第 29 章 指令集

目录

本章包括下面一些主要内容：

© 2004 Microchip Technology Inc.

29.1 简介

中档系列单片机的每个指令都是14位字，由指明指令类型的操作码和进一步说明指令具体操作的一个或多个操作数组成。表 29-1 是对中档单片机指令集的概括，表中列出了 MPASM 汇编器识别的指令。整个指令系统具有高度正交性，分为三种基本类型：

- **字节**操作类指令
- **位**操作类指令
- **立即数与控制**操作类指令

表 29-2 给出了对操作码各个字段的描述。

对于**字节**操作类指令，'f' 代表文件寄存器标识符，'d' 代表目标标识符。文件寄存器标识符指定了指令将会使用哪一个文件寄存器。

目标标识符指定了操作结果的存放位置。如果 'd' 为 0，操作结果存存入 W 寄存器中。如果 'd' 为 1，操作结果存存入指令指定的文件寄存器中。

对于**位**操作类指令，'b' 代表位段标识符，选择操作影响到的位，而 'f' 代表位在哪个文件寄存器中。

对于**立即数和控制**操作类指令，'k' 代表一个 8 位或 11 位的立即数或常数。

除条件测试为真的程序跳转或指令结果改变了 PC 值的指令以外，其它所有的指令都是单周期指令。对于前面的指令，执行指令需要两个指令周期，第二个周期中执行的是一条 NOP 指令。每个指令周期包括 4 个振荡周期，因此，如果振荡器频率为 4 MHz，通常指令的执行时间为 1 µs。对于条件测试为真的程序跳转或指令结果改变了 PC 值的指令，则其执行时间为 2 µs。

© 2004 Microchip Technology Inc.

表 29-1： 中档单片机指令集

助记符、操作数		描述	周期	14 位指令字 MSb			LSb	受影响的状态	注
针对字节的文件寄存器操作指令									
ADDWF	f, d	将 W 和 f 寄存器内容相加	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	W 和 f “与”运算	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	f 清零	1	00	0001	lfff	ffff	Z	2
CLRW	-	W 清零	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	f 取反	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	f 减 1	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	f 减 1，为 0 则间跳	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	f 加 1	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	f 加 1，为 0 则间跳	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	f 的内容和 W 的内容相或	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	f 内容送入 f 或 W	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	将 W 送至 f	1	00	0000	lfff	ffff		
NOP	-	空操作	1	00	0000	0xx0	0000		
RLF	f, d	f 寄存器带进位位循环左移	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	f 寄存器带进位位循环右移	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	f 减 W	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	f 半字节交换	1	00	1110	dfff	ffff		1,2
XORWF	f, d	W 与 f 异或运算	1	00	0110	dfff	ffff	Z	1,2
针对位的文件寄存器操作指令									
BCF	f, b	f 的 bit b 清零	1	01	00bb	bfff	ffff		1,2
BSF	f, b	f 的 bit b 置 1	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	检测 f 的 bit b，为 0 则间跳	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	检测 f 的 bit b，为 1 则间跳	1 (2)	01	11bb	bfff	ffff		3
立即数和控制操作指令									
ADDLW	k	W 加立即数	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	W 与立即数相与	1	11	1001	kkkk	kkkk	Z	
CALL	k	调用子程序	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	看门狗定时器清零	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	跳转	2	10	1kkk	kkkk	kkkk		
IORLW	k	立即数或 W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	立即数送 W	1	11	00xx	kkkk	kkkk		
RETFIE	-	中断返回	2	00	0000	0000	1001		
RETLW	k	立即数送W，子程序返回	2	11	01xx	kkkk	kkkk		
RETURN	-	子程序返回	2	00	0000	0000	1000		
SLEEP	-	进入休眠模式	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	立即数减 W	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	立即数与 W 异或	1	11	1010	kkkk	kkkk	Z	

注 1：当 I/O 口寄存器用自身内容修改自身时 (例如：MOVF PORTB,1)，使用的值将是出现在引脚上的值，而非锁存器中的值。例如，如果一引脚配置为输入，其数据锁存器中的值为 ‘1’，但此时外部器件将该引脚拉为低电平，则被写回数据锁存器的数据值将是 ‘0’。

2： 如果预分频器被分配给 Timer0 模块，那么对 TMR0 寄存器执行这条指令 （适当时， d = 1）将清零预分频器。

3：如果程序计数器 (PC) 被修改或者执行一个条件检测为真的跳转，则指令需要两个指令周期。第二个指令周期执行一条空操作 NOP 指令。

© 2004 Microchip Technology Inc.

29.2 指令格式

图 29-1 给出了指令的三种通用格式。从指令的通用格式可以看出，指令字的操作码部分可以有 3 位到 6 位的信息变化。这就是中档系列单片机指令可以有 35 条之多的原因。

注 1：	任何未使用的操作码都是保留的，使用任何保留的操作码将会导致异常操作。
注 2：	为保持对将来中档系列产品的向上兼容性，不要使用 OPTION 和 TRIS 指令。

所有的指令例子均使用下面的格式来表示十六进制数：

0xhh

其中 h 代表一个 16 进制位。

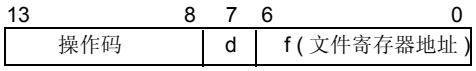
表示二进制数：

00000100b

b 为二进制数的标志。

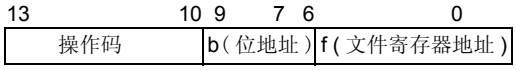
图 29-1： 指令的通用格式

针对字节的文件寄存器操作指令



d = 0，目标寄存器为 W
d = 1，目标寄存器为 f
f = 7 位文件寄存器地址

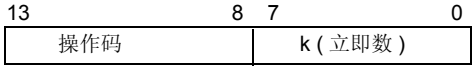
针对位的文件寄存器操作指令



b = 3 位位地址
f = 7 位文件寄存器地址

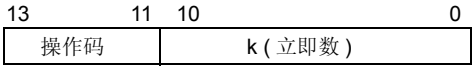
立即数和控制操作指令

一般指令



k = 8 位立即数

仅 CALL 和 GOTO 指令



k = 11 位立即数

© 2004 Microchip Technology Inc.

