PIC18Fxx2 Instruction Set

55:036

Spring, 2007

Instruction Set Architecture

- Most instructions use a one-operand format
  - i.e. the instruction designates a single register (or literal)
  - The other (implicit) operand is the WREG

E.g:

addwf 0x100 ; reg[WREG] ← reg[WREG]+reg[0x100]

Actually, you should never write an instruction like this
See next slide.

Designating the Destination Register

- Unlike a pure "accumulator architecture" many PIC instructions can place the result into either the WREG or the specified register f.
  - A bit in the instruction designates the destination [d]
- In the assembly language, representation, an additional field is used to designate the intended destination
  - In this field,:
    - W denotes that the result should be returned to WREG
    - F denotes that the result should be returned to the resister f.

F/W Examples

- Add register 0x02B to WREG and place result in W:

  addwf 0x02B, W ; reg[WREG]←reg[WREG]+reg[0x02B]

- Add register 0x02B to WREG and place result in register 0x02B:

  addwf 0x02B, F ; reg[0x02B] ←reg[WREG]+reg[0x02B]

- Although the assembler may allow the W/F designation to be omitted, you should ALWAYS provide it to enhance code readability and avoid possible program errors.

Move Instructions

- Move a literal into WREG:

  prototype: movlw k

  example: movlw 8 ; reg[WREG]←0x08

  Whether the literal is interpreted as hex or decimal depends on the *default radix* selected (in the example it does not matter).

Move Instructions

- Move WREG into a specified register:

  prototype: movwf f

  ex: movwf 0x023 ; reg[0x023]←reg[WREG]

  ex: movwf PORTB ; reg[PORTB]←reg[WREG]

  PORTB is just a symbolic name for 0xF81 defined in the include file <P18F452.INC>

Move Instructions

- Move a specified register to another specified register:

  prototype: movff fs, fd

  example:

  movff 0x02A, 0x03D ; reg[0x03D] ← reg[0x02A]
- Unlike most PIC instructions, the machine code for movff is two words (32 bits) long.
- This instruction requires 2 cycles to execute

Move Instructions

- Move a specified register to itself or WREG:

  prototype: movf f, F/W

  examples:

  movf 0x015, W ; reg[WREG]←reg[0x015]

  movf 0x015, F ; reg[0x015] ← reg[0x015]

  The second example appears useless but is has the *side effect* of affecting the Z and N status flags.

Move Instructions

- Move a (12-bit) literal into FSR0, FSR1, or FSR2:

  prototype: lfsr i, k (i = 0, 1, or 2)

  examples:

  lfsr 0, 0xABC ; FSR0←0xABC

  lfsr 2, 0xABC ; FSR2←0xABC

Move Instructions

- Move a (4-bit) literal into the bank select register (BSR):

  prototype: movlb k (0<=k<=15)

  ex: movlb 4 ; BSR←0x4

Note: for the 18F452, only banks 0-5 and 15 make sense

Single-Operand Instructions

- Bit set (to one) in specified register:

  prototype: bsf f, b (0<=b<=7)

  Examples:

  bsf 0x043, 5 ; reg[0x043]{5}←1

  bsf PORTB, 0 ; reg[PORTB]{0}←1

The bits are numbered 7 through 0 where bit 7 is the MSb and bit 0 is the LSb.

Single-Operand Instructions

- Bit clear (to zero) in specified register:

  prototype: bcf f, b (0<=b<=7)

  Example:

  bcf PORTB, 3 ; reg[PORTB]{3}←0

The bits are numbered 7 through 0 where bit 7 is the MSb and bit 0 is the LSb.

