The PIC Instruction Set

The PIC 16F87X Series Instruction Set (Complete!)

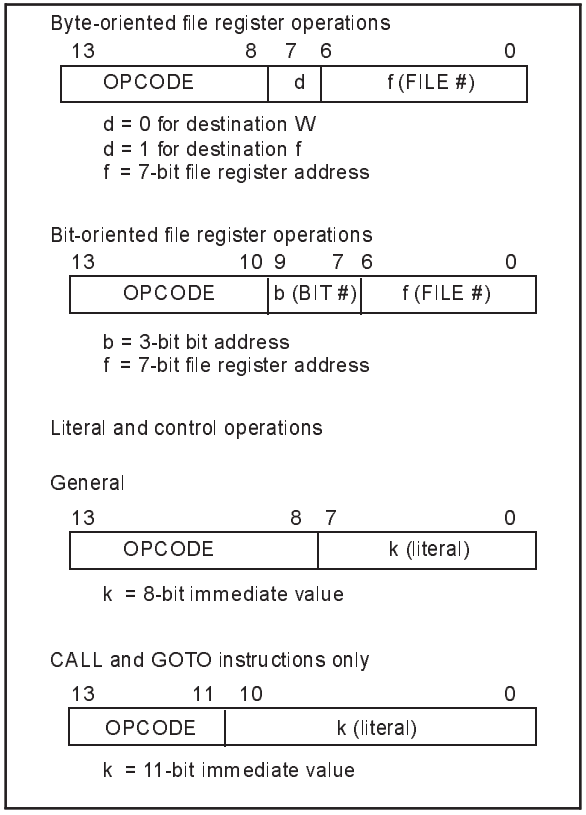
Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Figuring Out The Instruction Syntax

The PIC instruction set is nice, in that once you figure out one instruction you've figured out most of them, but it is a little different than what we're used to. The "secret codes" and how instructions are formed is described in the following tables:

TABLE 13-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
PC	Program Counter
TO	Time-out bit
PD	Power-down bit



Comparing The Old And The New

Let's look at two equivalent programs. One on our old friend, the 80x86, the other on this new-fangled PIC Microcontroller.

Consider the following x86 program:

```
      .Model  small
      .Data
var1  db    33h
var2  db    0A4h
var3  db    ?

      .Code
      .8086
start: mov  ax, @Data ;Set data segment.
       mov  ds, ax

       mov  al, var1  ;Set Al to 33h
       mov  bl, var2  ;Set BL to A4h
       add  al, bl    ;Compute al = al + bl
       mov  var3, al  ;Move sum to memory

stop:  jmp  stop      ;Stop program
end     start
```

A Complete, But Simple PIC Program

```
; Our first PIC Processor Example
;
        include p16f877.inc ;Include file for register definitions

; *****************************************************************
; equates
Bank0Ram   equ     0x20    ;Equates mean the same as before!
; *****************************************************************
; variables - Remember, these are in register file memory!
        cblock  Bank0Ram
        var1            ; 0x20 locations used for variables
        var2            ; 0x21
        var3            ; 0x22 temp location used for counter
        endc

; *****************************************************************
; start at the reset vector

        org  0x000      ; Set origin at memory address 000
        nop             ; very important - required for debugger
Start   bcf  STATUS,RP0 ; go to BANK 0 by setting
        bcf  STATUS,RP1 ; RP1:RP0 to 0 0.

; Initialize contents of variables.
;
      movlw  0x33 ; Move literal 33h into W.
      movwf  var1 ; Move from W to var1 (initialize var1)
      movlw  0A4h ; Move literal A4h into W.
      movwf  var2 ; Move from W to var2 (initialize var2)

;
; Do the addition.
      movf   var1,W     ; Get data from var1 into W.
      addwf  var2,W     ; Calculate W = var1 + var2.
      movwf  var3       ; Store result in var3.

stop  goto   stop
      End
```

The Hardest Thing About The PIC!

The hardest thing to understand about the PIC is the conditional jump. There are only two jump instructions, but the way they work is a little strange. Hang on!

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b < 7$
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

A Simple Decrement Loop On The PIC

```
; Our second PIC Processor Example
;
        include p16f877.inc ;Include file for register definitions

; *****************************************************************
; equates
count   equ     0x25    ;Manually assign register address!
 ; *****************************************************************
; start at the reset vector

        org  0x000      ; Set origin at memory address 000
        nop             ; very important - required for debugger
Start   bcf  STATUS,RP0 ; go to BANK 0 by setting
        bcf  STATUS,RP1 ; RP1:RP0 to 0 0.

; Initialize contents of variables.
;
        movlw  0x0A     ; Initialize count to 10 decimal by
        movwf  count    ;   setting W and moving W -> count.

;
; Do the simple loop.
loop    decf   count,F  ; count = count - 1.
        btfsc  STATUS,Z ; Skip next instruction if Z = 0.

stop    goto   stop     ; End program.

        goto   loop     ; Continue looping!

     End
```