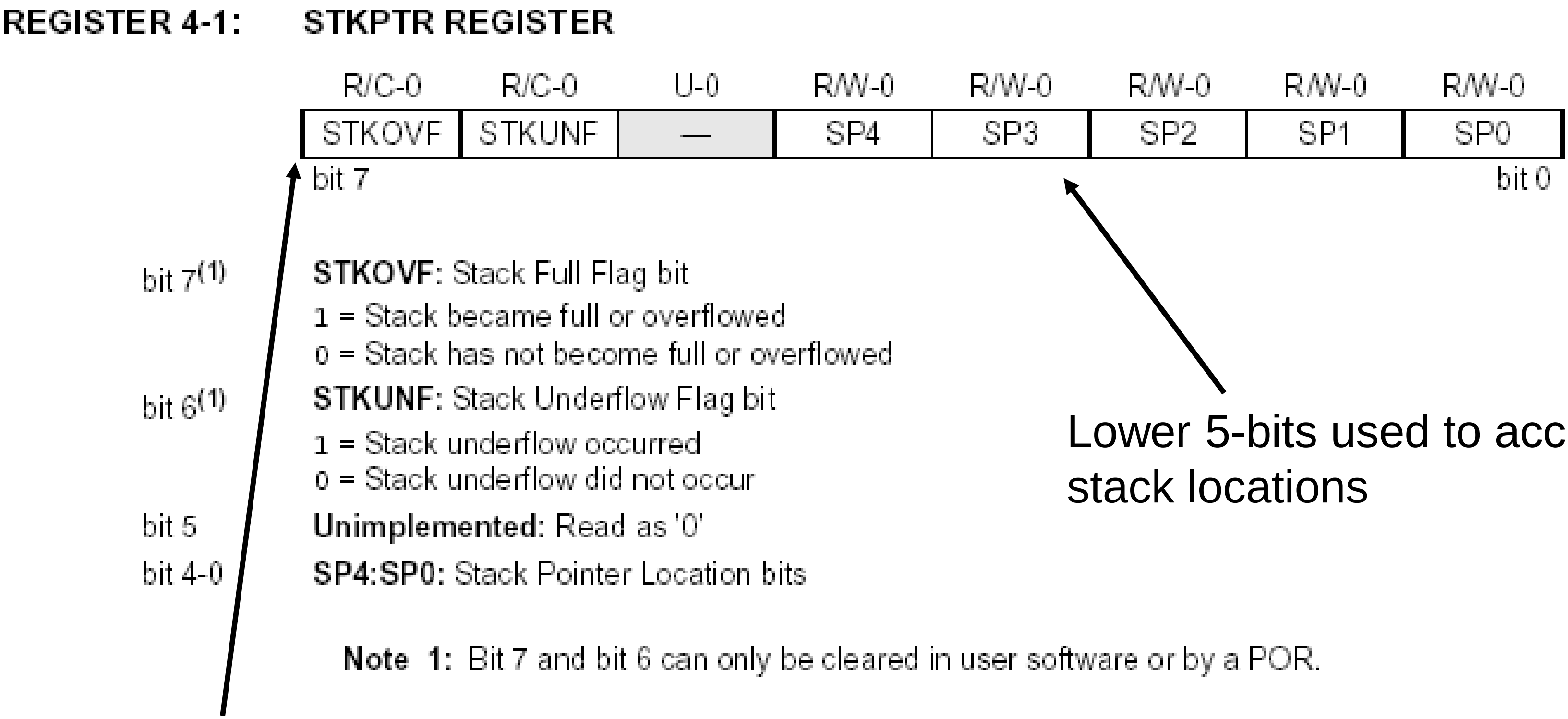


```
MOVF    ARG1, W
MULWF   ARG2         ; ARG1 * ARG2 ->
                     ; PRODH:PRODL
BTFSC   ARG2, SB     ; Test Sign Bit
SUBWF   PRODH, F     ; PRODH = PRODH
                     ;         - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB     ; Test Sign Bit
SUBWF   PRODH, F     ; PRODH = PRODH
                     ;         - ARG2
```

- ❑ Multiplier
- ❑ **Stack**
- ❑ Subroutine
- ❑ Macro
- ❑ Lab



STKPTR Register



REGISTER 4-1: STKPTR REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKOVF	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7[(1)] **STKOVF:** Stack Full Flag bit
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6[(1)] **STKUNF:** Stack Underflow Flag bit
1 = Stack underflow occurred
0 = Stack underflow did not occur

