



16 位 MCU 和 DSC 程序员参考手册

高性能单片机（MCU）
和数字信号控制器（DSC）

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 (Digital Millennium Copyright Act)》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，在 Microchip 知识产权保护下，不得暗中以其他方式转让任何许可证。

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。Microchip 的 PIC[®] MCU 与 dsPIC[®] DSC、KEELOQ[®] 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949 ==

商标

Microchip 的名称和徽标组合、Microchip 徽标、AnyRate、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、KeeLoq 徽标、Kleer、LANCheck、LINK MD、MediaLB、MOST、MOST 徽标、MPLAB、OptoLyzer、PIC、PICSTART、PIC32 徽标、RightTouch、SpyNIC、SST、SST 徽标、SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、ETHERSYNCH、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge 和 QUIET-WIRE 均为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKIT 徽标、CodeGuard、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、MiWi、motorBench、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICKtail、PureSilicon、RightTouch 徽标、REAL ICE、Ripple Blocker、Serial Quad I/O、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2011-2016, Microchip Technology Inc. 版权所有。

ISBN: 978-1-5224-0548-1

目录

| | 页码 |
|--|-----------|
| 第 1 章 简介 | 5 |
| 简介 | 6 |
| 本手册的宗旨 | 6 |
| 开发支持 | 6 |
| 风格和符号的约定 | 7 |
| 指令集符号 | 8 |
| 第 2 章 编程模型 | 9 |
| 16 位 MCU 和 DSC 内核架构概述 | 10 |
| 编程模型 | 14 |
| 工作寄存器阵列 | 18 |
| 默认工作寄存器 (WREG) | 18 |
| 软件堆栈帧指针 | 18 |
| 软件堆栈指针 | 19 |
| 堆栈指针限制寄存器 (SPLIM) | 19 |
| 累加器 A 和累加器 B (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 19 |
| 程序计数器 | 19 |
| TBLPAG 寄存器 | 19 |
| PSVPAG 寄存器 (PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件) | 19 |
| RCOUNT 寄存器 | 20 |
| DCOUNT 寄存器 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 20 |
| DOSTART 寄存器 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 20 |
| DOEND 寄存器 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 21 |
| 状态寄存器 | 21 |
| 内核控制寄存器 | 24 |
| 影子寄存器 | 24 |
| DO 堆栈 (dsPIC33E 器件) | 25 |
| 第 3 章 指令集概述 | 37 |
| 简介 | 38 |
| 指令集概述 | 38 |
| 指令集汇总表 | 40 |
| 第 4 章 指令集详解 | 51 |
| 数据空间寻址模式 | 52 |
| 程序空间寻址模式 | 61 |
| 指令停顿 | 62 |
| 字节操作 | 64 |
| 字传送操作 | 66 |
| 使用 10 位立即数操作数 | 69 |
| 软件堆栈指针和帧指针 | 70 |
| 条件转移指令 | 76 |
| Z 状态位 | 77 |
| 规定的工作寄存器用法 | 78 |
| DSP 数据格式 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 81 |
| 累加器的使用 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 83 |
| 累加器访问 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 84 |
| DSP MAC 类指令 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 84 |
| DSP 累加器类指令 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 88 |
| 使用 FBCL 指令换算数据 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 88 |
| 使用 FBCL 指令将累加器中内容归一化 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件) | 90 |

16 位 MCU 和 DSC 程序员参考手册

| | |
|---|------------|
| 使用混合符号乘法的扩展精度算术运算（仅限 dsPIC33E 器件） | 91 |
| 第 5 章 指令描述 | 93 |
| 指令符号 | 94 |
| 指令编码字段描述符介绍 | 94 |
| 指令描述示例 | 98 |
| 指令描述 | 99 |
| 第 6 章 内建函数 | 445 |
| 简介 | 446 |
| 内建函数列表 | 447 |
| 第 7 章 参考信息 | 483 |
| 指令位映射 | 484 |
| 指令集汇总表 | 486 |
| 版本历史 | 496 |
| 第 8 章 索引 | 497 |
| 第 9 章 全球销售及服务网点 | 502 |

第 1 章 简介

目录

本章包括以下主要内容：

| | | |
|-----|----------------|---|
| 1.1 | 简介 | 6 |
| 1.2 | 本手册的宗旨 | 6 |
| 1.3 | 开发支持 | 6 |
| 1.4 | 风格和符号的约定 | 7 |
| 1.5 | 指令集符号 | 8 |

1.1 简介

Microchip 主要生产满足嵌入式控制市场需求的产品。Microchip 是以下器件和产品的领先供应商：

- 8 位通用单片机（PIC® MCU）
- 16 位数字信号控制器（dsPIC® DSC）
- 16 位和 32 位单片机（MCU）
- 专用和标准的非易失性存储器件
- 安防器件（KEELOQ® 安防 IC）
- 专用标准产品

有关这些器件和产品的信息以及相应的技术文档，可从 Microchip 网站（www.microchip.com）获得。

1.2 本手册的宗旨

本手册是 16 位 MCU 和 DSC 系列器件软件开发人员的参考手册。本手册对指令集进行了详细介绍，并提供通用信息以帮助用户进行 16 位 MCU 和 DSC 系列器件的软件开发。

本手册的内容没有包括有关内核、外设、系统集成或针对具体器件的详细信息。有关器件内核、外设以及系统集成的信息，请参阅具体器件的系列参考手册。有关针对具体器件的信息，请参阅具体器件的数据手册。数据手册提供的信息包括：