表 29-2： 指令描述约定

字段	描述
f	文件寄存器地址 (0x00 到 0x7F)
W	工作寄存器 (累加器)
b	某 8 位文件寄存器内的位地址 (0 到 7)
k	立即数、常数或标号 （可能是 8 位或 11 位值）
x	与取值无关的位 (0 或 1) 汇编器将产生 x = 0 的代码，为了与所有的 Microchip 软件工具兼容，建议使用这种格式。
d	目标寄存器选择： d = 0：结果保存至 W d = 1：结果保存至文件寄存器 f
dest	目标寄存器， W 寄存器或指定的文件寄存器地址
label	标号名
TOS	栈顶
PC	程序计数器
PCLATH	程序计数器高位锁存器
GIE	全局中断允许位
WDT	看门狗定时器
$\overline{TO}$	超时标志位
$\overline{PD}$	掉电标志位
[ ]	可选的
( )	内容
→	赋值给
< >	寄存器位域
∈	表示属于某个集合
italics	用户定义项（字体为 courier）

29.3 作为源 / 目标寄存器的特殊功能寄存器

在本章中介绍的指令系统高度正交，可以实现对所有寄存器，包括特殊功能寄存器的读写。一些需要注意的特殊情况介绍如下：

29.3.1 STATUS 状态寄存器作为目标寄存器

如果一条指令写 STATUS 状态寄存器，Z、C、DC 和 OV 位会被指令运行结果置 1 或清零而覆盖掉写入的原始数据位。例如，执行 CLRF STATUS 会将 STATUS 寄存器清零，并将 Z 位置 1，寄存器的内容变为 0000 0100b 。

29.3.2 PCL 寄存器作为 源寄存器或目标寄存器

对 PCL 进行读、写或读－修改－写可能会导致以下结果：

读 PC：	PCL →dest； PCLATH 不变；
写 PCL：	PCLATH → PCH； 8 位目标值 → PCL
读－修改－写：	PCL→ ALU 操作数 PCLATH → PCH； 8 位结果 → PCL

其中 PCH 为程序计数器的高字节 (不可寻址寄存器)，PCLATH 为程序计数器高字节锁存器，dest 为目标寄存器 （W 寄存器或文件寄存器 f）。

29.3.3 位操作

所有位操作指令将首先读整个寄存器的内容，然后在选择的位上进行操作，最后将结果写回 (读－修改－写 (R-M-W)) 指定寄存器。当对一些特殊功能寄存器操作时 （如端口寄存器），应特别注意这一点。

注：	在 Q1 周期中，器件操作的状态位 (包括中断标志位) 被置 1 或清零，所以对包含这些位的寄存器执行 R-M-W 指令无效。

© 2004 Microchip Technology Inc.

29.4 Q 周期操作

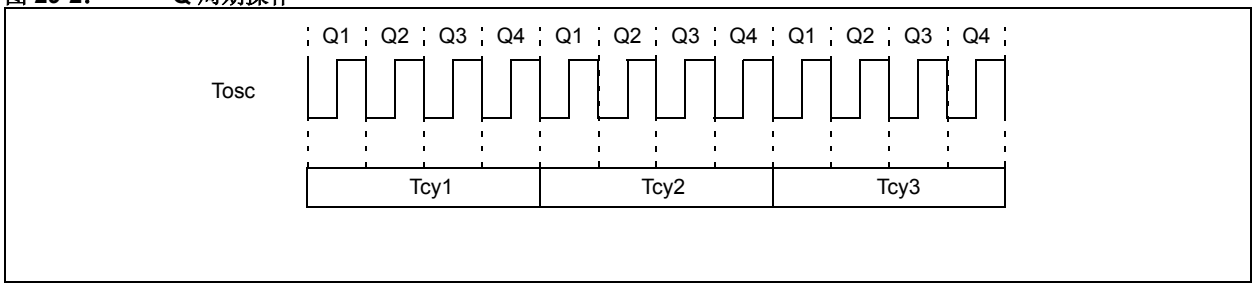
每一个指令周期 (Tcy) 由 4 个 Q 周期 (Q1-Q4) 组成。 Q 周期和器件振荡器周期 (Tosc) 相同。在每个指令周期， Q 周期提供译码、处理数据和读写操作的时序。下图给出了 Q 周期与指令周期的关系。

4 个 Q 周期组成一个指令周期 (Tcy)，它们是：

Q1：指令译码周期或无操作
Q2：指令读周期或无操作
Q3：处理数据
Q4：指令写周期或无操作

每条指令都有详细的 Q 周期操作。

图 29-2： Q 周期操作



29.5 指令描述

ADDLW 立即数与 W 寄存器内容相加

语法： [*标号*] ADDLW k

操作数： 0 ≤ k ≤ 255

操作： (W) + k → W

影响的状态位： C、 DC 和 Z

机器码：

11	111x	kkkk	kkkk

描述 8 位立即数 'k' 与 W 寄存器的内容相加，结果存入 W 寄存器。

指令字数： 1

指令周期： 1

Q 期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 'k'	处理 数据	写 W 寄存器

例 1

```
ADDLW   0x15
```

执行指令前：
W=0x10
执行后
W = 0x25

例 2

```
ADDLW   MYREG
```

执行指令前
W = 0x10
MYREG† 地址 = 0x37
† MYREG 是表示数据存储单元的符号
执行后
W = 0x47

例 3

```
ADDLW   HIGH (LU_TABLE)
```

执行指令前：
W = 0x10
LU_TABLE† 地址 = 0x9375
† LU_TABLE 为程序存储器中的地址标号
执行后
W = 0xA3

例 4

```
ADDLW   MYREG
```

执行指令前
W = 0x10
PCL† 地址 = 0x02
† PCL 是表示程序计数器低字节单元的符号
执行后
W = 0x12

© 2004 Microchip Technology Inc.