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

bit 4-0 **SP4:SP0:** Stack Pointer Location bits

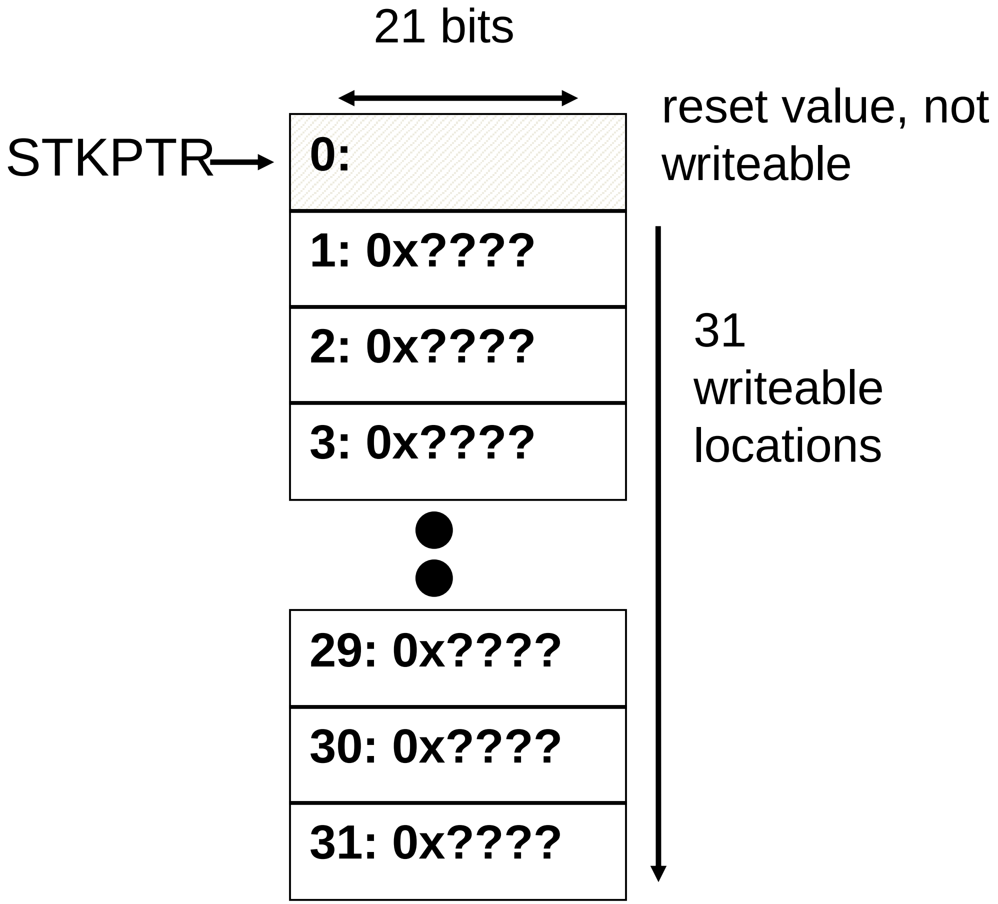
Note 1: Bit 7 and bit 6 can only be cleared in user software or by a POR.

Lower 5-bits used to access stack locations

Upper two bits are the stack overflow and underflow status bits.

The PIC18 Stack

The PIC18 stack has limited capability compared to other μPs. It resides within its on memory, and is limited to 31 locations.



21 bits

STKPTR

0:

reset value, not writeable

1: 0x????

2: 0x????

3: 0x????

29: 0x????

30: 0x????

31: 0x????