- 器件存储器映射
- 器件引脚和封装细节
- 器件电气特性
- 器件上包含的外设

本手册中给出了大量的代码示例。这些代码示例适用于 16 位 MCU 和 DSC 系列中的任何器件。

1.3 开发支持

Microchip 提供了大量的开发工具，使用户可以高效地开发和调试应用代码。Microchip 的开发工具可以分为四类：

- 代码生成
- 硬件 / 软件调试
- 器件编程器
- 产品评估板

用户可从 Microchip 网站（www.microchip.com）或当地的 Microchip 销售办事处获取有关最新工具、产品简介和用户指南的信息。

Microchip 还提供了其他参考工具来帮助用户加速开发过程，包括：

- 应用笔记
- 参考设计
- Microchip 网站
- 当地销售办事处提供的现场应用支持
- 公司的技术支持热线

Microchip 的网站列出了其他一些有用的链接。

1.4 风格和符号的约定

本文档采用了某些风格和字体格式约定。表 1-1 说明了本文档中所使用的约定。

表 1-1: 文档约定

| 符号或术语 | 说明 |
|----------------------------|--|
| 置 1 | 强制某一位 / 寄存器的值为逻辑 1。 |
| 清零 | 强制某一位 / 寄存器的值为逻辑 0。 |
| 复位 | 1. 强制某一寄存器 / 位回到默认状态。 2. 复位器件后的状态。某些位将被强制为 0（如中断允许位），而其他位被强制为 1（如 I/O 数据方向位）。 |
| 0xnnnn | 指定数据 nnnn 为十六进制数。这种约定用于代码示例中。例如，0x013F 或 0xA800。 |
| : (冒号) | 用来指定范围，或寄存器 / 位 / 引脚的组合。 如 ACCAU:ACCAH:ACCAL 表示由三个寄存器组成一个 40 位累加器。 组合顺序（左 - 右）通常指定一种位置关系（MSb 到 LSb，高位到低位）。 |
| < > | 指定特定寄存器中的位位置。 如 SR<7:5>（或 IPL<2:0>）指定了寄存器及其相关位或位位置。 |
| LSb 和 MSb | 表示位域中的最低位或最高位。 |
| LSB 和 MSB | 表示位域中的最低字节 / 最高字节。 |
| lsw 和 msw | 表示位域中的最低位字 / 最高位字。 |
| Courier New 字体 | 用于代码示例、二进制数以及文本中的指令助记符。 |
| Times New Roman 字体, 斜体 | 用于公式和变量。 |
| Times New Roman 字体, 粗体, 斜体 | 用于图表、公式或示例中的说明文本。 |
| 注: | “注”表示希望再次强调的信息，帮助您避免常见的错误，或提醒您注意同系列器件间的操作区别。“注”可以位于方框中；或用于表格或图中，这时它位于表格或图的下方。 |

1.5 指令集符号

在第 3.2 节“指令集概述”和第 7.2 节“指令集汇总表”中的汇总表以及第 5.4 节“指令描述”的指令描述中使用了表 1-2 中所示的符号。

表 1-2: 指令汇总表和说明中使用的符号

| 符号 ⁽¹⁾ | 说明 |
|-------------------|--------------------------------------|
| { } | 可选字段或操作 |
| [text] | 由文本寻址的地址 |
| (text) | 文本内容 |
| #text | 由文本定义的立即数 |
| $a \in [b, c, d]$ | a 一定在集合 [b, c, d] 中 |
| <n:m> | 寄存器位段 |
| {label:} | 可选标号名 |
| Acc | 累加器 A 或累加器 B |
| AWB | 累加器回写 |
| bit4 | 4 位宽位位置（对于字节模式为 0:7，对于字模式为 0:15） |
| Expr | 绝对地址、标号或表达式（由链接器解析） |
| f | 文件寄存器地址 |
| lit1 | 1 位立即数（0:1） |
| lit4 | 4 位立即数（0:15） |
| lit5 | 5 位立即数（0:31） |
| lit8 | 8 位立即数（0:255） |
| lit10 | 10 位立即数（对于字节模式为 0:255，对于字模式为 0:1023） |
| lit14 | 14 位立即数（0:16383） |
| lit16 | 16 位立即数（0:65535） |
| lit23 | 23 位立即数（0:8388607） |
| Slit4 | 有符号 4 位立即数（-8:7） |
| Slit6 | 有符号 6 位立即数（-32:31）（范围限制为 -16:16） |
| Slit10 | 有符号 10 位立即数（-512:511） |
| Slit16 | 有符号 16 位立即数（-32768:32767） |
| TOS | 栈顶 |
| Wb | 基本工作寄存器 |
| Wd | 目标工作寄存器（直接寻址和间接寻址） |
| Wdo | 目标工作寄存器（直接寻址和间接寻址，包括带偏移量的间接寻址） |
| Wm 和 Wn | 除法工作寄存器对（被除数和除数） |
| Wm * Wm | 乘法工作寄存器对（相同的源寄存器） |
| Wm * Wn | 乘法工作寄存器对（不同的源寄存器） |
| Wn | 既是源工作寄存器又是目标工作寄存器（直接寻址） |
| Wnd | 目标工作寄存器（直接寻址） |
| Wns | 源工作寄存器（直接寻址） |
| WREG | 默认工作寄存器（分配给 W0） |
| Ws | 源工作寄存器（直接和间接寻址） |
| Wso | 源工作寄存器（直接寻址和间接寻址，包括带偏移量的间接寻址） |
| Wx | 源寻址模式和用于 X 数据总线预取操作的工作寄存器 |
| Wxd | 用于 X 数据总线预取操作的目标工作寄存器 |
| Wy | 源寻址模式和用于 Y 数据总线预取操作的工作寄存器 |
| Wyd | 用于 Y 数据总线预取操作的目标工作寄存器 |

注 1: 每个符号的范围取决于指令。对于特定指令的范围，请参见第 5 章“指令描述”。

第 2 章 编程模型

目录

本章包括以下主要内容：

| | | |
|-----|-----------------------------|----|
| 2.1 | 16 位 MCU 和 DSC 内核架构概述 | 10 |
| 2.2 | 编程模型 | 14 |
| 2.3 | 工作寄存器阵列 | 18 |
| 2.4 | 默认工作寄存器（WREG） | 18 |
| 2.5 | 软件堆栈帧指针 | 18 |

2.1 16 位 MCU 和 DSC 内核架构概述

本节简要介绍以下器件系列的 16 位架构特性和功能：

- 16 位单片机（MCU）：
 - PIC24F
 - PIC24H
 - PIC24E
- 16 位数字信号控制器（DSC）：
 - dsPIC30F
 - dsPIC33F
 - dsPIC33E

2.1.1 特定于 16 位 MCU 和 DSC 内核的特性

16 位 MCU 和 DSC 器件采用改进的哈佛架构内核，数据线宽度为 16 位，采用增强指令集。内核采用具有 8 位操作码字段的 24 位指令字。程序计数器（Program Counter, PC）为 23 位宽，可对最大为 4M x 24 位的用户程序存储空间进行寻址。指令预取机制用来帮助维持吞吐量并提供可预测的执行。大多数指令都在单个周期内执行。

2.1.1.1 寄存器

16 位 MCU 和 DSC 器件拥有 16 个 16 位工作寄存器。每个工作寄存器可作为数据、地址或偏移量寄存器。第 16 个工作寄存器（W15）用作中断和调用时的软件堆栈指针。

2.1.1.2 指令集

16 位 MCU 和 DSC 架构具有几乎完全相同的指令集。该指令集包括多种寻址模式，且设计为确保最佳的 C 编译器效率。

2.1.1.3 数据空间寻址

数据空间可寻址为 32K 字或 64 KB。可以选择以任何 16K 程序字为边界将数据存储空间的高 32 KB 映射到程序存储空间，这一功能称为程序空间可视性（Program Space Visibility, PSV）。程序存储空间到数据存储空间的映射功能让任何指令都能像访问数据存储空间一样访问程序存储空间，这对于数据常数的存放是非常有用的。

注： 一些器件系列支持扩展数据（Extended Data Space, EDS）空间寻址。有关该特性的更多详细信息，请参见具体器件的数据手册和系列参考手册。

2.1.1.4 寻址模式

内核支持固有（无操作数）寻址、相对寻址、立即数寻址、存储器直接寻址、寄存器直接寻址、寄存器间接寻址以及寄存器偏移量寻址模式。根据功能性要求的不同，每一条指令都与预先定义的寻址模式组相关联。任何一条指令可支持多达 7 种寻址模式。

