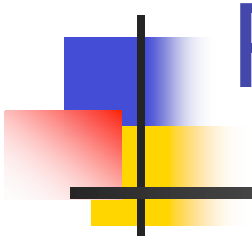
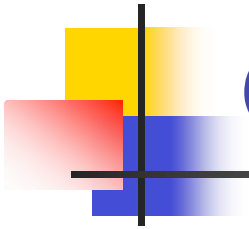


# PIC Microcontroller - 3

## PIC Instruction Set (I)

# Review: Data Memory Organization

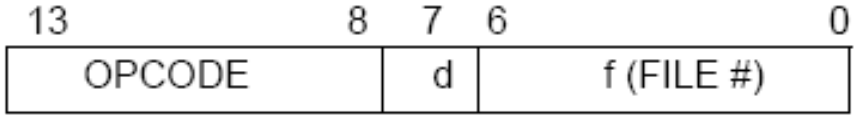


- Made up of SFRs and GFRs
- Banking: 7Fh (128) bytes chunks
- Addressing Modes
  - Direct addressing:
    - 7 bit address (within bank)
    - RP1:RP0 selects bank
  - Indirect addressing:
    - Access to INDF causes indirect addressing
    - Actual memory address in IRP+FSR
- Special Function Registers
  - W, PC (PCL+PCLATH), FSR, INDF, STATUS
  - I/O Ports

# PIC Instructio

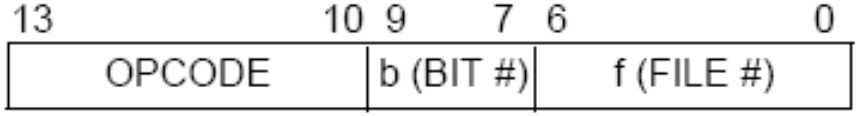
- 35 instructions
- Each instruction is 1 word long (14 bits)
- Byte-oriented OPCODE f, F(W)
  - Source f: name of a SFR or a RAM variable
  - Destination F(W):
    - F if the destination is to be the same as the source register
    - W if the destination is to be the working register
- Bit-oriented OPCODE f, b
  - Bit address b (0≤b≤7)
- Literal and control OPCODE k
  - Literal value k

Byte-oriented file register operations




d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

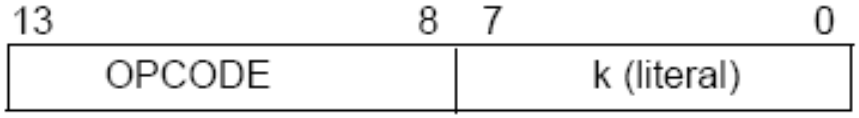
Bit-oriented file register operations




b = 3-bit bit address
f = 7-bit file register address

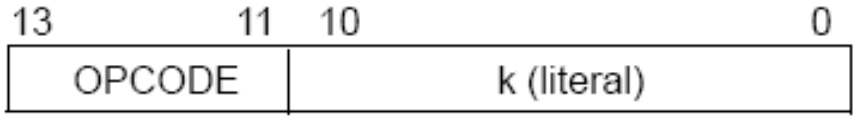
Literal and control operations

General




k = 8-bit immediate value

CALL and GOTO instructions only




k = 11-bit immediate value

## 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 | |

# Single Bit Manipulation

bcf f,b

Operation: Clear bit b of register f, where b=0 to 7

STATUS bits:

none

bsf f,b

Operation: Set bit b of register f, where b=0 to 7

Examples:

```
bcf   PORTB, 0        ;Clear bit 0 off PORTB
bsf   STATUS, C       ;Set the Carry bit
bcf   STATUS, RP1     ;
bsf   STATUS, RP0     ;Select Bank 1
```

# Clear/Move

```
clrw                    ; Clear W register
clrf   f                ; Clear f register
movlw  k                ; move literal value k to W
movwf  f                ; move W to f
movf   f, f(W)                  ; move f to F or W
swapf  f, F(W)                  ; swap nibbles of f, putting result in F or
                                W
```

STATUS bits:
clrw, clrf, movf: Z
movlw, movwf, swapf: none

Examples:

- clrf TEMP1 ;Clear variable TEMP1
- movlw 5 ;load 5 into W
- movwf TEMP1 ;move W into TEMP1
- movwf TEMP1, F ;Incorrect Syntax
- movf TEMP1, W ;move TEMP1 into W ;
- movf TEMP1, TEMP2 ;Incorrect Syntax
- swapf TEMP1, F ;Swap 4-bit nibbles of TEMP1
- swapf TEMP1, W ;Move TEMP1 to W, swap nibbles, leave TEMP1 unchanged

# Increment/Decrement/Complement

```
incf    f, F(W)          ; increment f, putting result in F
   or W
decf    f, F(W)     ;decrement f, putting result in F or W
comf    f, F(W)     ;complement f, putting result in F or
   W
```

STATUS bits:

Z

Examples:

- incf TEMP1, F ;Increment TEMP1
- incf TEMP1, W ;W <- TEMP1+1; TEMP1 unchanged
- decf TEMP1, F ;Decrement TEMP1
- comf TEMP1, F ;Change 0s and 1s to 1s and 0s