31 writeable locations

For a call, address of next instruction (nPC) is *pushed* onto the stack

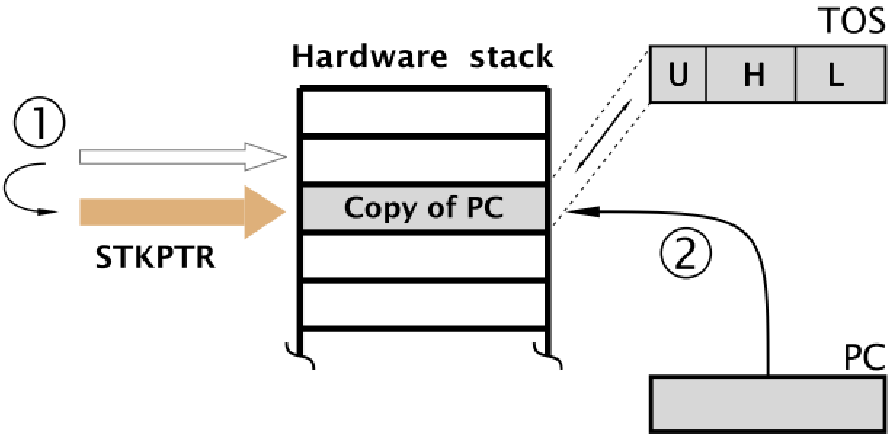
PUSH & POP

A **PUSH**

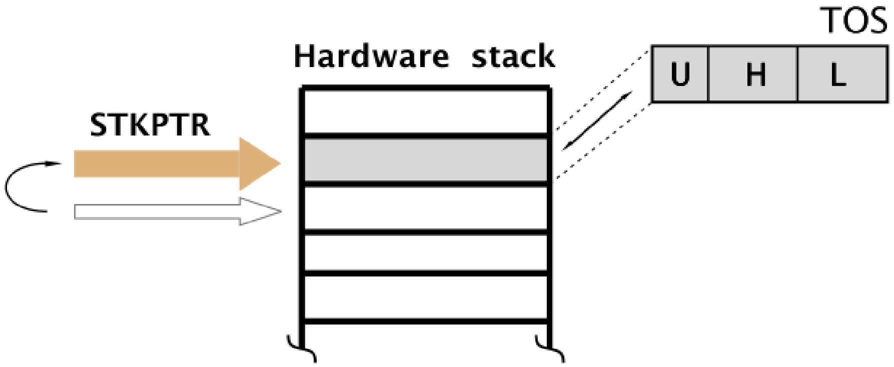
STKPTR++; [STKPTR] ← nPC

A **POP**

(PC ←[STKPTR], STKPTR--)