对于大多数指令，CPU 可在每一指令周期内执行数据（或程序数据）存储器读、工作寄存器（数据）读、数据存储器写以及程序（指令）存储器读操作。因此可支持 3 操作数指令，即允许在单个周期内执行 $A + B = C$ 操作。

2.1.1.5 算术逻辑单元

包括一个高速 17 位 x 17 位乘法器，显著提高了内核的运算能力和吞吐量。乘法器支持有符号、无符号和混合符号模式的 16 位 x 16 位或 8 位 x 8 位整数乘法。所有乘法指令都在单个周期内执行。

16 位算术逻辑单元（Arithmetic Logic Unit, ALU）用支持迭代不恢复余数除法算法的整数除法支持硬件进行了增强。它和 REPEAT 指令循环机制（以及一些迭代除法指令）一起使用，支持用 16 位整数除 32 位（或 16 位）的有符号和无符号除法。所有除法运算都需要 19 个周期来完成，但可以在任何周期边界被中断。

2.1.1.6 异常处理

16 位 MCU 和 DSC 器件具有向量异常机制，可支持最多 8 个不可屏蔽陷阱源和最多 246 个中断源。两个系列都可为每个中断源分配 7 个优先级之一。

2.1.2 PIC24E 和 dsPIC33E 特性

在第 2.1.1 节“特定于 16 位 MCU 和 DSC 内核的特性”中提供的信息之外，本节将介绍 PIC24E 和 dsPIC33E 器件系列的增强特性。

2.1.2.1 数据空间寻址

基本数据空间地址与读页寄存器或写页寄存器（DSRPAG 或 DSWPAG）配合使用，可构成扩展数据空间（Extended Data Space, EDS）地址，该地址还可用于 PSV 访问。EDS 可以作为 8M 字或 16 MB 寻址。有关 EDS、PSV 和表访问的更多详细信息，请参见《dsPIC33E/PIC24E 系列参考手册》中的第 3 章“数据存储”（DS70595）。

注：一些 PIC24F 器件还支持扩展数据空间。有关详细信息，请参见《PIC24F 系列参考手册》的第 44 章“具有 EDS 的 CPU”（DS39732）和第 45 章“具有 EDS 的数据存储器”（DS39733）。

2.1.2.2 自动混合符号乘法模式（仅限 dsPIC33E 器件）

除有符号和无符号 DSP 乘法外，dsPIC33E 器件还支持混合符号（无符号 - 有符号和有符号 - 无符号）乘法，无需因操作数格式不同而动态重新配置乘法模式和移位数据。该模式尤其有助于执行扩展精度（32 位和 64 位）算法。除 DSP 指令外，MCU 乘法（MUL）指令还能利用任一累加器作为结果的目标寄存器，从而实现更快速的扩展精度算术运算。关于混合符号 DSP 乘法的更多详细信息，请参见第 4.10.1 节“隐含的 DSP 操作数（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”和第 4.18 节“使用混合符号乘法的扩展精度算术运算（仅限 dsPIC33E 器件）”。

2.1.2.3 16 位结果的 MCU 乘法

16x16 位 MUL 指令包含一个将乘积存储到一个而非一对 16 位工作寄存器的选项。在相乘的数量值较小而预期得到 16 位结果的情况下，该特性可释放一个寄存器用于其他用途。更多详细信息，请参见第 5.4 节“指令描述”中各 MUL 指令的说明。

2.1.2.4 用于 DO 循环的硬件堆栈（仅限 dsPIC33E 器件）

单级深度 DO 循环影子寄存器集为 4 级深度的 DO 循环硬件堆栈所替代。这为最多 3 层的 DO 循环嵌套提供 DO 循环寄存器的自动保存 / 存储，从而可更高效地实现循环嵌套。关于 dsPIC33E 器件中的 DO 循环嵌套的更多详细信息，请参见第 2.19 节“DO 堆栈（dsPIC33E 器件）”。

2.1.2.5 DSP 现场切换支持（仅限 dsPIC33E 器件）

在 dsPIC33E 器件中，DSP 溢出和饱和状态位是可写的。这允许在 DSP 任务间切换时高效保存和存储 DSP 引擎的状态。关于 DSP 状态位的更多详细信息，请参见第 2.16.4 节“DSP ALU 状态位（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”。

2.1.2.6 扩展的 CALL 和 GOTO 指令

新的 CALL.L Wn 和 GOTO.L Wn 指令通过使能计算转移 / 调用目标的 32 位地址，扩展了 CALL Wn 和 GOTO Wn 指令的功能。在这些增强型指令中，目标地址由一对工作寄存器而非一个 16 位寄存器提供。更多详细信息，请参见第 5 章“指令描述”中 CALL.L 和 GOTO.L 指令的说明。

2.1.2.7 比较 - 转移指令

dsPIC33E/PIC24E 器件具有条件比较 - 转移（CPBxx）指令。这些指令允许转移而非仅仅跳过单条指令，从而扩展了比较 - 跳过（CPSxx）指令的功能。关于比较 - 转移指令的更多详细信息，请参见第 5 章“指令描述”中 CPBEQ、CPBNE、CPBGT 和 CPBLT 指令的说明。

2.1.3 dsPIC30F、dsPIC33F 和 dsPIC33E 特性

在第 2.1.1 节“特定于 16 位 MCU 和 DSC 内核的特性”中提供的信息之外，本节将介绍 dsPIC30F、dsPIC33F 和 dsPIC33E 系列器件的 DSP 增强特性。

2.1.3.1 程序循环结构

使用 DO 指令支持无开销的程序循环结构，该指令可被中断。

2.1.3.2 DSP 类指令

DSP 类指令无缝地集成到架构中，并从单个执行单元执行。

2.1.3.3 数据空间寻址

数据空间被分成两块，称为 X 和 Y 数据存储空间。每个存储块有各自独立的地址发生单元（Address Generation Unit, AGU）。MCU 类指令只通过 X 数据空间 AGU 进行操作，可将整个存储器映射作为一个线性数据空间访问。那些具有两个源操作数的 DSP 类指令通过 X 和 Y 的 AGU 进行操作，这将数据地址空间分成两个部分，X 和 Y 数据空间的边界视具体器件而定。

2.1.3.4 模寻址和位反转寻址

X 和 Y 地址空间都支持无开销循环缓冲区（模寻址）。模寻址省去了 DSP 算法的软件边界检查开销。此外，X AGU 的循环寻址可以与任何 MCU 类指令一起使用。X AGU 还支持位反转寻址，大幅简化了基为 2 的 FFT 算法对输入或输出数据的重新排序。