# Multi-bit Manipulation

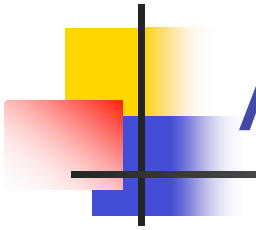
```
andlw   k          ; AND literal value k into W
andwf   f, F(W)    ; AND W with F, putting result in F or W
iorlw   k          ; Inclusive-OR literal value k into W
iorwf   f, F(W)    ; Inclusive-OR W with F, putting result in F or W
xorlw   k          ; Exclusive-OR literal value k into W
xorwf   f, F(W)    ; Exclusive-OR W with F, putting result in F or W
```

Examples:

```
andlw   B'00000111'   ;force upper 5 bits of W to zero
andwf   TEMP1, F      ;TEMP1 <- TEMP1 AND W
andwf   TEMP1, W      ;W <- TEMP1 AND W
iorlw   B'00000111'   ;force lower 3 bits of W to one
iorwf   TEMP1, F      ;TEMP1 <- TEMP1 OR W
xorlw   B'00000111'   ;complement  lower 3 bits of W
xorwf   TEMP1, W      ;W <- TEMP1 XOR W
```

STATUS bits:

Z



# Addition/Subtraction

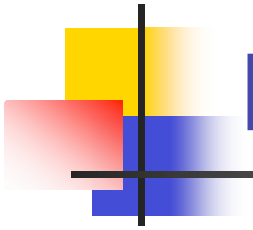
```
addlw    k          ;add literal value k into W
addwf    f, F(W)    ;add w and f, putting result in F or W
sublw    k          ;subtract W from literal value k, putting
                    ;result in W
subwf    f, F(W)    ;subtract W from f, putting result in  F or
   W
```

Examples:

- addlw 5 ; W <= 5+W
- addwf TEMP1, F ; TEMP1 <- TEMP1+W
- sublw 5 ; W <= 5-W (not W <= W-5 )
- subwf TEMP1, F ; TEMP1 <= TEMP1 - W

STATUS bits:

C, DC, Z



# Rotate

```
rlf    f, F(W)   ; copy f into F or W; rotate F or W left through
                 ; the carry bit
rrf    f, F(W)   ; copy f into F or W; rotate F or W right
   through
                 ; the carry bit
```

STATUS bits:

C

Examples:

- rlf TEMP1, F ; C <- TEMP1.bit(7), TEMP1.bit(i+1) <- TEMP1.bit(i)
  ; TEMP1.bit(0) <- C
- rrf TEMP1, W ; Copy TEMP1 to W and rotate W right through C
  ; TEMP1 unchanged

# Goto/Call/Retrun/Return from Interrupt

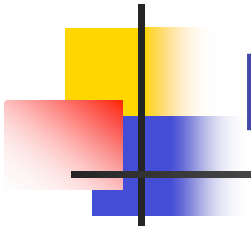
```
goto     label      ; Go to labeled instruction
call     label      ; Call labeled subroutine
return              ; Return from subroutine
retlw    k          ; Return from subroutine, putting literal
                    ; value in W
retfie              ; Return from interrupt service routine;
                    ; re-enable interrupts
```

STATUS bits:

none

Examples:

```
goto     There      ; Next instruction to be executed is labeled “There”
call     Task1      ; Push return address; Next instruction to be
                    ; executed is labeled “Task1”
return              ; Pop return address off of stack
retlw    5          ; Pop return address; W <- 5
retfie              ; Pop return address; re-enable interrupts
```



# Miscellaneous

```
clrwdt      ; clear watchdog timer
sleep       ; go into standby mode
nop         ; no operation
```

STATUS bits:
clrwwdt, sleep:
NOT_TO, NOT_PD
nop: none

Examples:

- clrwdt ; if watchdog timer is enabled, this instruction will reset
  ; it (before it resets the CPU)
- sleep ; Stop clock; reduce power; wait for watchdog timer or
  ; external signal to begin program execution again
- nop ; Do nothing; wait one clock cycle

# Comparing The Old and The New

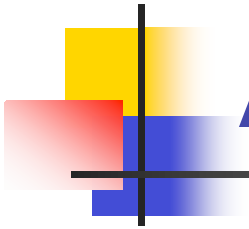
```
        .model small
        .data
var1    db      0A4h
var2    db      22h
var3    db      ?
        .code
        .8086
start:  mov     ax, @data       ;set data segment
        mov     ds, ax

        mov     al, var1        ; set  AL to 0A4h
        mov     bl, var2        ; set BL to 22h
        add     al, bl          ; AL<- AL + BL
        mov     var3, al        ; mov sun to memory location var3

stop:   jmp     stop
        end     start
```

# An Equivalent PIC Program

```
; PIC Processor Example
        include p16f877.inc   ;Include file for register definitions
; ********  Equates ***************************************************
Bank0Ram    equ             0x20    ;Equates mean the same as before!
; ******** Variables ***************************************************
        cblock Bank0Ram
        var1        ; 0x20 locations used for variables
        var2        ; 0x21
        var3        ; 0x22 temp location used for counter
        endc
; ******** Vectors *****************************************************
        org         0x000           ; Set origin at memory address 000
        goto        Mainline
        org         0x004
iStop   goto        iStop
; ******** Mainline Program ********************************************
Mainline
        bcf         STATUS,RP0      ; go to BANK 0 by setting
        bcf         STATUS,RP1      ; RP1:RP0 to 0 0.
;
        movlw       0xA4            ; Move literal A4h into W.
        movwf       var1            ; Move from W to var1 (initialize var1)
        movlw       0x22            ; Move literal 22h into W.
        movwf       var2            ; Move from W to var2 (initialize var2)
;
        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
```



# Acknowledgement

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