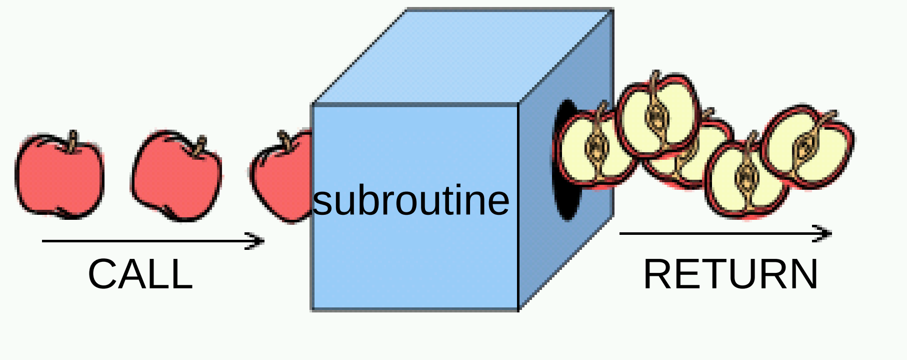


(a) push



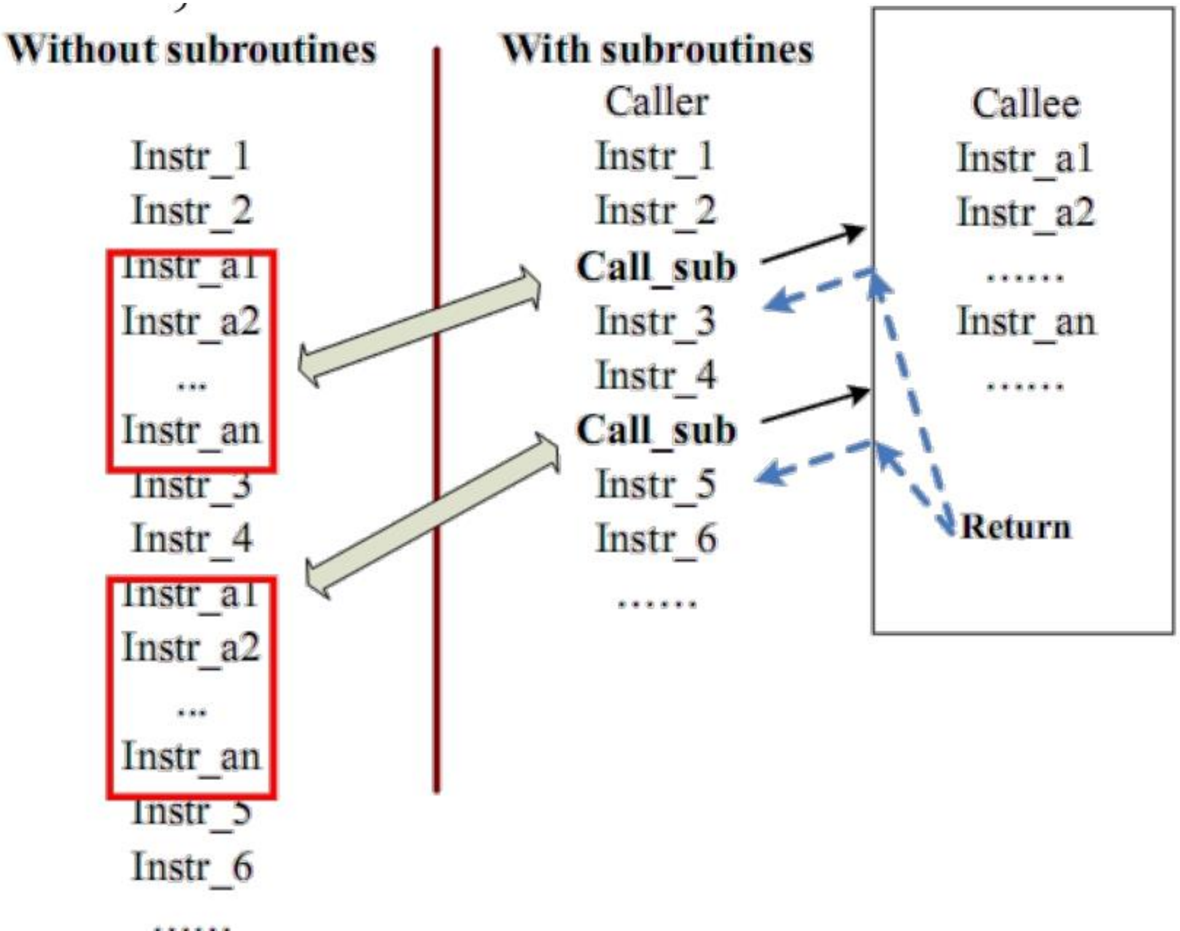
(b) pop

- ❑ Multiplier
- ❑ Stack
- ❑ Subroutine
- ❑ Macro
- ❑ Lab



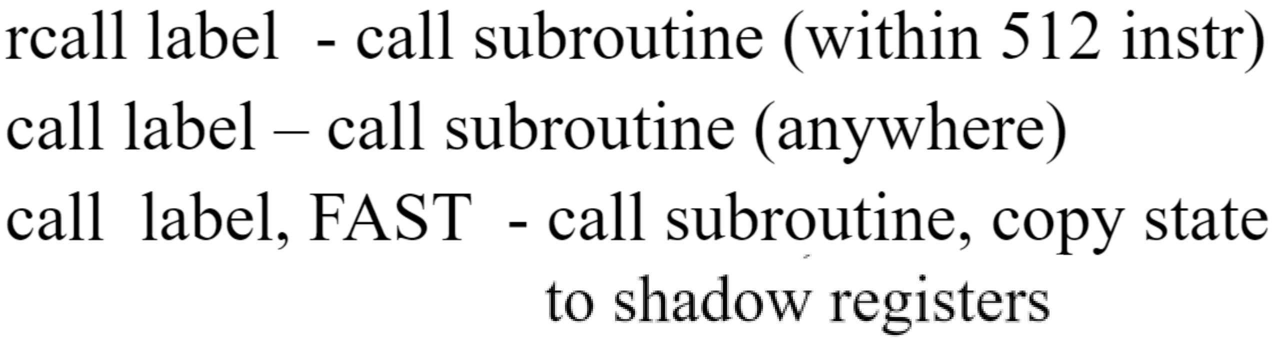
Subroutine Introduction

A subroutine is a section of code, or program, than can be called as and when you need it. Subroutines are used if you are performing the same function more than once.



Subroutine handling instruction

- CALL

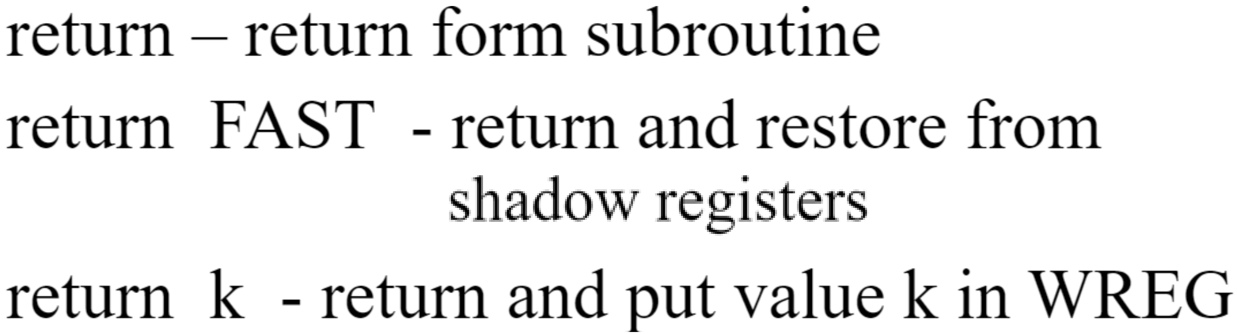


rcall label - call subroutine (within 512 instr)

call label – call subroutine (anywhere)

