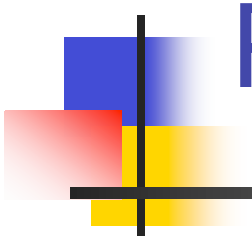


PIC Microcontroller - 4

PIC Instruction Set (II)

TABLE 13-2: PIC16F87X INSTRUCTION SET

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	

Conditional Branch (1)

STATUS bits:

none

```
btfsc   f, b    ;Test bit b of register f, where b=0 to 7, skip if clear
btfss   f, b    ;Test bit b of register f, where b=0 to 7, skip if set
```

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 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.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ 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.

Examples:

- btfsc TEMP1, 0 ; Skip the next instruction if bit 0 of TEMP1 equals 0
- btfss STATUS, C ; Skip the next instruction if C==1

Conditional Branch (2)

STATUS bits:

none

decfsz f, F(W) ;decrement f, putting result in F or W, skip if zero
incfsz f, F(W) ;increment f, putting result in F or W, skip if zero

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

Examples:

- decfsz TEMP1, F ; Decrement TEMP1, skip if TEMP1==0
- incfsz TEMP1, W ; W <- TEMP1+1 , skip if W==0 (TEMP1==H'FF')
  ; Leave TEMP1 unchanged

Simple Operations (1)

Either-Or sequence

```
            btfsc    STATUS, Z        ;Test Z bit, skip if clear
            goto     Zset
Zclear
            :                         ; Instructions to execute, if Z==0
            goto     Zdone
Zset
            :                          ; Instructions to execute, if Z==1
Zdone                                  ; Carry on
```

Compare two numbers (unsigned), if NUM1<NUM2, run code Below, otherwise run code Above

80x86

```
    mov     ah, NUM1
    cmp     ah, NUM2
    jb      Below
Above:
    …
    jmp     CmpDone
Below:
    …
CmpDone:
```

PIC

```
    movf    NUM2, W     ; NUM2->W
    subwf   NUM1, W     ; NUM1 - W -> W
    btfss   STATUS, C    ; C==1 indicates no borrow, NUM1 >= NUM2
    goto    Below       ; C==0 indicates borrow, NUM1<NUM2
Above
    …
    goto    CmpDone
Below
    …
CmpDone
```

Simple Operations (2)

Assume a 16-bit counter, the upper byte of the counter is called COUNTH and the lower byte is called COUNTL.

Decrement a 16-bit counter

```
        movf    COUNTL, F       ; Set Z if lower byte == 0
        btfsc   STATUS,  Z
        decf    COUNTH, F       ; if so, decrement COUNTH
        decf    COUNTL, F       ; in either case decrement COUNTL
```

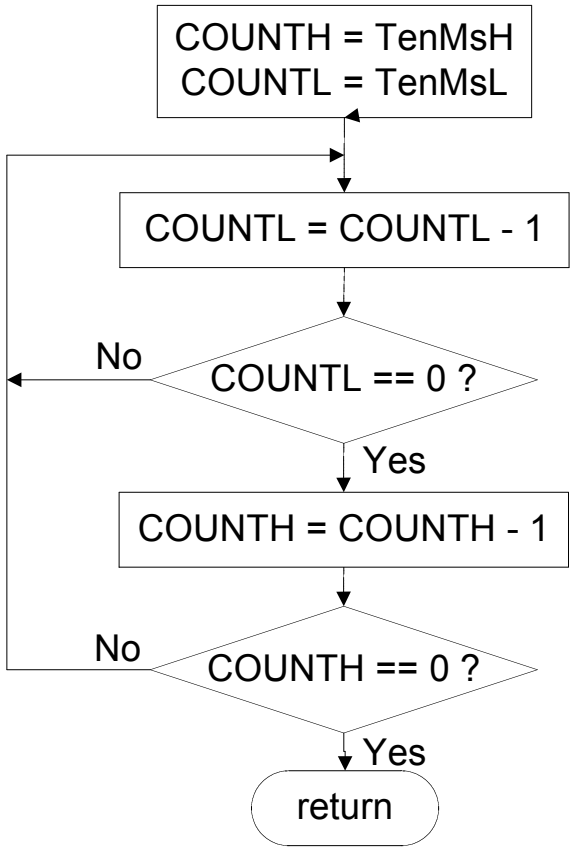
Test a 16-bit variable for zero

```
        movf    COUNTL, F       ; Set Z if lower byte == 0
        btfsc   STATUS, Z       ; If not, then done testing
        movf    COUNTH, F       ; Set Z if upper byte == 0
        btfsc   STATUS, Z       ; if not, then done
        goto    BothZero        ; branch if 16-bit variable == 0
CarryOn
```

A Delay Subroutine

```
; ********************************************************************************
; TenMs subroutine and its call inserts a delay of exactly ten milliseconds
; into the execution of code.
; It assumes a 4 MHz crystal clock.
; TenMsH    equ  13       ; Initial value of TenMs Subroutine's counter
; TenMsL    equ  250
; COUNTH and COUNTL are two variables

TenMs
      nop                          ; one cycle
      movlw  TenMsH                ; Initialize COUNT
      movwf  COUNTH
      movlw  TenMsL
      movwf  COUNTL
Ten_1
      decfsz  COUNTL,F             ; Inner loop
      goto    Ten_1
      decfsz  COUNTH,F             ; Outer loop
      goto    Ten_1
      return
```

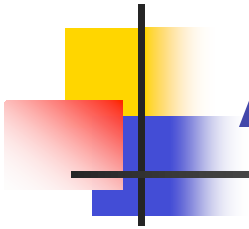


```
                                                                                                  Cycles
call    TenMs                                                                                        2
nop                                                                                                  1
movlw   13      (TenMsH)                                                                             1
movwf   COUNTH                                                                                       1
movlw   250     (TenMsL)                                                                             1
movlw   COUNTL                                                                                       1


decfsz  COUNTL,F  }  COUNTL: 250→249→...→2→1                          3* x 249          =        747
goto    Ten_1     }
decfsz  COUNTL,F     COUNTL: 1→0                                                                     2
decfsz  COUNTH,F     COUNTH: 13→12                                                                   1
goto    Ten_1                                                                                        2


decfsz  COUNTL,F  }  COUNTL: 0→255→254→...→2→1     3 x 255 = 765   }
goto    Ten_1     }                                                 }
decfsz  COUNTL,F     COUNTL: 1→0                              2     }   770** x 11        =       8470
decfsz  COUNTH,F     COUNTH: 12→11                            1     }
goto    Ten_1                                                2     }
                                                        ---------  }
                                                            770    }
        Repeat this block eleven times as COUNTH: 12→11→...→2→1


decfsz  COUNTL,F  }  COUNTL: 0→255→254→...→2→1                        3 x 255           =        765
goto    Ten_1     }
decfsz  COUNTL,F     COUNTL: 1→0                                                                     2
decfsz  COUNTH,F     COUNTH: 1→0                                                                     2
return                                                                                               2
                                                                                            ---------
                                                                                  Total  =     10,000
```



Acknowledgement

Some slides are revised based on lecture notes used in WPI ECE 2801