ADDWF W 与 f 相加

语法： [*标号*] ADDWF f,d

操作数： $0 \le f \le 127$
$d \in [0,1]$

操作： (W) + (f) → 目标寄存器

影响的状态位： C、 DC 和 Z

机器码：

00	0111	dfff	ffff

描述： W 寄存器与 'f' 寄存器的内容相加。如果 'd' 为 0，结果存储在 W 寄存器中。如果 'd' 为 1，结果存储在 'f' 寄存器中。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标存储器

例 1
```
ADDWF   FSR, 0
```
执行指令前：
W = 0x17
FSR = 0xC2
执行后
W = 0xD9
FSR = 0xC2

例 2
```
ADDWF   INDF, 1
```
执行指令前
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x20
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x37

例 3
```
ADDWF   PCL
```
情况 1:
执行指令前
W = 0x10
PCL = 0x37
C = x
执行后
PCL = 0x47
C = 0

情况 2:
执行指令前
W = 0x10
PCL = 0xF7
PCH = 0x08
C = x
执行后
PCL = 0x07
PCH = 0x08
C = 1

ANDLW 立即数与 W 相与

语法： [*标号*] ANDLW k

操作数： 0 ≤ k ≤ 255

操作： (W).AND. (k) → W

影响的状态位： Z

机器码：

11	1001	kkkk	kkkk

描述： W 寄存器的内容与 8 位立即数 'k' 相与，结果存入 W 寄存器中。

指令字数： 1

指令周期： 1

Q 周期操作

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	写 W 寄存器

例 1

```
  ANDLW   0x5F

执行指令前：                    ; 0101 1111  (0x5F)
        W   =  0xA3             ; 1010 0011  (0xA3)
执行后                          ;----------  ------
        W   =  0x03             ; 0000 0011  (0x03)
```

例 2

```
  ANDLW   MYREG
```

执行指令前

W = 0xA3

MYREG† 地址 = 0x37

† MYREG 是数据存储单元的标号

执行后

W = 0x23

例 3

```
  ANDLW   HIGH (LU_TABLE)
```

执行指令前

W = 0xA3

LU_TABLE† 地址 = 0x9375

† LU_TABLE 为程序存储单元的标号

执行后

W = 0x83

© 2004 Microchip Technology Inc.

ANDWF　　W 与 f 相与

语法：　[*标号*] ANDWF　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(W).AND. (f) → 目标寄存器

影响的状态位：　Z

机器码：

00	0101	dfff	ffff

描述：　W 寄存器与 'f' 寄存器相与。如果 'd' 为 0，结果存入 W 寄存器。如果 'd' 为 1，结果存入 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写目标 寄存器

例 1

```
ANDWF   FSR, 1

执行指令前                          ; 0001 0111  (0x17)
        W   =  0x17                 ; 1100 0010  (0xC2)
        FSR =  0xC2                 ;----------  ------
执行后                              ; 0000 0010  (0x02)
        W   =  0x17
        FSR =  0x02
```

例 2

```
ANDWF   FSR, 0

执行指令前                          ; 0001 0111  (0x17)
        W   =  0x17                 ; 1100 0010  (0xC2)
        FSR =  0xC2                 ;----------  ------
执行后                              ; 0000 0010  (0x02)
        W   =  0x02
        FSR =  0xC2
```

例 3

```
ANDWF   INDF, 1

执行指令前
        W   =  0x17
        FSR =  0xC2
        地址 (FSR) 的内容 = 0x55
执行后
        W   =  0x17
        FSR =  0xC2
        地址 (FSR) 的内容 = 0x15
```

BCF 位清零

语法： [*标号*] BCF f,b

操作数： 0 ≤ f ≤ 127
0 ≤ b ≤ 7

操作： 0 → f<b>

影响的状态位： 无

机器码：

01	00bb	bfff	ffff

描述： 将寄存器 'f' 的位 'b' 清零。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 寄存器 'f'

例 1

```
BCF     FLAG_REG, 7
```

执行指令前

FLAG_REG = 0xC7 ; **1**100 0111

执行后

FLAG_REG = 0x47 ; **0**100 0111

例 2

```
BCF     INDF, 3
```

执行指令前

W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x2F

执行后

W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x27

© 2004 Microchip Technology Inc.

BSF 位置 1

语法： [*标号*] BSF f,b

操作数： 0 ≤ f ≤ 127
0 ≤ b ≤ 7

操作： 1 → f<b>

影响的状态位： 无

机器码：

01	01bb	bfff	ffff

描述： 将寄存器 'f' 的位 "b' 置 1。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 寄存器 'f'

例 1

```
BSF     FLAG_REG, 7
```

执行指令前

```
FLAG_REG =0x0A          ; 0000 1010
```

执行后

```
FLAG_REG =0x8A          ; 1000 1010
```

例 2

```
BSF     INDF, 3
```

执行指令前

```
W   =  0x17
FSR =  0xC2
地址 (FSR) 的内容 = 0x20
```

执行后

```
W   =  0x17
FSR =  0xC2
地址 (FSR) 的内容 = 0x28
```

BTFSC 位测试，为 0 间跳

语法： [*标号*] BTFSC f,b

操作数： 0 ≤ f ≤ 127
0 ≤ b ≤ 7

操作： skip if (f<b>) = 0

影响的状态位： 无

机器码：

01	10bb	bfff	ffff

描述： 如果寄存器 'f' 的位 'b' 为 '0'，则跳过下一条指令。
如果位 'b' i 为 '0' ，那么下一条指令 (在当前指令执行期间取指) 被丢弃而执行一条 NOP 指令，使该指令变成 2 周期指令。

指令字数： 1

指令周期： 1(2)

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	无操作

I 如果间跳 (第二个 周期):

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例 1

```
HERE    BTFSC   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

情况 1: 执行指令前
PC = 地址 HERE
FLAG= xxx0 xxxx
执行后
由于 FLAG<4>= 0,
PC = 地址 TRUE

情况 2: 执行指令前
PC = 地址 HERE
FLAG= xxx1 xxxx
执行后
由于 FLAG<4>=1,
PC = 地址 FALSE

© 2004 Microchip Technology Inc.

BTFSS 位测试 , 为 1 间跳

语法： [*标号*] BTFSS f,b

操作数： 0 ≤ f ≤ 127
0 ≤ b < 7

操作： 跳出，如果 (f<b>) = 1

影响的状态位： 无

机器码：

01	11bb	bfff	ffff

描述： 如果寄存器 'f' 的位 b' 为 '1'，则跳过下一条指令。
如果位 b' 为 '1', 那么下一条指令 (在当前指令执行期间取指) 被丢弃而执行一条 NOP 指令，使该指令变成 2 周期指令。

指令字数： 1

指令周期： 1(2)

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	无操作

I 如果间跳 (第二个周期):

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例 1

```
HERE    BTFSS   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

情况 1: 执行指令前
PC = 地址 HERE
FLAG= xxx0 xxxx
执行后
由于 FLAG<4>= 0,
PC = 地址 FALSE

情况 2: 执行指令前
PC = 地址 HERE
FLAG= xxx1 xxxx
执行后
由于 FLAG<4>=1,
PC = 地址 TRUE

CALL 调用子程序

语法： [*标号*] CALL k

操作数： $0 \le k \le 2047$

操作： (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

影响的状态位： 无

机器码：

10	0kkk	kkkk	kkkk

描述： 调用子程序。首先， 13 位返回地址 (PC+1) 被压入堆栈。11 位立即数地址被装入 PC 位 <10:0>。PC 高位从 PCLATH<4:3> 装入。CALL 为一条两周期指令。

指令字数： 1

指令周期： 2

Q 周期操作：

第一个周期：

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	无操作

第二个周期：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例 1

```
HERE    CALL   THERE
```

执行指令前
PC = 地址HERE
执行后
TOS = 地址 HERE+1
PC = 地址 THERE

© 2004 Microchip Technology Inc.