call label, FAST - call subroutine, copy state
to shadow registers

- Return

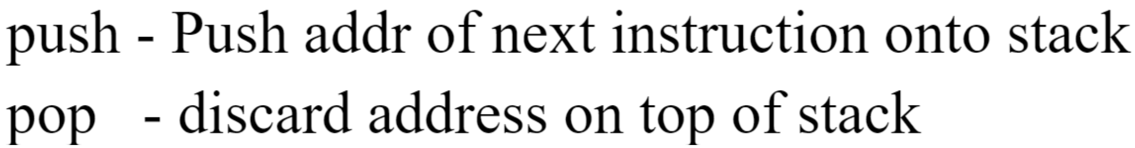


return – return form subroutine

return FAST - return and restore from
shadow registers

return k - return and put value k in WREG

- Stack



push - Push addr of next instruction onto stack

pop   - discard address on top of stack

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 02

Stack pointer: 00

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	--	
02	--	
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label         Instruction
00        START:       MOVLW H'2'
02                     MOVWF LATA
04                     RCALL FIRST
06                     NOP
08                     NOP
10       FIRST:        MOVLW H'3'
12                     MOVWF LATB
14                     RCALL MORE
16                     INCF LATB
18                     RCALL ALSO
20                     RETURN
22       MORE:         MOVFF LATA, LATC
26                     RETURN
28       ALSO:         MOVFF LATB,LATD
32                     RETURN
                       END
```

Register:

Program counter: 04

Stack pointer: 00

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	--	
02	--	
03	--	
04	--	

Subroutine-sample1.asm

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label       Instruction
00        START:     MOVLW H'2'
02                   MOVWF LATA
04                   RCALL FIRST
06                   NOP
08                   NOP
10       FIRST:      MOVLW H'3'
12                   MOVWF LATB
14                   RCALL MORE
16                   INCF LATB
18                   RCALL ALSO
20                   RETURN
22       MORE:       MOVFF LATA, LATC
26                   RETURN
28       ALSO:       MOVFF LATB,LATD
32                   RETURN
                     END
```

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

Register:

Program counter: 10

Stack pointer: 01

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	--	
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label       Instruction
00        START:     MOVLW H'2'
02                   MOVWF LATA
04                   RCALL FIRST
06                   NOP
08                   NOP
10       FIRST:      MOVLW H'3'
12                   MOVWF LATB
14                   RCALL MORE
16                   INCF LATB
18                   RCALL ALSO
20                   RETURN
22       MORE:       MOVFF LATA, LATC
26                   RETURN
28       ALSO:       MOVFF LATB,LATD
32                   RETURN
                     END
```

Register:

Program counter: 12

Stack pointer: 01

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	--	
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
  LIST p=18f4520
  #include <p18f4520.inc>
  ORG 0X00
  Address  Label      Instruction
  00        START:    MOVLW H'2'
  02                  MOVWF LATA
  04                  RCALL FIRST
  06                  NOP
  08                  NOP
  10       FIRST:     MOVLW H'3'
  12                  MOVWF LATB
→ 14                  RCALL MORE
  16                  INCF LATB
  18                  RCALL ALSO
  20                  RETURN
  22       MORE:      MOVFF LATA, LATC
  26                  RETURN
  28       ALSO:      MOVFF LATB,LATD
  32                  RETURN
                      END
```

Register:

Program counter: 14

Stack pointer: 01

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
→ 01	06	NOP
02	--	
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label       Instruction
00        START:     MOVLW H'2'
02                   MOVWF LATA
04                   RCALL FIRST
06                   NOP
08                   NOP
10       FIRST:      MOVLW H'3'
12                   MOVWF LATB
14                   RCALL MORE
16                   INCF LATB
18                   RCALL ALSO
20                   RETURN
→ 22     MORE:       MOVFF LATA, LATC
26                   RETURN
28       ALSO:       MOVFF LATB,LATD
32                   RETURN
                     END
```

Register:

Program counter: 22

Stack pointer: 02

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
→ 02	16	INCF LATB
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 26

Stack pointer: 02

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	16	INCF LATB
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 16

Stack pointer: 01

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	16	INCF LATB
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 18

Stack pointer: 01

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	16	INCF LATB
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 28

Stack pointer: 02

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	20	RETURN
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
20                  RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 32

Stack pointer: 02

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	20	RETURN
03	--	
04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label      Instruction
00        START:    MOVLW H'2'
02                  MOVWF LATA
04                  RCALL FIRST
06                  NOP
08                  NOP
10       FIRST:     MOVLW H'3'
12                  MOVWF LATB
14                  RCALL MORE
16                  INCF LATB
18                  RCALL ALSO
→ 20                RETURN
22       MORE:      MOVFF LATA, LATC
26                  RETURN
28       ALSO:      MOVFF LATB,LATD
32                  RETURN
                    END
```

Register:

Program counter: 20

Stack pointer: 01

Stack(in memory):