2.1.3.5 DSP 引擎

DSP 引擎具备一个高速 17 位 x 17 位乘法器、一个 40 位 ALU、两个 40 位饱和累加器和一个 40 位双向桶形移位寄存器。该桶形移位寄存器可在单个周期内将一个 40 位的值右移最多 16 位或左移最多 16 位。DSP 类指令可以无缝地与所有其他指令一起操作，设计为可实现最佳的实时性能。MAC 类指令和其他相关指令可以同时从存储器中取出两个数据操作数并将两个工作寄存器相乘。这要求数据空间对于这些指令拆分为两块，但对所有其他指令保持线性。这是通过为每个地址空间指定某些工作寄存器，以透明和灵活的方式实现的。

2.1.3.6 异常处理

dsPIC30F 器件具有向量异常机制，可支持最多 8 个不可屏蔽陷阱源和最多 54 个中断源。dsPIC33F 和 dsPIC33E 具有相似的向量异常机制，不同的是可分别支持最多 118 个中断源和最多 246 个中断源。所有三个系列都可为每个中断源分配 7 个优先级之一。

关于异常处理的更多详细信息，请参见《dsPIC30F 系列参考手册》的第 6 章和第 28 章“中断”、《dsPIC33F/PIC24H 系列参考手册》的第 6、29、32、41、47 和 53 章以及《dsPIC33E/PIC24E 系列参考手册》的第 6 章。

