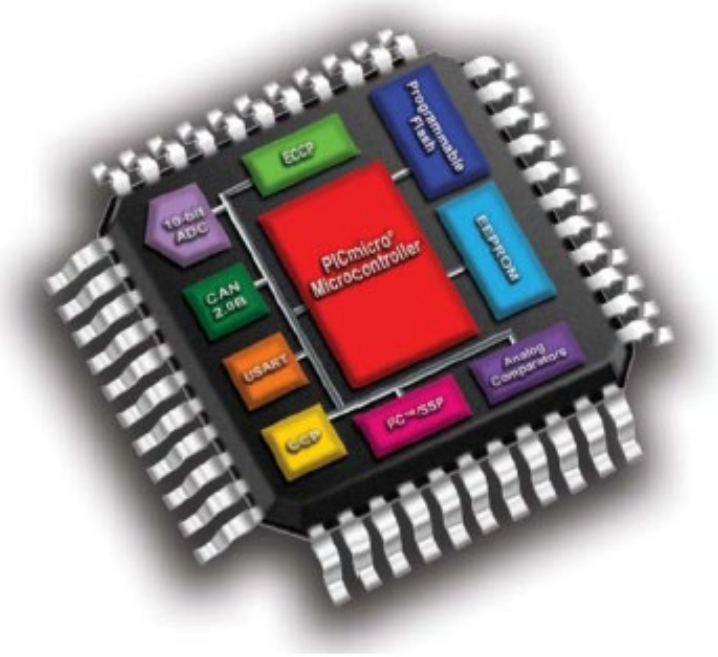
PIC Microcontroller Instruction Set



Tópicos

- Conjunto de Instruções
- Descrição das Instruções
- Diretivas do Assembler

Instruction Set

- PIC16Cxx @ PIC16Fxx: 14-bit word (opcode)
- Byte-oriented, bit-oriented & literal and control

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 compat-ibility 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

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	

Descrição das Instruções

ADDLW K

- Add the literal value K to register WREG and put the result back in the WREG register
- **(W) + k → (W)**
- K is an 8-bit value: 0-255 (decimal), 00-FF (hex)
- L: literal (actual value)
- Affect STATUS bits: C, DC, Z

ADDLW K

cont…

Instruction: ADDLW 15H

Before	After
W = 10H	W = 25H

ADDWF f, d

- Add together contents of WREG and a file register location (SFR @ GPR).
- Put the result in the register WREG if d = 0 otherwise it is stored back in register f.
- **(W) + (f) → (d)**
- $0 \leq f \leq 127$, $d \in [0, 1]$
- Affect STATUS bits: C, DC, Z

ADDWF f, d

cont…

Instruction:

```
MOVLW   17H
ADDWF   5H, 0
```

Before	After
W = 0H 5H = 0H	W = 17H 5H = 0H

MOVF f, d

- Move the content of f register upon the status of d
- (f) $\rightarrow$ (d)
- Affect bit 'Z' of STATUS register

Instruction: MOVF FSR, 0

Before	After
W = 09AH FSR = 0H	W = value in FSR register Z = 1

MOVLW k

- Load k literal into WREG register
- k $\rightarrow$ (W)
- Don't cares will be assembled as 0's
- Not affect bit of STATUS register

Instruction: MOVLW 5AH

Before	After
W = 09AH	W = 5AH

MOVWF f

- Move data from WREG register to f register
- (W) $\rightarrow$ (f)
- Not affect bit of STATUS register

Instruction: MOVWF PORTB

Before	After
PORTB = 00H W = 09AH	PORTB = 09AH W = 09AH

ANDLW k

- Logical AND k literal with the content of WREG register & the result is placed in the WREG register
- (W) AND k $\rightarrow$ (W)
- Affect Z bit of STATUS register

Instruction: ANDLW 5FH

Before	After
W = A3H	W = 03H

ANDWF f, d

- AND the WREG register with f register
- (W) + (f) → (d)
- Affect Z bit of STATUS register

Instruction: ANDWF FSR, 1

Before	After
W = 17H FSR = 0C2H	W = 17H FSR = 02H

BCF f, b

- Clear bit ‘b’ in file register
- 0 → (f<b>)
- Not affect on STATUS register

Instruction: `BCF STATUS, 5`

Before	After
STATUS = 0A7H	STATUS = 087H

BSF f, b

- Set bit ‘b’ in f register
- 1 → (f<b>)
- Not affect on STATUS register

Instruction: `BSF INTCON, 7`

Before	After
INTCON = 0BH	INTCON = 08BH

BTFSS f, b

- Execute the next instruction if bit ‘b’ in file register ‘f’ is ‘0’, otherwise discard executing next instruction
- 2-cycle instruction
- Not affect on STATUS register