CLRF 寄存器 f 清零

语法： [*标号*] CLRF f

操作数： 0 ≤ f ≤ 127

操作： 00h → f
1 → Z

影响的状态位： Z

机器码：

00	0001	1fff	ffff

描述： 寄存器 'f' 被清零， Z 位置 1。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 寄存器 'f'

例 1

```
CLRF    FLAG_REG
```

执行指令前：
FLAG_REG=0x5A
执行后
FLAG_REG=0x00
Z = 1

例 2

```
CLRF    INDF
```

执行指令前
FSR = 0xC2
地址 (FSR)的内容 =0xAA
执行后
FSR = 0xC2
地址 (FSR) 的内容=0x00
Z = 1

CLRW　　W 寄存器清零

语法：　[*标号*]　CLRW

操作数：　无

操作：　00h → W
1 → Z

影响的状态位：　Z

机器码：

00	0001	0xxx	xxxx

描述：　W 寄存器被清零。 Z 位置 1。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 寄存器 'W'

例 1

```
CLRW
```

执行指令前
W = 0x5A
执行后
W = 0x00
Z = 1

© 2004 Microchip Technology Inc.

CLRWDT 看门狗定时器 WDT 清零

语法： [*标号*] CLRWDT

操作数： 无

操作： 00h → WDT
0 → WDT 预分频器计数，
1 → $\overline{TO}$
1 → $\overline{PD}$

影响的状态位： $\overline{TO}$, $\overline{PD}$

机器码：

00	0000	0110	0100

描述： CLRWDT 指令清零看门狗定时器，同时将 WDT 的预分频器计数值清零。状态位 $\overline{TO}$ 和 $\overline{PD}$ 被置 1。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无操作	处理数据	清零 WDT 计数器

例 1

```
CLRWDT
```

执行指令前
WDT 计数器= x
WDT 预分频器分频比 =1:128
执行后
WDT 计数器=0x00
WDT 预分频器计数=0
$\overline{TO}$ = 1
$\overline{PD}$ = 1
WDT 预分频器分频比 =1:128

注： CLRWDT 指令并不影响 WDT 预分频器的分配。

COMF　　f 寄存器内容取反

语法：　[*标号*]　COMF　f,d

操作数：　$0 \le f \le 127$
$d \in [0,1]$

操作：　$(\bar{f}) \rightarrow$ 目标寄存器

影响的状态位：　Z

机器码：

00	1001	dfff	ffff

描述：　寄存器 'f' 的内容取反。如果 'd' 为 0，结果存入 W；如果 'd' 为 1，结果存入寄存器 'f'。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标寄存器

例 1

```
COMF    REG1, 0
```

执行指令前
REG1=　0x13
执行后
REG1=　0x13
W　=　0xEC

例 2

```
COMF    INDF, 1
```

执行指令前
FSR =　0xC2
地址 (FSR)的内容 =0xAA
执行后
FSR =　0xC2
地址 (FSR)的内容=0x55

例 3

```
COMF    REG1, 1
```

执行指令前
REG1=　0xFF
执行后
REG1=　0x00
Z　=　1

© 2004 Microchip Technology Inc.

DECF　　f 寄存器内容减 1

语法：　[*标号*]　DECF f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(f) - 1 → 目标寄存器

影响的状态位：　Z

机器码：

00	0011	dfff	ffff

描述：　寄存器 'f' 内容减 1。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1，结果存回 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标寄存器

例 1

```
DECF    CNT, 1
```

执行指令前
CNT = 0x01
Z = 0
执行后
CNT = 0x00
Z = 1

例 2

```
DECF    INDF, 1
```

执行指令前
FSR = 0xC2
地址 (FSR) 的内容 = 0x01
Z = 0
执行后
FSR = 0xC2
地址 (FSR) 的内容 = 0x00
Z = 1

例 3

```
DECF    CNT, 0
```

执行指令前
CNT = 0x10
W = x
Z = 0
执行后
CNT = 0x10
W = 0x0F
Z = 0

DECFSZ　　f 寄存器内容减 1，为 0 间跳

语法：　[*标号*]　DECFSZ　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(f) - 1 → 目标寄存器；结果为 0 间跳

影响的状态位：　无

机器码：

00	1011	dfff	ffff

描述：　寄存器 'f' 的内容减 1。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1，结果存入寄存器 'f'。
如果结果为 0，那么下一条指令 (在当前指令执行期间取指) 被丢弃而执行一条 NOP 指令，使该指令变成 2 周期指令。

指令字数：　1

指令周期：　1(2)

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标寄存器

如果间跳 (第二个周期):

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例

```
HERE     DECFSZ   CNT, 1
         GOTO     LOOP
CONTINUE •
         •
         •
```

情况 1:　执行指令前
PC = 地址 HERE
CNT = 0x01
执行后
CNT = 0x00
PC = 地址 CONTINUE

情况 2:　执行指令前
PC = 地址 HERE
CNT = 0x02
执行后
CNT = 0x01
PC = 地址 HERE + 1

© 2004 Microchip Technology Inc.