Single-Operand Instructions

- Bit toggle in specified register:

  prototype: btg f, b

  Example:

  btg PORTB, 2

  ; if reg[PORTB]{2}=0 then

  ; reg[PORTB]{2}←1

  ; else reg[PORTB]{2}←0

Single-Operand Instructions

- Swap the high and low nibbles of the specified register:

  prototype: swapf f, F/W

  Example:

  swapf NUM, F

  ; reg[NUM]{3:0}←reg[NUM]{7:4}

  ; and reg[NUM]{7:4}←reg[NUM]{3:0}

Single-Operand Instructions

- Move 0x00 into a specified register:

  prototype: clrf f

  Example:

  clrf NUM ; reg[NUM]←0x00

  NUM is a symbolic name for a register defined previously by the assembly-language programmer.

Single-Operand Instructions

- Move 0xFF into a specified register:

  prototype: setf f

  Example:

  setf NUM ; reg[NUM]←0xFF

Single-Operand Instructions

- Rotate specified register right through carry:

  prototype: rrcf f, F/W

  Example:

  rrcf NUM

  ; reg[NUM]{6:0}←reg[NUM]{7:1}

  ; and reg[NUM]{7}←C

  ; and C←reg[NUM]{0}

Single-Operand Instructions

- Rotate specified register left though carry:

  prototype: rlcf f, F/W

  Example:

  rlcf NUM, W

  ; W{7:1}←reg[NUM]{6:0}

  ; and W{0}←C and C←reg[NUM]{7}

Single-Operand Instructions

- Rotate specified register right or left <u>not</u> through carry:

  prototypes: rrncf f, F/W

  rlncf f, F/W

  Example:

  rrncf NUM, W

  ; W{6:0}←reg[NUM]{7:1}

  ; and W{7}←reg[NUM]{0}

Single-Operand Instructions

- Increment or decrement a specified register:

  prototypes: incf f, F/W

  decf f, F/W

  Examples:

  incf NUM, W ; reg[WREG]←reg[NUM]+1

  decf NUM, F ; reg[NUM] ←reg[NUM]-1

Single-Operand Instructions

- Complement a specified register:
  prototype:
  comf f, F/W
  Examples:
  ; suppose reg[NUM] initially = 0xA1
  comf NUM, W ; reg[WREG] ←0x5E
  comf NUM, F ; reg[NUM]←0x5E

  0xA1 = B'10100001'
  0x5E = B'01011110'

Single-Operand Instructions

- Multiply 2s-complement value in specified register by –1 (negate):
  prototype:
  negf f, F/W
  Example;
  ; suppose NUM initially = 0xFE
  negf NUM, W ; reg[WREG]←0x02

  0xFE = -2 0x02 = 2

Logical Instructions

- AND (bitwise) literal with W:
  prototype:
  andlw k (k is an 8-bit literal)
  Example:
  andlw 0x55 ; W←W AND 0x55
- AND specified register with W:
  prototype:
  andwf f, F/W
  Example;
  andwf NUM, W ; reg[WREG]←reg[WREG] AND reg[NUM]

Logical Instructions

- Logical inclusive OR:
  prototypes:
  iorlw k iorwf f, F/W
- Logical exclusive OR:
  prototypes:
  xorlw k xorwf f, F/W

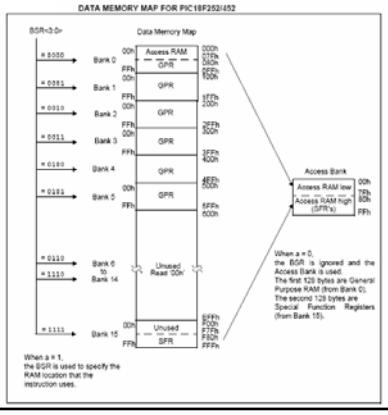
Logical Instruction Use

- Logical AND: force specified bits to 0
  Example:
  andlw B'11111110' ; reg[WREG]{0}←0
- Logical OR: force specified bits to 1
  Example:
  iorlw B'00110000' ; reg[WREG]{5:4}←B'11'
- Logical XOR: toggle specified bits
  Example
  xorlw B'00000010' ; toggle reg[WREG]{1}

A REMINDER About Access/Banked Addressing

- The examples thus far have ignored the choice of addressing mode.
- If BANKED addressing is used, the BSR must be set to designate the proper bank prior to the execution of any instruction that accesses a register or SFR.
- The programmer can select the desired mode by using the keywords ACCESS or BANKED
  e.g. negf num, W, BANKED
- The MPASM assembler defaults to ACCESS mode for addresses in the range 0x00-0x7F and SFRs. Otherwise, the default is banked.