2.2 编程模型

图 2-1 至图 2-4 给出了 16 位 MCU 和 DSC 系列器件的编程模型图。

图 2-1: PIC24F 和 PIC24H 编程模型图

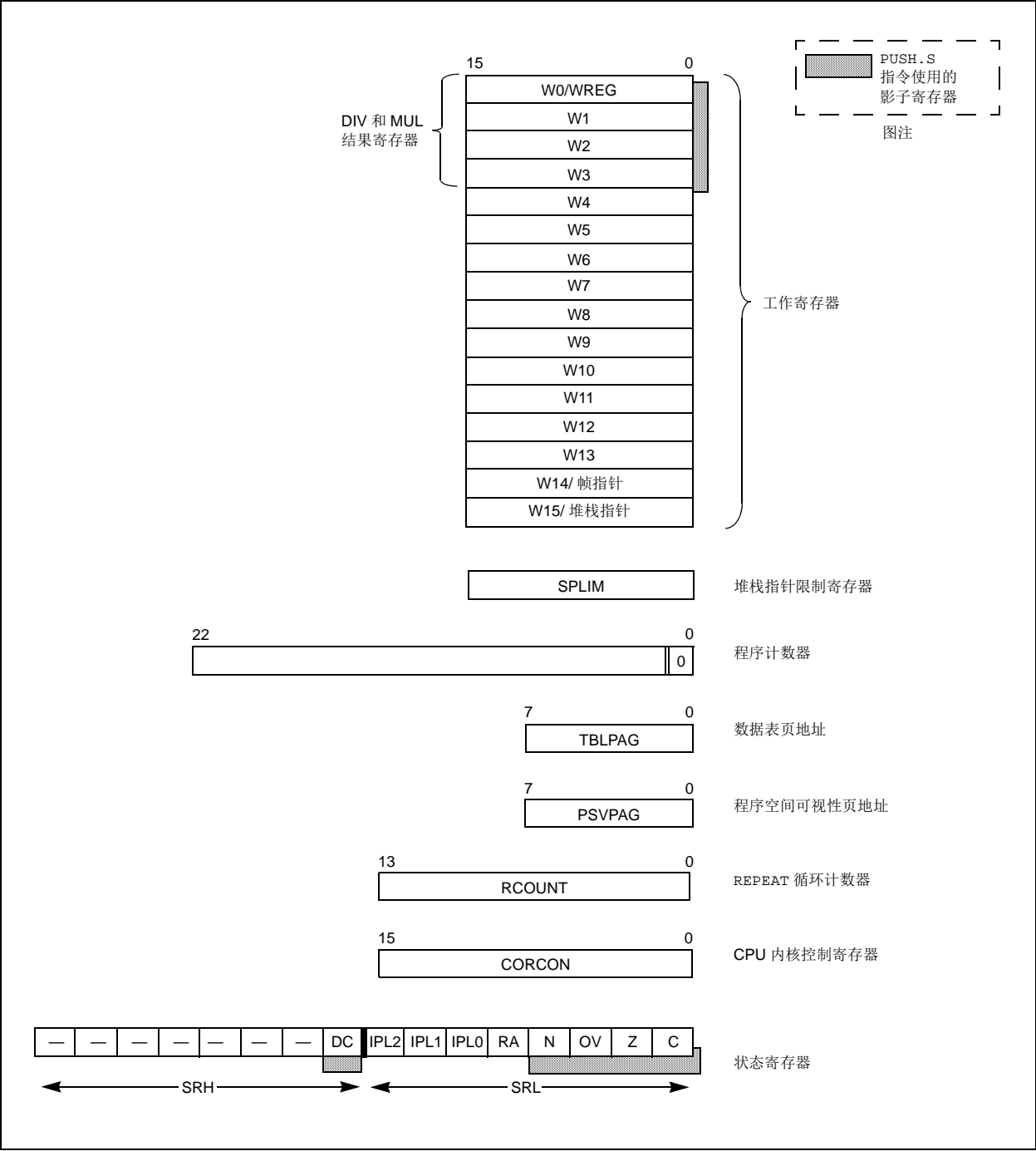


图 2-2: PIC24E 编程模型图

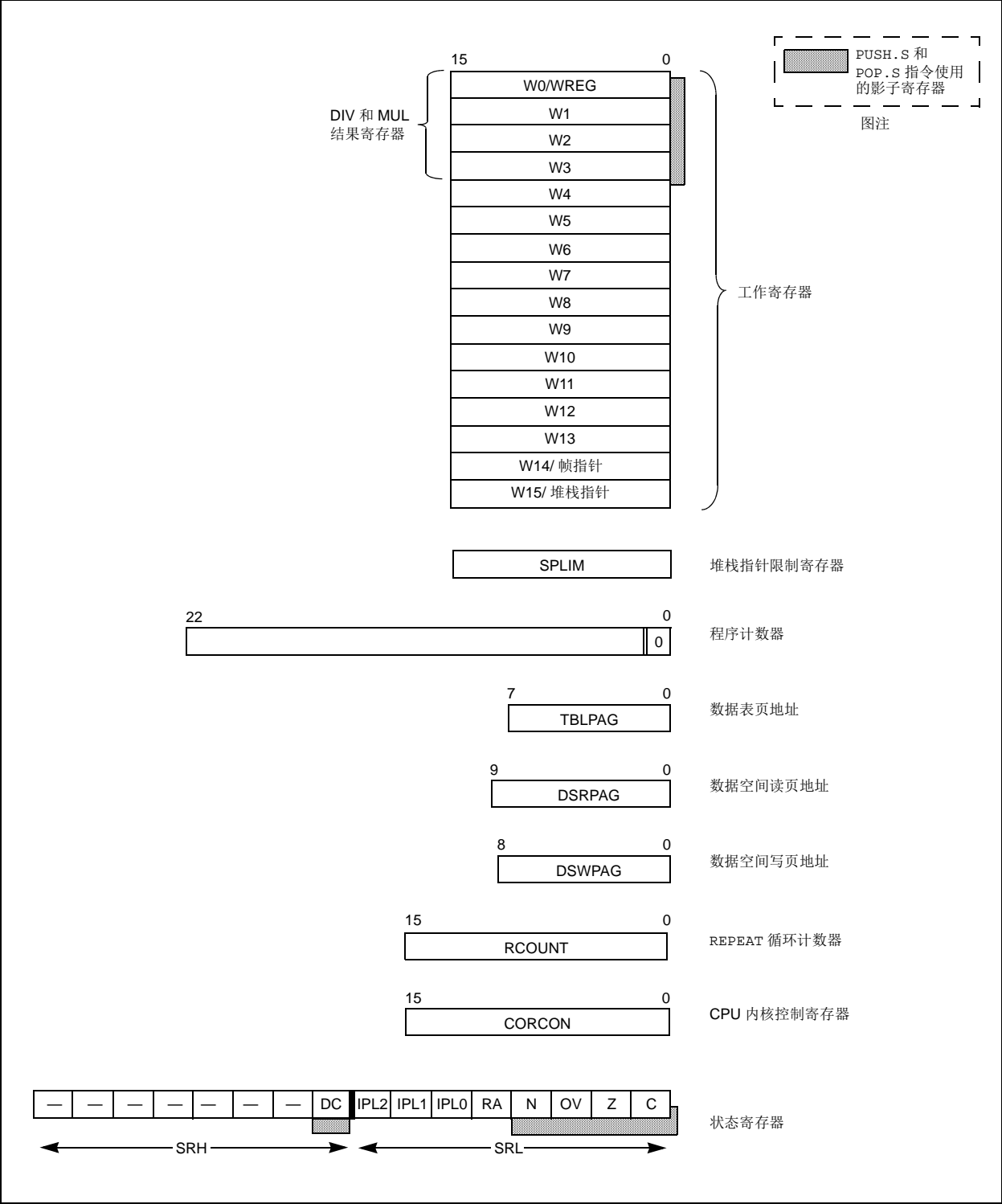


图 2-3: dsPIC30F 和 dsPIC33F 编程模型图

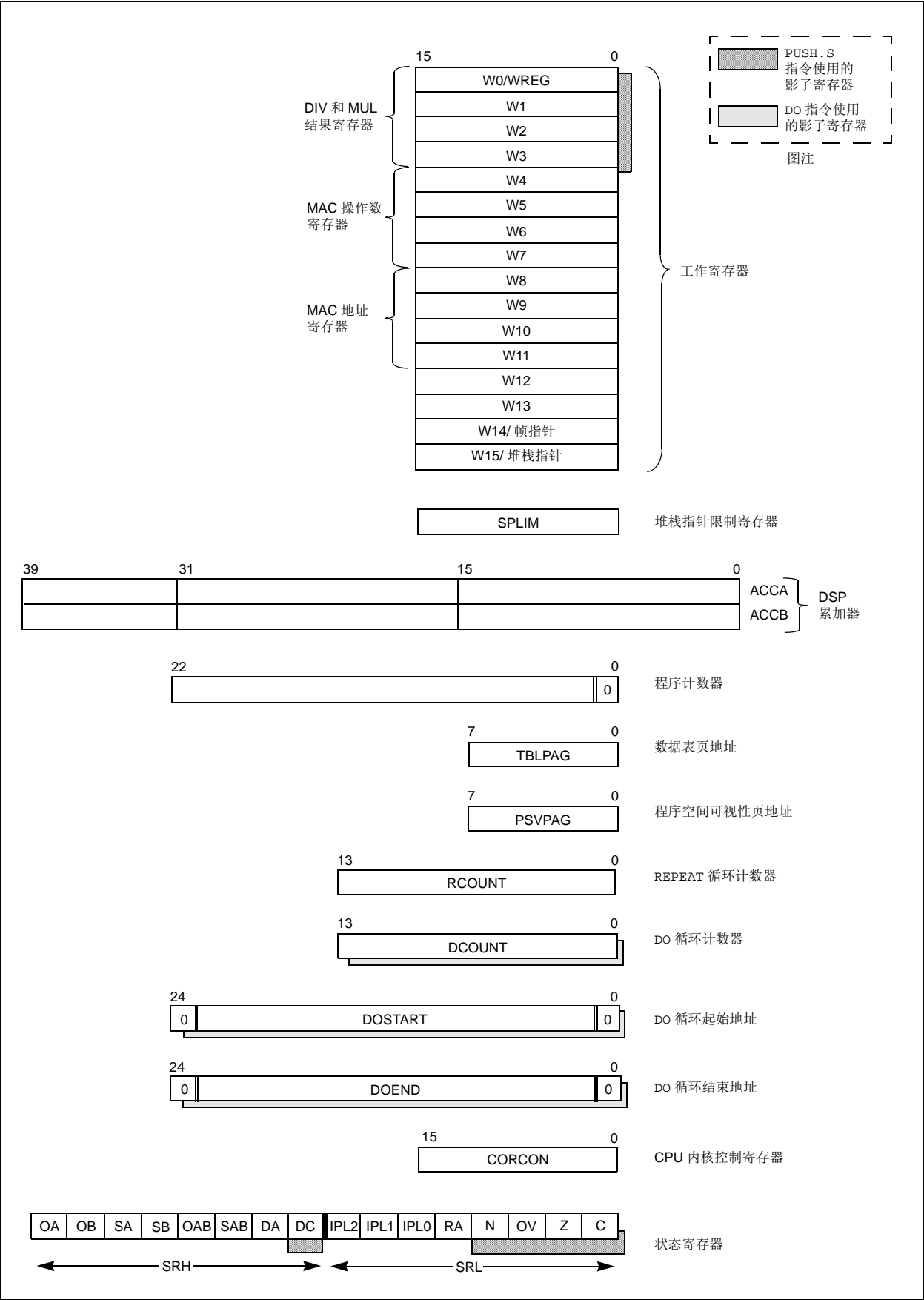
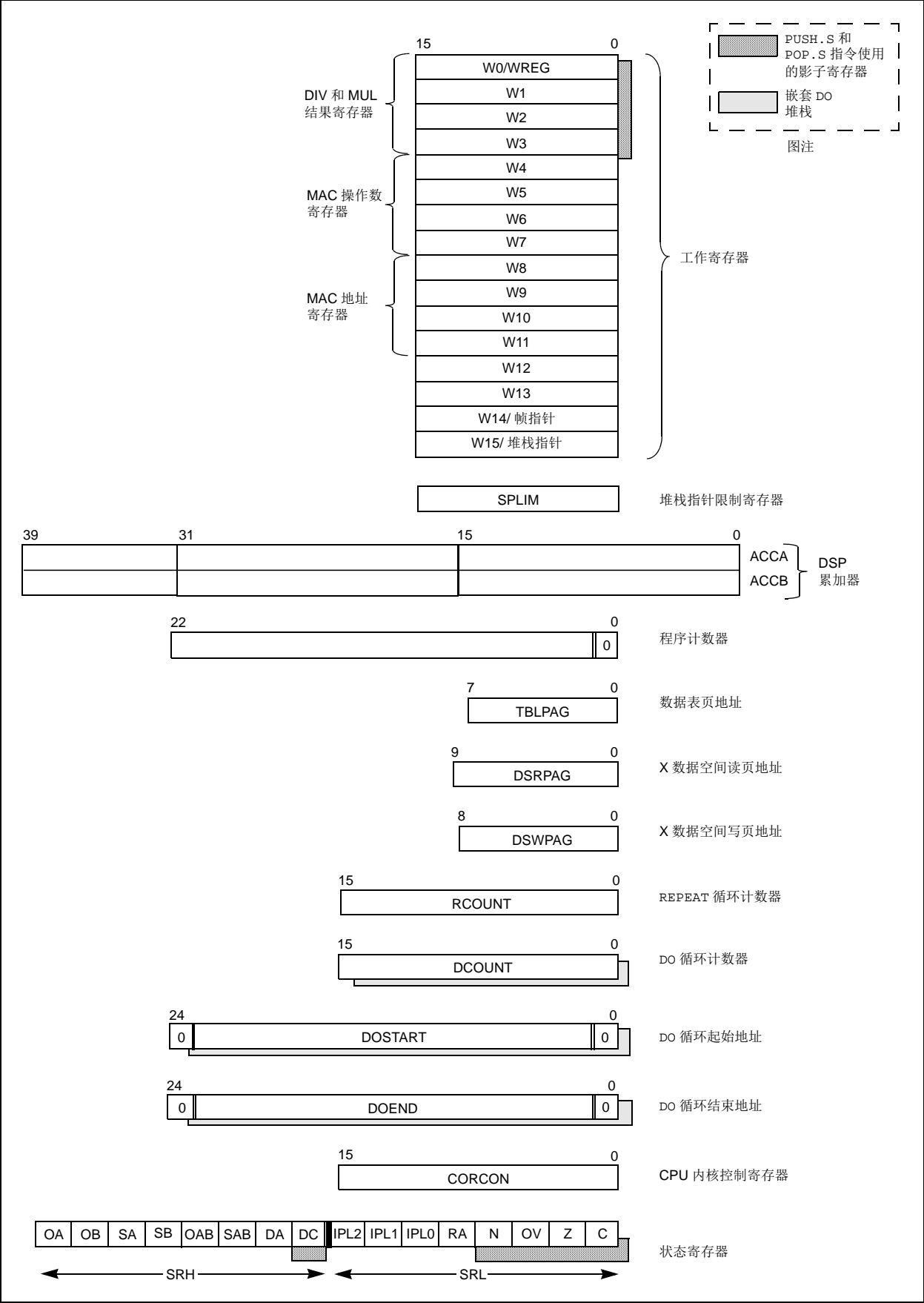