GOTO 无条件转移

语法： [*标号*] GOTO k

操作数： $0 \le k \le 2047$

操作： $k \rightarrow PC<10:0>$
$PCLATH<4:3> \rightarrow PC<12:11>$

影响的状态位： 无

机器码：

10	1kkk	kkkk	kkkk

描述： GOTO 是无条件转移指令。11 位立即数被装入 PC 位 <10:0>，PC 的高位从 PCLATH<4:3> 装入。GOTO 为 2 周期指令。

指令字数： 1

指令周期： 2

Q 周期操作：

第一个周期：

Q1	Q2	Q3	Q4
译码	读数 'k'<7:0>	处理数据	无操作

第二个周期：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例

```
GOTO THERE
```

执行后

PC = 地址 THERE

INCF　　f 寄存器内容加 1

语法：　[*标号*]　INCF　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(f) + 1 → 目标寄存器

影响的状态位：　Z

机器码：

00	1010	dfff	ffff

描述：　寄存器 'f' 的内容加 1。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1 结果存回 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标寄存器

例 1

```
INCF    CNT, 1
```

执行指令前
CNT = 0xFF
Z = 0
执行后
CNT = 0x00
Z = 1

例 2

```
INCF    INDF, 1
```

执行指令前
FSR = 0xC2
地址 (FSR) 的内容 = 0xFF
Z = 0
执行后
FSR = 0xC2
地址 (FSR) 的内容 = 0x00
Z = 1

例 3

```
INCF    CNT, 0
```

执行指令前
CNT = 0x10
W = x
Z = 0
执行后
CNT = 0x10
W = 0x11
Z = 0

© 2004 Microchip Technology Inc.

INCFSZ　　f 寄存器内容加 1, 为 0 间跳

语法：　[*l 标号*]　INCFSZ　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(f) + 1 → 目标寄存器，结果为 0 间跳

影响的状态位：　无

机器码：

00	1111	dfff	ffff

描述：　寄存器 'f' 的内容加 1。如果 'd' 为 0 结果存入 W 寄存器；如果 'd' 为 1 结果存回 'f' 寄存器。
如果结果为 0，那么下一条指令 (在当前指令执行期间取指) 被丢弃而执行一条 NOP 指令，使该指令变成 2 周期指令。 .

指令字数：　1

指令周期：　1(2)

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 'f'	处理 数据	写 目标寄存器

如果间跳 (第二个周期):

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

例

```
HERE     INCFSZ   CNT, 1
         GOTO     LOOP
CONTINUE •
         •
         •
```

情况 1:　执行指令前
PC = 地址 HERE
CNT = 0xFF
执行后
CNT = 0x00
PC = 地址 CONTINUE

情况 2:　执行指令前
PC = 地址 HERE
CNT = 0x00
执行后
CNT = 0x01
PC = 地址 HERE + 1

IORLW　　8 位立即数和 W 寄存器内容相或

语法：　[*标号*]　IORLW　k

操作数：　0 ≤ k ≤ 255

操作：　(W).OR. k → W

影响的状态位：　Z

机器码：

11	1000	kkkk	kkkk

描述：　W 寄存器的内容与 8 位立即数 'k' 相或，结果存入 W 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 'k'	数据处理	写 W 寄存器

例 1

```
IORLW   0x35
```

执行指令前
W = 0x9A
执行后
W = 0xBF
Z = 0

例 2

```
IORLW   MYREG
```

执行指令前
W = 0x9A
MYREG† 地址 = 0x37
† MYREG 是数据存储单元的符号
执行后
W = 0x9F
Z = 0

例 3

```
IORLW   HIGH (LU_TABLE)
```

执行指令前
W = 0x9A
LU_TABLE† 地址 = 0x9375
† LU_TABLE 是程序存储单元的标号
执行后
W = 0x9B
Z = 0

例 4

```
IORLW   0x00
```

执行指令前
W = 0x00
执行后
W = 0x00
Z = 1

© 2004 Microchip Technology Inc.

IORWF　　W 寄存器内容与 f 寄存器内容相或

语法：　[*l 标号*]　IORWF　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(W).OR. (f) → 目标寄存器

影响的状态位：　$\overline{Z}$

机器码：

00	0100	dfff	ffff

描述：　IW 寄存器内容与 f 寄存器内容相或。如果 'd' 为 0 ，结果存入 W 寄存器；如果 'd' 为 1，结果存入 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
IORWF   RESULT, 0
```

执行指令前
RESULT=0x13
W = 0x91
执行后
RESULT=0x13
W = 0x93
Z = 0

例 2

```
IORWF   INDF, 1
```

执行指令前
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x30
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x37
Z = 0

例 3

```
IORWF   RESULT, 1
```

情况 1:　执行指令前
RESULT=0x13
W = 0x91
执行指令前
RESULT=0x93
W = 0x91
Z = 0

情况 2:　执行指令前
RESULT=0x00
W = 0x00
执行后
RESULT=0x00
W = 0x00
Z = 1

MOVLW 立即数送入 W 寄存器

语法： [*标号*] MOVLW k

操作数： $0 \le k \le 255$

操作： $k \rightarrow W$

影响的状态位： 无

机器码：

11	00xx	kkkk	kkkk

描述： 8 位立即数 'k' 送入 W 寄存器。无关的位置为 0'。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	写 W 寄存器

例 1

```
MOVLW   0x5A
```

执行后

W = 0x5A

例 2

```
MOVLW   MYREG
```

执行指令前

W = 0x10

MYREG† 地址 = 0x37

† MYREG 是数据存储单元的符号

执行后

W = 0x37

例 3

```
MOVLW   HIGH (LU_TABLE)
```

执行指令前

W = 0x10

of LU_TABLE 地址 = 0x9375

† LU_TABLE 程序存储单元的标号

执行后

W = 0x93

© 2004 Microchip Technology Inc.

MOVF 寄存器 f 传送

语法： [*标号*] MOVF f,d

操作数： 0 ≤ f ≤ 127
d ∈ [0,1]

操作： (f) → 目标寄存器

影响的状态位 Z

机器码：

00	1000	dfff	ffff

描述： 将寄存器 'f' 的内容送入目标寄存器。如果 'd' = 0，目标寄存器为 W 寄存器；如果 'd' = 1，目标寄存器为 'f' 寄存器本身。由于状态标志 位 Z 受到指令结果的影响， 'd' = 1 可用于检测文件寄存器内容是否为 0。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
MOVF    FSR, 0
```

执行指令前
W = 0x00
FSR = 0xC2
执行后
W = 0xC2
Z = 0

例 2

```
MOVF    INDF, 0
```

执行指令前
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x00
执行后
W = 0x00
FSR = 0xC2
地址 (FSR) 的内容 = 0x00
Z = 1

例 3

```
MOVF    FSR, 1
```

情况 1:
执行指令前
FSR = 0x43
执行后
FSR = 0x43
Z = 0

情况 2:
执行指令前
FSR = 0x00
执行后
FSR = 0x00
Z = 1

© 2004 Microchip Technology Inc.