Instruction:

```
H  BTFSS  STATUS, 2
I  GOTO   LOOP
J  ……
   ……
```

Before	After
PC = address H	PC = Add. J if STATUS<2> = 1, PC = Add. I if STATUS<2> = 0

BTFSC f, b

- Execute the next instruction if bit ‘b’ in file register ‘f’ is ‘1’, otherwise discard executing next instruction
- 2-cycle instruction
- Not affect on STATUS register

Instruction:

```
H  BTFSC  PORTA, 3
I  GOTO   LOOP
J  ……
   ……
```

Before	After
PC = address H	PC = Add. J if PORTA<3> = 0, PC = Add. I if PORTA<3> = 1

CALL k

- Call subroutine
- (PC) + 1 $\rightarrow$ TOS (top of stack)
- k $\rightarrow$ PC<10:0>
- (PCLATCH<4:3>) $\rightarrow$ PC<12:11>
- 2-cycle instruction
- Not affect on STATUS register

Instruction: SO CALL THEN

.....

.....

Before	After
PC = add. SO	PC = add. THEN; TOS = add. SO+1

CLRF f

- Clear the content of ‘f’ register
- 00h $\rightarrow$ (f)
- 1 $\rightarrow$ Z

Instruction: CLRF PORTA

Before	After
PORTA = 5AH	PORTA = 00H Z = 1

CLRW

- Clear the content WREG register
- 00h $\rightarrow$ (W)
- 1 $\rightarrow$ Z

Instruction: CLRW

Before	After
W = 5AH	W = 00H Z = 1

COMP f, d

- Complement the content of ‘f’ register
- $(\bar{f}) \rightarrow (d)$

Instruction: COMP ONE, 0

Before	After
ONE = 13H W = 02H	ONE = 13H W = 0ECH

DECF f, d

- Decrease ‘f’ register
- (f) – 1 → (d)

Instruction: DECF CNT, 1

Before	After
CNT = 01H Z = 0	CNT = 00H Z = 1

DECFSZ f, d

- Decrease ‘f’ register and skip the next instruction if the result is 0; otherwise execute the next instruction
- (f) – 1 → (d), skip if result = 0
- 2-cycle instruction

Instruction:

```
HERE   DECFSZ  CNT, 1
       GOTO    HERE
CONT   ……
       ……..
```

Before	After
PC = add. HERE	CNT = CNT - 1 PC = add. CONT if CNT = 0; PC = add. HERE + 1 if CNT $\neq$ 0

GOTO k

- Unconditional branch
- k → PC<10:0>
- (PCLATCH<4:3>) → PC<12:11>
- 2-cycle instruction

Instruction: HERE GOTO THERE

.....

THERE .....

Before	After
PC = add. HERE	PC = add. THERE

INCF f, d

- Increase the content of ‘f’ register
- (f) + 1 $\rightarrow$ (d)
- d is destination

Instruction: `INCF SATU, 1`

Before	After
SATU = 0FFH	SATU = 00H Z = 1

INCFSZ f, d

- Increase the content of ‘f’ register and skip the next instruction if the result is 0; otherwise execute the next instruction
- (f) + 1 $\rightarrow$ (d), skip if result = 0
- 2-cycle instruction

Instruction:

```
HERE    INCFSZ  CNT, 1
        GOTO    loop
CONT    ……
        ……..
```

Before	After
PC = add. HERE	CNT = CNT + 1 PC = add. CONT if CNT = 0; else add. HERE + 1

IORLW k

- Inclusive OR literal ‘k’ with the content of WREG register
- (W) OR k $\rightarrow$ (W)
- Affect bit ‘Z’ of STATUS register

Instruction: IORLW 35H

Before	After
W = 09AH Z = ?	W = 0BFH Z = 0

IORWF f, d

- Inclusive OR the content of WREG register with f register
- (W) OR (f) → (d)
- Affect bit 'Z' of STATUS register

Instruction: `IORWF RESULT, 0`

Before	After
RESULT = 13H W = 91H	RESULT = 13H W = 93H Z = 0

RETFIE

- Return from interrupt
- TOS $\rightarrow$ PC
- 1 $\rightarrow$ GIE (Global Interrupt Enable)
- Not affect bit of STATUS register

Instruction: RETFIE

Before	After
	PC = TOS GIE = 1

RETLW k

- Return with loading literal ‘k’ onto WREG register, k → (W)
- TOS → PC
- Not affect on STATUS register