	Address	RetAdrs.	Location
	00	0	STKPTR
→	01	06	NOP
	02	20	RETURN
	03	--	
	04	--	

Subroutine-sample1.asm

STKPTR的更動不會將值給POP或PUSH，除非裡用POP、PUSH指令才會，否則值依舊在裡面，不然就是被更新為別的值

```
LIST p=18f4520
#include <p18f4520.inc>
ORG 0X00
Address  Label        Instruction
00        START:      MOVLW H'2'
02                    MOVWF LATA
04                    RCALL FIRST
06                    NOP
08                    NOP
10       FIRST:       MOVLW H'3'
12                    MOVWF LATB
14                    RCALL MORE
16                    INCF LATB
18                    RCALL ALSO
20                    RETURN
22       MORE:        MOVFF LATA, LATC
26                    RETURN
28       ALSO:        MOVFF LATB,LATD
32                    RETURN
                      END
```

Register:

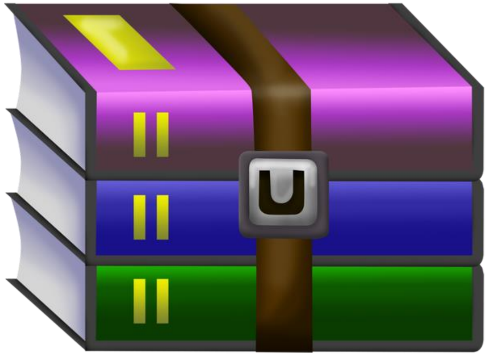
Program counter: 06

Stack pointer: 00

Stack(in memory):

Address	RetAdrs.	Location
00	0	STKPTR
01	06	NOP
02	20	RETURN
03	--	
04	--	

- ❑ Multiplier
- ❑ Stack
- ❑ Subroutine
- ❑ **Macro**
- ❑ Lab

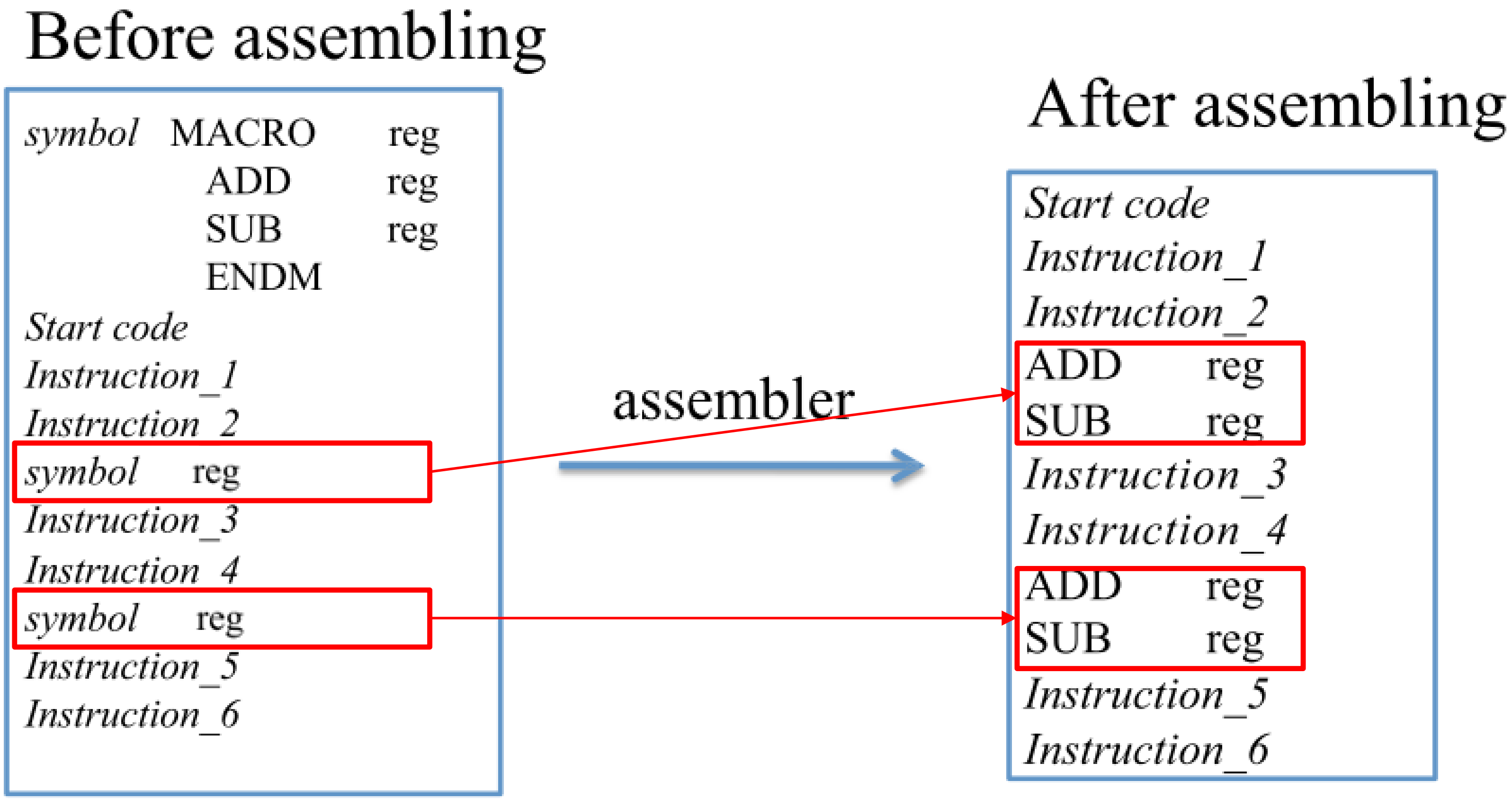


MACRO

- What is Macro?
  - Replace a sequence of instructions by a macro name
  - Make more productive
  - Make the program more readable