MOVWF　　W 内容传送至 f

语法：　[*标号*]　MOVWF　f

操作数：　0 ≤ f ≤ 127

操作：　(W) → f

影响的状态位：　无

机器码：

00	0000	1fff	ffff

描述：　将数据从 W 寄存器送入 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写寄存器 'f'

例 1

```
MOVWF   OPTION_REG
```

执行指令前
OPTION_REG=0xFF
W = 0x4F
执行后
OPTION_REG=0x4F
W = 0x4F

例 2

```
MOVWF   INDF
```

执行指令前
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x00
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x17

© 2004 Microchip Technology Inc.

NOP 空操作

语法： [*标号*] NOP

操作数： 无

操作： 空操作

影响的状态位： 无

机器码：

00	0000	0xx0	0000

描述： 无操作

指令字数： 1

指令周期： 1

Q 周期操作

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作

例：

```
HERE    NOP
```

执行指令前
PC = HERE 地址
执行后
PC = HERE 地址 + 1

OPTION　　Option 寄存器赋值

语法：	[*标号*]　OPTION
操作数：	无
操作：：	(W) → OPTION
影响的状态位：：	无
机器码：	00 | 0000 | 0110 | 0010
描述：：	此指令将 W 寄存器的内容送入 OPTION 寄存器。支持这条指令是为了与 PIC16C5X 产品代码兼容。 OPTION 为一个可读 / 写寄存器，用户可对其直接寻址。.
指令字数：	1
指令周期：：	1

为保持与未来 PIC16CXX 产品的向上兼容，不要使用该指令。

© 2004 Microchip Technology Inc.

RETFIE 中断返回

语法： [*标号*] RETFIE

操作数： 无

操作： TOS → PC，
1 → GIE

影响的状态位 无

机器码：

00	0000	0000	1001

描述： 从中断返回。栈顶（TOS）的 13 位地址装入 PC。全局中断允许位 GIE (INTCON<7>)，自动置 1，允许中断。这是一条 2 周期指令。

指令字数： 1

指令周期： 2

Q 周期操作：

第一个周期：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	空操作

第二个周期：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

例

```
RETFIE
```

执行后

```
PC  =  TOS
GIE =  1
```

RETLW 带立即数子程序返回

语法： [*标号*] RETLW k

操作数： 0 ≤ k ≤ 255

操作： k → W ；
TOS → PC

影响的状态位： 无

机器码：

11	01xx	kkkk	kkkk

描述： 8 位立即数 'k' 装载至 W 寄存器，栈顶的 13 位地址 (返回地址) 送入程序计数器。这是一条 2 周期指令。

指令字数： 1

指令周期： 2

Q 周期操作：

第一个周期：

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	写 W 寄存器

第二个周期：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

Example

```
HERE    CALL TABLE   ; W contains table
                     ; offset value
        •            ; W now has table value
        •
        •
TABLE   ADDWF PC     ;W = offset
        RETLW k1     ;Begin table
        RETLW k2     ;
        •
        •
        •
        RETLW kn     ; End of table
```

执行指令前

W = 0x07

执行后

W = k8 的值

PC = TOS = Here 地址 + 1

© 2004 Microchip Technology Inc.

RETURN 子程序返回

语法： [*标号*] RETURN

操作数： 无

操作： TOS → PC

影响的状态位： 无

机器码：

00	0000	0000	1000

描述： 从子程序返回。堆栈中的数据被弹出，栈顶 (TOS) （即返回地址）被装入程序计数器。这是一条 2 周期指令。

指令字数： 1

指令周期： 2

Q 周期操作：

第一个指令周期：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	空操作

第二个指令周期：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

例

```
HERE    RETURN
```

执行后

```
PC = TOS
```

RLF 带进位位循环左移

语法： [*标号*] RLF f,d

操作数： $0 \le f \le 127$
$d \in [0,1]$

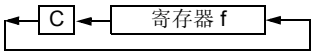
操作： 参见下面的描述

影响的状态位： C

机器码：

00	1101	dfff	ffff

描述： 寄存器 'f' 的内容带进位位循环左移。如果 'd' 为 0 ，结果存入 W 寄存器；如果 'd' 为 1，结果存入 'f' 寄存器 。



指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
RLF     REG1,0
```

执行指令前
REG1= 1110 0110
C = 0
执行后
REG1=1110 0110
W =1100 1100
C =1

例 2

```
RLF     INDF, 1
```

情况 1: 执行指令前
W = xxxx xxxx
FSR = 0xC2
地址 (FSR) 的内容 = 0011 1010
C = 1
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0111 0101
C = 0

情况 2: 执行指令前
W = xxxx xxxx
FSR = 0xC2
地址 (FSR) 的内容 = 1011 1001
C = 0
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0111 0010
C = 1

© 2004 Microchip Technology Inc.

RRF 带进位位循环右移

语法： [*标号*] RRF f,d

操作数： 0 ≤ f ≤ 127
d ∈ [0,1]

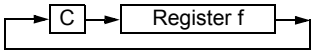
操作： 参见下面的描述

影响的状态位： C

机器码：

00	1100	dfff	ffff

描述： 寄存器 'f' 的内容带进位循环右移。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1，结果存入 'f' 寄存器。



指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
RRF          REG1,0
```

执行指令前
REG1= 1110 0110
W = xxxx xxxx
C = 0
执行后
REG1= 1110 0110
W = 0111 0011
C = 0

例 2

```
RRF     INDF, 1
```

情况 1: 执行指令前
W = xxxx xxxx
FSR = 0xC2
地址 (FSR) 的内容 = 0011 1010
C = 1
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 1001 1101
C = 0

情况 2: 执行指令前
W = xxxx xxxx
FSR = 0xC2
地址 (FSR) 的内容 = 0011 1001
C = 0
执行后
W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0001 1100
C = 1

SLEEP

语法： [*标号*] SLEEP

操作数： 无

操作： 00h → WDT，
0 → WDT 预分频器计数，
1 → $\overline{TO}$,
0 → $\overline{PD}$

影响的状态位 $\overline{TO}$,、 $\overline{PD}$

机器码：

00	0000	0110	0011

描述： 掉电状态位 $\overline{PD}$ 清零。超时状态位 $\overline{TO}$ 位置 1。看门狗定时器及其预分频器计数清零。
振荡器停止工作，单片机进入休眠模式。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	进入休眠模式

例： SLEEP

注： SLEEP 指令并不影响 WDT 预分频器的分配。

© 2004 Microchip Technology Inc.