Instruction: RETLW 088H

Before	After
W = 09AH	W = 088H

RETURN

- Return from subroutine
- POP the TOS and load into the PC
- 2-cycle instruction

Instruction: RETURN

Before	After
	PC = TOS

RLF f, d

- Rotate left f through carry
- Affect bit ‘C’ of STATUS register

Instruction: RLF REG, 1

Before	After
REG = 1111 1111 = 0FFH C = 0	REG = 1111 1110 = 0FEH C = 1

RRF f, d

- Rotate right f through carry
- Affect bit ‘C’ of STATUS register

Instruction: RRF REG, 1

Before	After
REG = 1111 0111 = 0F7H C = 0	REG = 0111 1011 = 07BH C = 1

SLEEP

- 00h $\rightarrow$ WDT
- 0 $\rightarrow$ WDT prescalar
- 1 $\rightarrow$ $\overline{TO}$
- 0 $\rightarrow$ $\overline{PD}$
- Affect $\overline{TO}$ & $\overline{PD}$ bits of STATUS register

Instruction: SLEEP

SUBLW k

- Subtract WREG register (2's complement) from literal 'k' and put the result onto WREG register
- k – (W) → (W)
- Affect C, DC & Z bits of STATUS register

Instruction: SUBLW 02H

Before	After
W = 01H C = ? Z = ?	W = 01H C = 1 Z = 0

SUBWF f, d

- Subtract WREG register (2's complement) from f register
- (f) – (W) → (d)
- Affect C, DC & Z bits of STATUS register

Instruction: SUBWF 02H, 0

Before	After
W = 01H F = 05H C = ? Z = ?	W = 04H F = 05H C = 1 Z = 0

SWAPF f, d

- Exchange the upper & lower nibbles of f register
- (f<3:0) → (d<7:4>), (f<7:4) → (d<3:0>)
- Not affect STATUS register

Instruction: SWAPF ON, 1

Before	After
ON = 0F4H W = 09AH	ON = 04FH W = 09AH

XORLW k

- Exclusive OR (XOR) the content of WREG register with k literal
- (W) XOR k $\rightarrow$ (W)
- Store the result in WREG register
- Affect bit 'Z' of STATUS register

Instruction: XORLW 0AFH

Before	After
W = 0B5H	W = 01AH

XORWF f, d

- Exclusive OR (XOR) the content of WREG register with f register
- (W) XOR (f) → (d)
- Affect bit 'Z' of STATUS register

Instruction: XORWF REG, 1

Before	After
REG = 0AFH W = 0B5H	REG = 01AH W = 0B5H

Assembler Directives

- Also known as pseudo-instructions
- Give directions to assembler
- EQU, ORG, END

EQU directive:

- Define a constant value or a fixed address

```
COUNT    EQU    0x25
....
MOVLW    COUNT
```

Assembler Directives

Using EQU for fixed data assignment:

```
;in hexadecimal
DATA1    EQU    39
DATA2    EQU    0x39
DATA3    EQU    39H
DATA4    EQU    H'39'
DATA5    EQU    h'39'

;in binary
DATA6    EQU    B'00110101'
DATA7    EQU    b'00110101'

;in decimal
DATA8    EQU    D'28'
DATA9    EQU    d'28'

;in ASCII
DATA10   EQU    A'2'
DATA11   EQU    a'2'
DATA12   EQU    '2'
```

Assembler Directives

Using EQU for SFR address assignment:

```
COUNTER  EQU     0x00
PORTB    EQU     0x06
MOVLW    COUNTER
MOVWF    PORTB
INCF     PORTB, F
INCF     PORTB, F
INCF     PORTB, F
```

Assembler Directives

Using EQU for RAM address assignment:

```
MYREG   EQU   0x12
MOVLW   0
MOVWF   MYREG
MOVLW   22H
ADDWF   MYREG, F
ADDWF   MYREG, F
ADDWF   MYREG, F
```

Assembler Directives

cont…

SET directive:

- Define a constant value or a fixed address
- Identical with EQU directive, the only difference is the value assigned by the SET directive may be reassigned later

END directive:

- Indicate the end of the source (asm) file

Assembler Directives

cont…

LIST directive:

- Unique to PIC assembler
- Indicate specific PIC chip for which the program should be assembled

```
LIST P = 16F84A
```

#include directive:

- Tells the PIC assembler to use the libraries associated with the specific chip to compile the program

Assembler Directives

cont…

_config directive:

- Tells the assembler the configuration bits for the target device
- Incorrect use may cause the chip unusable ☠

```
CONFIG      OSC=HS
CONFIG      WDT=OFF
```

radix directive:

- Indicate numbering system whether it is hexadecimal or decimal

```
RADIX       DEC
```