图 2-4: dsPIC33E 编程模型图



编程模型中的所有寄存器皆为存储器映射的，并能直接被指令集操作。表 2-1 提供了每一个寄存器的说明。

注： 除非另外指定，否则表 2-1 中的编程模型寄存器说明适用于所有 MCU 和 DSC 器件系列。

表 2-1: 编程模型寄存器说明

| 寄存器 | 说明 |
|----------------------------|-------------------|
| CORCON | CPU 内核配置寄存器 |
| PC | 23 位程序计数器 |
| PSVPAG ⁽¹⁾ | 程序空间可视性页地址寄存器 |
| DSRPAG ⁽²⁾ | 扩展数据空间（EDS）读页寄存器 |
| DSWPAG ⁽²⁾ | 扩展数据空间（EDS）写页寄存器 |
| RCOUNT | REPEAT 循环计数器寄存器 |
| SPLIM | 堆栈指针限制值寄存器 |
| SR | ALU 和 DSP 引擎状态寄存器 |
| TBLPAG | 表存储器页地址寄存器 |
| W0-W15 | 工作寄存器阵列 |
| ACCA 和 ACCB ⁽³⁾ | 40 位 DSP 累加器 |
| DCOUNT ⁽³⁾ | DO 循环计数器寄存器 |
| DOSTART ⁽³⁾ | DO 循环起始地址寄存器 |
| DOEND ⁽³⁾ | DO 循环结束地址寄存器 |

注 1： 该寄存器仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件上可用。

注 2： 该寄存器仅在 PIC24E 和 dsPIC33E 器件上可用。

注 3： 该寄存器仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件上可用。

2.3 工作寄存器阵列

16 个工作（W）寄存器可作为数据、地址或偏移量寄存器。W 寄存器的功能由访问它的指令所决定。

字节指令以工作寄存器阵列作为操作对象，将只对目标寄存器的低字节（Least Significant Byte, LSB）产生影响。由于工作寄存器是存储器映射的，因此可以字节宽度的数据存储空间访问方式对低字节和高字节进行操作。

2.4 默认工作寄存器（WREG）

指令集中的指令可分为两类：工作寄存器指令和文件寄存器指令。工作寄存器指令使用工作寄存器阵列作为数据值或作为指向某一存储单元的地址。相反，文件寄存器指令对包含在指令操作码中的特定存储地址进行操作。

文件寄存器指令也使用工作寄存器，但并不指定指令中所使用的工作寄存器，而是采用默认的工作寄存器（WREG）。工作寄存器 W0 指定为 WREG。用户无法对 WREG 的指定进行编程设定。

2.5 软件堆栈帧指针

帧是堆栈中用户定义的存储区域，函数使用帧来为局部变量分配存储区。W14 被指定作为分配堆栈帧（LNK）和释放堆栈帧（ULNK）指令中使用的堆栈帧指针。然而在未使用堆栈帧指针、LNK 和 ULNK 指令时，W14 可被任何指令以与其他 W 寄存器相同的方式使用。在 dsPIC33E 和 PIC24E 器件上，可以使用堆栈帧激活（SFA）状态位来支持嵌套堆栈帧。有关帧指针的详细信息，请参见第 4.7.2 节“软件堆栈帧指针”。

2.6 软件堆栈指针

W15 专用作软件堆栈指针，将会被函数调用、异常处理以及返回自动修改。然而，W15 可像所有其他 W 寄存器一样被任何指令引用。这将简化对堆栈指针的读、写和控制操作。有关堆栈指针的详细信息，请参见第 4.7.1 节“软件堆栈指针”。