SUBLW 立即数与 W 寄存器的内容相减

语法： [*标号*] SUBLW k

操作数： $0 \le k \le 255$

操作： $k - (W) \rightarrow W$

影响的状态位： C、 DC 和 Z

机器码：

11	110x	kkkk	kkkk

描述： 8 位立即数减去 W 寄存器内容 (通过二进制补码方法进行减法运算)，结果存入 W 寄存器。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	写 W 寄存器

例 1:
```
SUBLW   0x02
```
情况 1: 执行指令前
```
W   =  0x01
C   =  x
Z   =  x
```
执行后
```
W   =  0x01
C   =  1        ; 结果为正
Z   =  0
```
情况 2: 执行指令前
```
W   =  0x02
C   =  x
Z   =  x
```
执行后
```
W   =  0x00
C   =  1        ; 结果为 0
Z   =  1
```
情况 3: 执行指令前
```
W   =  0x03
C   =  x
Z   =  x
```
执行后
```
W   =  0xFF
C   =  0        ; 结果为负
Z   =  0
```
例 2
```
SUBLW   MYREG
```
执行指令前
```
W   =  0x10
```
MYREG† 地址 = 0x37

† MYREG 是数据存储单元的符号

执行后
```
W   =  0x27
C   =  1    ; 结果为正
```

SUBWF　　f 寄存器内容减 W 寄存器内容

语法：　[*标号*]　SUBWF　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(f) - (W) → 目标寄存器

影响的状态位　C、 DC 和 Z

机器码：

00	0010	dfff	ffff

描述：　f 寄存器内容减去 W 寄存器内容。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1，结果存入 'f' 寄存器。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写 目标寄存器

例 1:
```
SUBWF    REG1,1
```
情况 1:　执行指令前
```
REG1= 3
W   = 2
C   = x
Z   = x
```
执行后
```
REG1= 1
W   = 2
C   = 1          ; 结果为正
Z   = 0
```
情况 2:　执行指令前
```
REG1= 2
W   = 2
C   = x
Z   = x
```
执行后
```
REG1= 0
W   = 2
C   = 1          ; 结果为 0
Z   = 1
```
情况 3:　执行指令前
```
REG1= 1
W   = 2
C   = x
Z   = x
```
执行后
```
REG1= 0xFF
W   = 2
C   = 0          ; 结果为负
Z   = 0
```


© 2004 Microchip Technology Inc.

SWAPF 寄存器半字节交换

语法： [*标号*] SWAPF f,d

操作数： 0 ≤ f ≤ 127
d ∈ [0,1]

操作： (f<3:0>) → 目标寄存器 <7:4>,
(f<7:4>) → 目标寄存器 <3:0>

影响的状态位： 无

机器码：

00	1110	dfff	ffff

描述： 寄存器 'f' 的高半字节和低半字节交换。如果 'd' 为 0 结果，存入 W 寄存器；如果 'd' 为 1，结果存入 'f' 寄存器。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
SWAPF   REG, 0
```

执行指令前

REG1= 0xA5

执行后

REG1= 0xA5
W = 0x5A

例 2

```
SWAPF   INDF, 1
```

执行指令前

W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x20

执行后

W = 0x17
FSR = 0xC2
地址 (FSR) 的内容 = 0x02

例 3

```
SWAPF   REG, 1
```

执行指令前

REG1= 0xA5

执行后

REG1= 0x5A

TRIS 装载 TRIS 寄存器

语法：	[*标号*] TRIS f
操作数：	5 ≤ f ≤ 7
操作：	(W) → TRIS 寄存器 f；
影响的状态位：	无
机器码：	00 | 0000 | 0110 | 0fff
描述：	支持该指令是为了与 PIC16C5X 产品代码兼容。 TRIS 寄存器可读写，用户可直接对它们寻址。
指令字数：	1
指令周期：	1

例

为保持与未来 PIC16CXX 产品的向上兼容，不要使用该指令。

© 2004 Microchip Technology Inc.

XORLW 立即数异或 W 寄存器

语法： [*标号*] XORLW k

操作数： $0 \le k \le 255$

操作： (W).XOR. k → W

影响的状态位： Z

机器码：

11	1010	kkkk	kkkk

描述： 寄存器 W 的内容与 8 位立即数 'k' 进行异或运算。结果存入 W 寄存器。

指令字数： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 'k'	处理数据	写 W 寄存器

例 1

```
XORLW   0xAF                ; 1010 1111   (0xAF)
执行指令前                  ; 1011 0101   (0xB5)
        W   =  0xB5         ; ---------   ------
执行后                      ; 0001 1010   (0x1A)
        W   =  0x1A
        Z   =  0
```

例 2

```
XORLW   MYREG
执行指令前
        W   =  0xAF
        MYREG† 地址 = 0xB7
        † MYREG 是数据存储单元的符号
执行后
        W   =  0x18
        Z   =  0
```

例 3

```
XORLW   HIGH (LU_TABLE)
执行指令前
        W   =  0xAF
        LU_TABLE† 地址 = 0x9375
        † LU_TABLE 是程序存储单元的标号
        执行后
        W   =  0x3C
        Z   =  0
```

XORWF　　W 寄存器内容异或 f 寄存器内容

语法：　[*标号*]　XORWF　f,d

操作数：　0 ≤ f ≤ 127
d ∈ [0,1]

操作：　(W).XOR. (f) → 目标寄存器

影响的状态位：　Z

机器码：

00	0110	dfff	ffff

描述：　W 寄存器内容与 f 寄存器内容进行异或运算。如果 'd' 为 0，结果存入 W 寄存器；如果 'd' 为 1，结果存入寄存器 "f"。

指令字数：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 'f'	处理数据	写目标寄存器

例 1

```
XORWF    REG, 1                ; 1010 1111   (0xAF)
```

执行指令前

```
                               ; 1011 0101   (0xB5)
         REG=  0xAF            ; ---------   ------
         W   =  0xB5           ; 0001 1010   (0x1A)
```

执行后