- MACRO
  - SYNTAX: name MACRO [,argument]
- ENDM
  - SYNTAX: ENDM
- LOACL
  - SYNTAX: LOCAL symbol [,symbol] …
  - 因為在程式中很有可能會定義到某個label與在Macro內所定義的label同名,這時候就要用Local宣告定義該label

The result of Macro



Before assembling

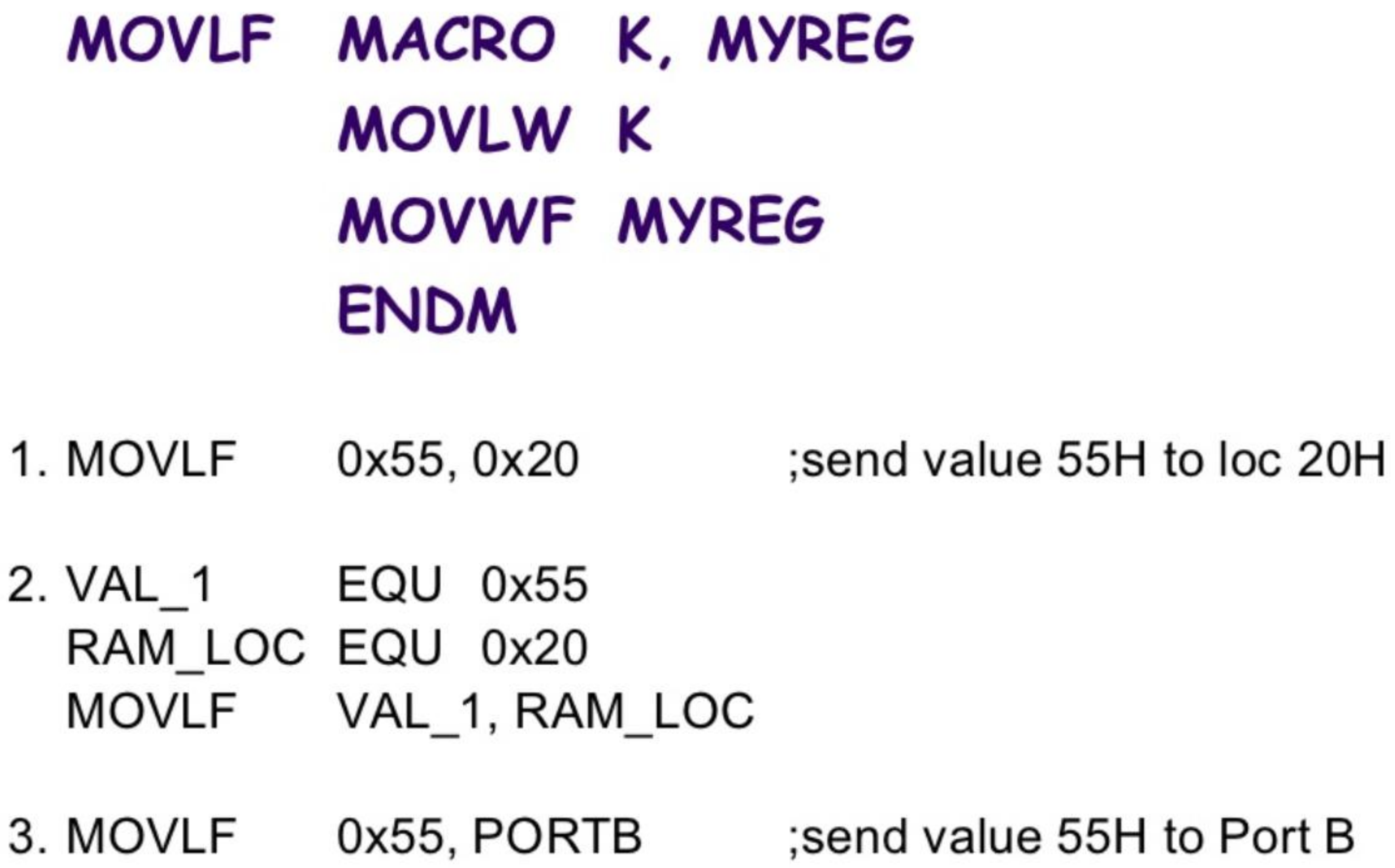
```
symbol   MACRO     reg
           ADD     reg
           SUB     reg
           ENDM
Start code
Instruction_1
Instruction_2
symbol    reg
Instruction_3
Instruction_4
symbol    reg
Instruction_5
Instruction_6
```

assembler

After assembling

```
Start code
Instruction_1
Instruction_2
ADD      reg
SUB      reg
Instruction_3
Instruction_4
ADD      reg
SUB      reg
Instruction_5
Instruction_6
```