2.7 堆栈指针限制寄存器（SPLIM）

SPLIM 是与堆栈指针有关的 16 位寄存器，用来防止堆栈指针上溢以及对超出用户分配堆栈存储区的存储空间进行访问。有关 SPLIM 的详细信息，请参见第 4.7.3 节“堆栈指针上溢”。

2.8 累加器 A 和累加器 B（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

累加器 A（ACCA）和累加器 B（ACCB）皆为 40 位宽的寄存器。DSP 类指令通过累加器执行数学和移位操作。每个累加器都是由 3 个存储区映射的寄存器构成：

- AccxU（bit 39-32）
- AccxH（bit 31-16）
- AccxL（bit 15-0）

在 dsPIC33E 器件中，也可将累加器 A 和累加器 B 用作 MCU MUL.xx 指令中的目标寄存器。这有助于缩短扩展精度算术运算的执行时间。

有关 ACCA 和 ACCB 使用的详细信息，请参见第 4.12 节“累加器的使用（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”。

2.9 程序计数器

程序计数器（PC）为 23 位宽。通过 PC<22:1> 对 4M x 24 位用户程序存储空间中的指令进行寻址，其中 PC<0> 总是设定为 0，以保持指令字对齐并提供与数据存储空间寻址的兼容性。这表明在正常的指令执行过程中，PC 值将以 2 为步长进行递增。

位于 0x800000 及其以上地址的程序存储区用于存放器件配置数据、用户部件 ID 以及器件 ID。该存储区域不能用来执行用户代码且 PC 也不能访问这个区域。然而，用户可通过使用表指令对这个区域进行访问。有关访问配置数据、用户部件 ID 和器件 ID 的详细信息，请参见具体器件的系列参考手册。

2.10 TBLPAG 寄存器

TBLPAG 寄存器用来在表读和表写操作期间存放程序存储地址的最高 8 位。表指令用来在程序存储空间和数据存储空间之间传送数据。有关使用表指令访问程序存储区的详细信息，请参见具体器件的系列参考手册。

2.11 PSVPAG 寄存器（PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件）

程序空间可视性功能允许用户将 32 KB 的程序存储空间映射到高 32 KB 的数据地址空间。该功能允许通过对数据存储区进行操作的指令透明访问常量数据。PSVPAG 寄存器用于选择映射到数据地址空间的 32 KB 程序存储空间。有关程序空间可视性功能的详细信息，请参见具体器件的系列参考手册。

2.12 RCOUNT 寄存器

14 位 RCOUNT 寄存器（对于 PIC24E 和 dsPIC33E 器件为 16 位）包含用于 REPEAT 指令的循环计数器。当执行 REPEAT 指令时，RCOUNT 将装载指令重复执行的次数，如“REPEAT #lit14”指令中的“lit14”（对于 PIC24E 和 dsPIC33E 器件为“REPEAT #lit15”指令中的“lit15”）或“REPEAT Wn”指令中 Wn 寄存器的低 14 位（对于 PIC24E 和 dsPIC33E 器件为整个 Wn 的内容）。REPEAT 循环将被执行 RCOUNT + 1 次。

- 注 1:** 如果 REPEAT 循环执行过程被中断，则当再次进入前台代码时中断服务程序可将 RCOUNT 清零以退出 REPEAT 循环。
- 2:** 有关 REPEAT 循环的详细信息，请参见具体器件的系列参考手册。

2.13 DCOUNT 寄存器（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

14 位 DCOUNT 寄存器（对于 dsPIC33E 器件为 16 位）包含用于硬件 DO 循环的循环计数器。当执行 DO 指令时，DCOUNT 将装载指令的循环计数值，如“DO #lit14,Expr”指令中的“lit14”（对于 dsPIC33E 器件为“DO #lit15,Expr”指令中的“lit15”）或“DO Ws,Expr”指令中 Ws 寄存器的低 14 位（对于 dsPIC33E 器件为整个 Ws 的内容）。DO 循环将被执行 DCOUNT + 1 次。

- 注 1:** 在 dsPIC30F 和 dsPIC33F 器件中，DCOUNT 寄存器包含一个影子寄存器。有关影子寄存器的信息，请参见第 2.18 节“影子寄存器”。
- 2:** dsPIC33E 器件具有一个 4 级深度的嵌套 DO 堆栈来代替影子寄存器。
- 3:** 有关 DO 循环的详细信息，请参见具体器件的系列参考手册。

2.14 DOSTART 寄存器（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DOSTART 寄存器包含硬件 DO 循环的起始地址。当执行 DO 指令时，DOSTART 将装载紧接 DO 指令的下一条指令的地址。该存储地址为 DO 循环的起始地址。当循环激活时，程序将在 DO 循环的最后一条指令执行之后继续执行 DOSTART 地址中存储的指令。这种机制可实现零开销循环。

- 注 1:** 对于 dsPIC30F 和 dsPIC33F 器件，DOSTART 具有一个影子寄存器。有关影子寄存器的信息，请参见第 2.18 节“影子寄存器”。
- 2:** dsPIC33E 器件具有一个 4 级深度的嵌套 DO 堆栈来代替影子寄存器。在 dsPIC33E 器件中，DOSTART 寄存器是只读的。
- 3:** 有关 DO 循环的完整详细信息，请参见具体器件的系列参考手册。

2.15 DOEND 寄存器（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DOEND 寄存器包含硬件 DO 循环的终止地址。当执行 DO 指令时，DOEND 将装载 DO 指令中的表达式所指定的地址。该存储地址指定 DO 循环中的最后一条指令。在循环激活且 DOEND 地址中存储的指令被执行后，程序将继续从 DO 循环起始地址（存储在 DOSTART 寄存器中）开始执行。

- 注**
- 1: 对于 dsPIC30F 和 dsPIC33F 器件，DOEND 具有一个影子寄存器。有关影子寄存器的信息，请参见第 2.18 节“影子寄存器”。
 - 2: dsPIC33E 器件具有一个 4 级深度的嵌套 DO 堆栈来代替影子寄存器。
 - 3: 有关 DO 循环的详细信息，请参见具体器件的系列参考手册。

2.16 状态寄存器

16 位状态寄存器用于保存最近执行指令的状态信息。其中操作状态位用于 MCU 操作、循环操作以及 DSP 操作。此外，状态寄存器还包含用于中断处理的 CPU 中断优先级位 IPL<2:0>。

根据不同的 MCU 和 DSC 系列，可以使用以下状态寄存器之一：

- 寄存器 2-1，适用于 PIC24F、PIC24H 和 PIC24E 器件
- 寄存器 2-2，适用于 dsPIC30F 和 dsPIC33F 器件
- 寄存器 2-3，适用于 dsPIC33E 器件

2.16.1 MCU ALU 状态位

指令集中的大多数指令都会影响或使用 MCU 操作状态位。大多数逻辑、数学、循环 / 移位和位操作指令都将在执行后修改 MCU 状态位。条件转移指令使用各状态位的状态来确定程序执行流。第 4.8 节“条件转移指令”列出了所有条件转移指令。