```
         REG=  0x1A
         W   =  0xB5
```

例 2

```
XORWF    REG, 0                ; 1010 1111   (0xAF)
```

执行指令前

```
                               ; 1011 0101   (0xB5)
         REG=  0xAF            ; ---------   ------
         W   =  0xB5           ; 0001 1010   (0x1A)
```

执行后

```
         REG=  0xAF
         W   =  0x1A
```

例 3

```
XORWF   INDF, 1
```

执行指令前

```
         W   =  0xB5
         FSR =  0xC2
         地址 (FSR) 的内容 = 0xAF
```

执行后

```
         W   =  0xB5
         FSR =  0xC2
         地址 (FSR) 的内容 = 0x1A
```


© 2004 Microchip Technology Inc.

29.6 设计技巧

问 1： ***如何直接修改 W 寄存器的值？我想将 W 寄存器内容递减。***

答 1：

有多种方法，现给出两种方法：

1. 对于中档单片机来说，有几条对立即数和 W 寄存器进行操作的指令。例如，如果希望 W 寄存器减 1，可以用 ADDLW 0xFF （0x 向汇编器指明这是十六进制数）。
2. 注意所有指令都可以直接修改文件寄存器中的值。这意味着，可以直接将文件寄存器中的值减 1，而不需要先将文件寄存器中的值传送至 W 寄存器。如果想将文件寄存器的值减 1 后，再移至其它寄存器，可以先使用以 W 寄存器为目标寄存器的减 1 指令 (DECF 寄存器，W)，然后将 W 的值放到指定寄存器。这样做和直接将寄存器中的值移到另一个寄存器所用的指令数一样，但这样做的同时还完成了减 1 操作。

问 2： ***对于 PIC16CXXX 系列单片机，在使用 TRIS 指令时有危险吗？数据手册中建议不要使用该指令。***

答 2：

从代码兼容性和产品升级来说，不推荐使用 TRIS 指令。应该注意 TRIS 指令只能用于端口 A、B 和 C。将来的器件可能不支持 TRIS 指令。

问 3： ***在使用 TRIS 指令之前，一定要切换到数据存储器的 Bank1 吗（指存储器中包含 TRIS 寄存器的器件）？***

答 3：

不需要。TRIS 指令与存储区无关。不推荐使用 TRIS 指令。

问 4： ***读了数据手册上关于读－修改－写指令的描述后，我不太明白，能解释一下吗？为什么需要了解这些呢？***

答 4：

读－修改－写 (R-M-W) 指令的一个简单例子是位清零指令 BCF。你或许会认为单片机对端口输出引脚执行位清零指令，会直接将引脚清零。而实际上，单片机首先读整个端口（寄存器），然后进行位清零操作，最后将修改的新值写入端口（寄存器）。实际上，任何依赖于寄存器中当前值的指令都是读－修改－写指令，这包括 ADDWF、 SUBWF、 BCF、 BSF、 INCF 和 XORWF 等等。不依赖于寄存器当前值的指令，像 MOVWF、 CLRF 等，不是读－修改－写指令。

对于经常在输入和输出之间切换的端口，需要考虑读－修改－写指令的影响。例如，将所有 TRISB 位置 0，将所有 PORTB 引脚配置为输出，然后向 PORTB 寄存器写 0XFF，那么所有 PORTB 引脚都变为高电平。现在，假设将 RB3 引脚设置为输入，而其输入为低电平，然后执行 BCF PORTB,6 指令，将 RB6 引脚驱动为低电平。这时如果将 RB3 重新设置为输出，尽管最后写到这一位的是 ‘1’，但其输出为低电平。这是由于对 RB6 引脚进行 BCF 操作时，先读入整个 B 端口，包括 RB3 引脚为输入时的 ‘0’。然后 bit 6 清零，结果被写回 B 端口的锁存器，这时由于 RB3 读为 ‘0’，这个 ‘0’ 值被写回数据锁存器，将原来的值 ‘1’ 覆盖掉。因此当引脚变为输出时，RB3 将输出新值 （‘0’）。

问 5： ***当执行 BCF 指令时，端口的其它引脚被清零了，为什么？***

答 5：

这有几种可能性，其中两种可能性为：

1. 另举一个读－修改－写指令意外改变其它引脚值的例子：假设将PORTC所有引脚配置为输出，并将引脚驱动为低电平。在每个端口引脚上连接一个另一端接地的 LED，这样引脚输出高电平时将点亮 LED。每个 LED 旁并联了一个 100 µF 的电容。同样假设单片机的运行速度非常快，比方说 20 MHz。现在如果依次置 1 各个引脚：BSF PORTC,0，然后 BSF PORTC,1 ，然后 BSF PORTC,2 等等，你可能会发现此时只有最后一个引脚被置 1，只有最后一个 LED 被点亮。这是因为电容充电需要时间。当置 1 每个引脚时，其前一个引脚并没有充电到高电平，所以前一个引脚读为零，这个零值被写回端口锁存器 (记住：执行的是读－修改－写指令) ，从而使该位清零。当单片机运行速度比较高，进行连续端口操作时，通常需要考虑这个问题。
2. 如果使用的是 PIC16C7X 系列单片机，你可能没有在 ADCON1 寄存器中正确地配置 I/O 引脚。如果一个引脚配置为模拟输入，那么无论引脚上的电压为多少，每次读该引脚时，其结果均为零。总是读引脚状态是一个特例。你可以通过 TRIS 寄存器将模拟引脚设置成输出，然后写该引脚使其输出高电平或低电平，但是读该引脚时结果将始终为零。因此执行读－修改－写指令 (参见前面的问题) 时 ，所有模拟引脚均读为零，那些未被指令直接修改的读入值可以将端口锁存器重新写为零。对于设置成模拟的引脚，其引脚值可能既不是逻辑高电平也不是逻辑低电平，也不是浮动的。数字引脚应禁止输入端悬空，因为可能导致输入缓冲器电流消耗过大而使输入缓冲器工作失效。

© 2004 Microchip Technology Inc.

29.7 相关应用笔记

本部分列出了与本章内容相关的应用笔记。这些应用笔记并非都是专门针对中档单片机系列而写的（即有些针对低档系列，有些针对高档系列），但是其概念是相近的，通过适当修改并受到一定的限制即可使用。目前与指令集相关的应用笔记有：

目前没有相关的应用笔记

29.8 版本历史

版本 A

这是描述指令集的初始发行版。

© 2004 Microchip Technology Inc.