MACRO-sample



```
MOVLF   MACRO   K, MYREG
        MOVLW   K
        MOVWF   MYREG
        ENDM
```

```
1. MOVLF      0x55, 0x20          ;send value 55H to loc 20H

2. VAL_1      EQU   0x55
   RAM_LOC    EQU   0x20
   MOVLF      VAL_1, RAM_LOC

3. MOVLF      0x55, PORTB         ;send value 55H to Port B
```

Macro vs subroutine

	Macro	Subroutine
方式	直接把程式碼取代symbol	跳到symbol位置往下執行
程式碼大小	較大	較小
執行速度	較快	較慢
	適合寫較小塊的block	適合寫較長的副程式

lab會遇到的bug

- MACRO

```
ADDFF MACRO PAR1, PAR2
          MOVF PAR1,W
          ADDWF PAR2
          ENDM
```

```
LOOP:   CPFSGT TRIS
        ADDFF TRISA,TRISB
        GOTO LOOP
```

assembler

```
LOOP:   CPFSGT TRIS
        MOVF TRISA,W
        ADDWF TRISB
        GOTO LOOP
```

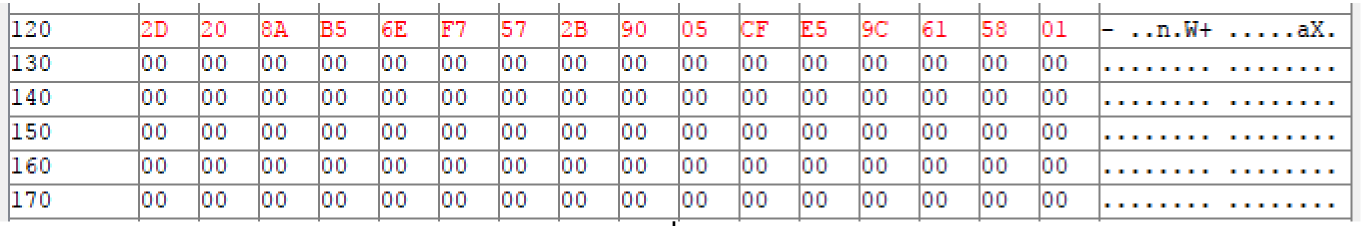
- SUBROUTINE

```
LOOP:   CPFSGT TRIS
        RCALL ADDLF
        GOTO LOOP
ADDLF:  MOVF TRISA,W
        ADDWF TRISB
        RETURN
```

- ❑ Multiplier
- ❑ Stack
- ❑ Subroutine
- ❑ Macro
- ❑ Lab

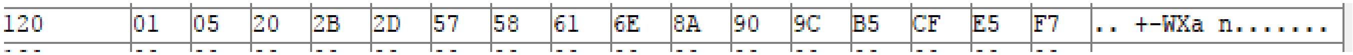
Lab4-Bubble sort

- 請用組語的方式寫出泡沫排序法,將120-12F的16個數字由小到大排列
- 內迴圈的swap請另外用macro寫



```
120   2D 20 8A B5 6E F7 57 2B 90 05 CF E5 9C 61 58 01   -  ..n.W+ .....aX.
130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ........ ........
140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ........ ........
150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ........ ........
160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ........ ........
170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ........ ........
```

Result:



```
120   01 05 20 2B 2D 57 58 61 6E 8A 90 9C B5 CF E5 F7   .. +-WXa n.......
```

Lab4-Bubble sort

- Hint1:利用間接定址法,將hint2的i,j視為address
- Hint2: i = 120, j = 120 ;

MAIN

SBRT subroutine

```
do {
    j=i;
   do {
        if ( num[i] > num[j] )
          swap( num[i], num[j] ) ;  // 請用macro寫
        j++;
   } while ( j < 16 )
    i++;
 } while ( i < 16)
```

參考資料

- PIC18F4520 datasheet
  - http://ww1.microchip.com/downloads/en/devicedoc/39631a.pdf
- PIC18F4520 instruction set
  - http://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/
- Microchip 教材 102ASP
  - http://www.microchip.com.tw/Data_CD/Workshop/8-Bits/102ASP%20PIC18F452.zip
- Macro&subroutine資料
  - http://www.romux.com/tutorials/pic-tutorial/macros-and-subprograms