18F452 Data Memory



A Note About Access/Banked Addressing--Continued

- Some instructions can have both a F/W designation and a BANKED/ACCESS designation. In these cases the F/W designation must come first
  e.g. decf NUM, W, ACCESS

Arithmetic Instructions

- Addition with a literal:
  prototype:
  addlw k
  example:
  addlw 5 ; reg[WREG]←5+reg[WREG]
- Addition with a specified register:
  prototype:
  addwf f, F/W
  Example:
  addwf NUM, W ; reg[WREG]←reg[NUM]+reg[WREG]

Note: All status bits are affected by these instructions

Arithmetic Instructions

- Addition with specified register and carry:
  prototype:
  addwfc f, F/W
  Example:
  addwfc NUM, F ; reg[NUM]←reg[NUM]+reg[WREG]+C

C is the carry bit conditioned by a previous instruction.

Arithmetic Instructions

- Subtraction <u>from</u> a literal:
  prototype:
  sublw k
  Example:
  sublw 5 ; reg[WREG]←5-reg[WREG]
- Subtract W from a specified register:
  prototype:
  subwf f, F/W
  Example:
  subwf NUM, W ; reg[WREG]←reg[NUM] - reg[WREG]

Arithmetic Instructions

- Subtract W and borrow from specified reg.:
  prototype:
  subwfb f, F/W
  Example:
  subwfb NUM, W ; reg[WREG]←reg[NUM]-reg[WREG]-$\overline{C}$

The borrow is just the complement of the carry bit C.

Arithmetic Instructions

- Reverse-order subtraction with borrow:

  prototype:

  subfwb f, F/W

  Example:

  subfwb NUM, W ; reg[WREG]←reg[WREG]-reg[NUM]-$\overline{C}$

Arithmetic Instructions

- Decimal adjust W (for packed BCD addition):

  prototype:

  daw

  1) If the least-significant nibble of W is 0xA or greater, 0xA is subtracted and the most-significant nibble (MSN) is incremented.

  2) If the MSN if 0xA or greater, 0xA is subtracted from the MSN and the carry flag changed to 1.

Arithmetic Instructions

- Multiplication by a literal:

  prototype:

  mullw k

  Example:

  mullw 5 ; PRODH:PRODL←reg[WREG]*5

  PRODH (0xFF4) is the most-significant byte (MSB) of the 16-bit result.

  PRODL (0xFF3) is the LSB.

Arithmetic Instructions

- Multiplication by a specified register:

  prototype: mulwf f

  Example:

  mulwf NUM ; PRODH:PRODL←Reg[NUM]*reg[WREG]

Branch Instructions

- Branch if carry flag is set (C = 1):
  prototype:
  bc label
  Example:
  bc Target ; If C=1, then jump to
  ; instruction preceded by label
  ; "Target",
  ; else continue to next
  ; sequential instruction.

Branch Instructions

- Other conditional branch instructions:

bnc label	If C=0, jump to label
bz label	If Z=1, jump to label
bnz label	If Z=0, jump to label
bn label	If N=1, jump to label
bnz label	If N=0, jump to label
bov label	If OV=1, jump to label
bnov label	If OV=0, jump to label

Branch Instructions

- Conditional branches are "near" branches – "label" must be within +/- 64 words of the instruction.
- Most instructions are one-word instructions, but the few two-word instructions must be counted as 2 in the +/- 64 restriction above.
- The conditional branch instructions are themselves one-word instruction (the restriction is a result of having only 7 bits available for the jump distance).

Branch Instructions

- Unconditional branch (to near target):
  prototype: bra label
  Example:
  bra Target ; jump to instruction labeled "Target"
- Unconditional branch (any target):
  prototype:
  goto label
  Example:
  goto Target ; jump to "Target"

"bra" is a one-word instruction and "goto" is a two-word instruction.

Conditional Skip Instructions

- Skip if specified register equals W:

  prototype:

  cpfseq f

  Example:

```
cpfseq NUM  ; If reg[WREG]==reg[NUM]
            ;  then skip next instruction word,
            ; else execute next instruction word.
```

Additional Conditional Skip Instructions

- Skip if specified register greater than W:

  Prototype:

  cpfsgt f

  Skip if specified register less than W:

  Prototype:

  cpsflt f
- Skip if specified register contains 0x00:

  Prototype:

  tstfsz f

Values are assumed to be **unsigned** for both "cpfsgt" and "cpfslt".

Still More Conditional Skip Instructions

- Decrement specified register, skip if 0x00:

  Prototype:

  decfsz f, F/W

  Example:

```
decfsz NUM, F  ; reg[NUM]←reg[NUM]-1,
               ;  if reg[NUM]=0x00,
               ;  then skip next instr.
```

Still More Conditional Skip Instructions

- Decrement specified register, skip if not 0x00:

  Prototype:

  dcfsnz f, F/W
- Increment specified register, skip if 0x00:

  Prototype:

  incfsz f, F/W
- Increment specified register, skip if not 0x00:

  Prototype:

  infsnz f, F/W

Subroutine Call and Return

- Near (relative) call (+/- 512 one-word instructions):

  Prototype:

  rcall label

  Example:

```
rcall SubOne  ; Jump to instruction labeled
              ; "SubOne" and save return
              ;  address on stack.
```

  This is a one-word instruction.

Subroutine Call and Return

- Far call (to anywhere):

  prototype:

  call label

  Example:

```
call SubOne  ; Jump to instruction labeled
             ; "SubOne" and push return
             ;  address on stack (same as rcall).
```

  This is a two-word instruction.

Subroutine Call and Return

- Return from subroutine (near or far):

  prototype:

  return

  Example:

```
return  ; Pop return address off the top of
        ; the stack and jump to this
        ; instruction address
```

Subroutine Call and Return

- Return from subroutine and place literal in WREG:

  prototype:

  retlw k

  Example:

```
retlw 5  ; Pop return address from stack
         ; and reg[WREG]←5
```

This instruction is here for historical reasons (not used much with PIC18FXXX).

Subroutine Call and Return

- FAST option on “call” and “return”:

  1) This should only be used if high-priority interrupts are not used.

  2) The “call” will copy WREG, STATUS and BSR to *shadow* registers.

  3) The “return” will restore WREG, STATUS and BSR from the shadow registers

  e.g call SubOne, FAST

Note: If the FAST call option is used, the return statement must also specify the FAST option—e.g. return FAST

Return from Interrupt

- Return from interrupt and re-enable interrupts:

  Prototype:

  retfie

  Example:

```
retfie  ; Pop address off stack and jump to that
        ; address (like return) and turn interrupt
        ; enable bit back on.
```

  Interrupt enable bits are turned off automatically when the interrupt occurs.

retfie also has a FAST option (like return)

Miscellaneous Instructions

- No operation (do nothing):
  Prototype:
  nop
- Reset (as if MCLR voltage went low):
  Prototype:
  reset
- Sleep (stop executing instructions until awakened by some event):
  Prototype:
  sleep

Miscellaneous Instructions

- Clear watchdog timer:

  Prototype:

  clrwdt
- Instructions to expand hardware stack into RAM (if need more than 31 entries):

push	Push address of next instruction onto HW stack
pop	Discard entry on top of HW stack