进位（C）、全零（Z）、溢出（OV）、负（N）和半进位（DC）位用来指示 MCU ALU 的当前状态。这些状态位的状态分别表明指令执行结果是否导致进位、全零、溢出、负的结果或半进位等事件。当执行减法操作时，C 标志位可用作借位标志。

Z 状态位对于扩展精度的算术运算很有用。除使用进位或借位输入的指令（ADDC、CPB、SUBB 和 SUBBR）外，所有指令都使用 Z 状态位作为普通 Z 标志。更多详细信息，请参见第 4.9 节“Z 状态位”。

- 注**
- 1: 在执行 PUSH.S 指令时，所有 MCU 位都被保存到影子寄存器中。在执行 POP.S 指令时，它们将恢复原先的内容。
 - 2: 在异常处理过程中，除 DC 标志（不在 SRL 中）外的所有 MCU 位都将被压入堆栈（见第 4.7.1 节“软件堆栈指针”）。

2.16.2 REPEAT 循环状态位

REPEAT 激活（REPEAT Active，RA）状态位用来指明循环是否处于激活状态。RA 标志表明 REPEAT 指令是否正在执行，而且只有 REPEAT 指令才会对该标志位的状态产生影响。当被重复的指令开始执行时，RA 标志位将被置 1。当被重复的指令完成最后一次执行时，该标志位将被清零。

由于 RA 标志也是只读位，因此也不能对该位直接清零。然而，如果 REPEAT 或其目标指令被中断，中断服务程序可对位于堆栈中 SRL 的 RA 标志位清零。该操作将在程序执行从中断服务程序返回时禁止循环操作，因此此时恢复的 RA 将为 0。

2.16.3 DO 激活 (DA) 状态位 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

DO 激活 (DO Active, DA) 状态位用来指明循环是否处于激活状态。DO 指令将对 DA 标志产生影响以表明 DO 循环处于激活状态。当执行 DO 循环中的第一条指令时，DA 标志位将被置 1。当循环中的最后一条指令执行完时，DA 标志位将被清零。

DA 标志为只读位。这表明不能通过写 1 到 DA 来启动循环，也不能通过清零 DA 来终止循环。如果必须提前终止 DO 循环，应使用 EDT 位 (CORCON<11>)。

2.16.4 DSP ALU 状态位（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DSP 类指令使用状态寄存器的高位字节（SRH），当数据经过其中一个加法器时，SRH 中内容将被修改。SRH 为两个累加器均提供了溢出和饱和状态信息。饱和 A、饱和 B、溢出 A 和溢出 B（SA、SB、OA 和 OB）状态位提供了单独的累加器状态。而饱和 AB 和溢出 AB（SAB 和 OAB）状态位提供了联合的累加器状态。SAB 和 OAB 状态位使得软件开发人员可高效地检查寄存器是否出现饱和或溢出状况。

OA 和 OB 状态位用来指示一个操作是否产生溢出至相应累加器的警戒位（bit 32-39）。只有当处理器处于超饱和模式或饱和模式被禁止时，这种情况才会发生。这表明用累加器的低 31 位无法表示操作产生的数值。在 dsPIC33E 器件中，OA 和 OB 位是可写的。

SA 和 SB 状态位用来指示一个操作是否从相应累加器的最高位产生溢出。无论何种饱和模式（禁止、普通或超饱和），SA 和 SB 位都处于“粘住”状态。即当 SA 或 SB 位置 1 时，该位只能由软件手动清零，而与后续的 DSP 操作无关。在需要时，可以使用 BCLR 指令对 SA 或 SB 位进行清零。

此外，在 dsPIC33E 器件中，SA 和 SB 位可以用软件置 1，从而支持高效的现场状态切换。

为简便起见，OA 和 OB 状态位通过逻辑或形成 OAB 标志，而将 SA 和 SB 状态位通过逻辑或形成 SAB 标志。当实现算法时，通过这些累加器状态位可实现高效的溢出和饱和检查。在进行算术运算溢出检查时，只需查询 OAB 即可完成，而不再需要单独查询 OA 和 OB 状态位。同样，当进行饱和检查时，可只检查 SAB 而不是所有 SA 和 SB 状态位。注意，清零 SAB 标志将同时对 SA 和 SB 位进行清零。

2.16.5 中断优先级状态位

SRL 中的三个中断优先级（Interrupt Priority Level, IPL）位 SR<7:5> 以及 IPL3 位（CORCON<3>）对用于异常处理的 CPU IPL 进行设定。异常包括中断和硬件陷阱。中断具有的优先级为 0 至 7，可由用户进行定义，而硬件陷阱的优先级是固定的，为 8 至 15。第 4 个中断优先级位 IPL3 为一个特殊的 IPL 位，只可由用户对其进行读或清零。该位只在硬件陷阱激活时才被置 1，在陷阱处理后被清零。

CPU 的 IPL 用来标识可中断处理器的最低优先级的异常。当等待处理的异常具有比 CPU IPL 更高的中断优先级时，CPU 才会对其进行处理。这表明，如果 IPL 为 0，所有优先级为 1 或更高的异常都可中断处理器。如果 IPL 为 7，只有硬件陷阱才可中断处理器。

当正在处理一个异常时，IPL 将自动设定为该异常的优先级，这将禁止所有同等或更低优先级的异常。然而，由于 IPL 位域为可读/写，用户可在中断服务程序中修改 IPL 的低 3 位值以控制何种异常具有优先处理权。由于在异常处理过程中 SRL 将被压入堆栈，因此总是在异常处理之后恢复原先的 IPL。如果需要，用户也可通过置 1 NSTDIS 位（INTCON1<15>）来阻止异常嵌套。

注： 有关异常处理的更多详细信息，请参见具体器件的系列参考手册。

2.17 内核控制寄存器

对于所有 MCU 和 DSC 器件，16 位 CPU 内核控制寄存器（CORCON）用来对 CPU 的配置进行设定。该寄存器能将程序存储空间映射到数据存储空间。

除设置 CPU 模式外，CORCON 寄存器还包含表明是否正在处理陷阱异常 IPL<3> 状态位的状态信息。

根据不同的 MCU 和 DSC 系列，可以使用以下 CORCON 寄存器之一：

- 寄存器 2-4，适用于 PIC24F 和 PIC24H 器件
- 寄存器 2-5，适用于 PIC24E 器件
- 寄存器 2-6，适用于 dsPIC30F 和 dsPIC33F 器件
- 寄存器 2-7，适用于 dsPIC33E 器件

2.17.1 dsPIC30F、dsPIC33F 和 dsPIC33E 的特定位

除了设置 CPU 模式之外，还可以通过 CORCON 寄存器使用以下功能：

- 使能 ACCA 和 ACCB 的饱和功能
- 设定数据空间写饱和模式
- 设定累加器饱和和舍入模式
- 设定 DSP 操作的乘法器模式
- 提前终止 DO 循环
- 提供 DO 循环嵌套层级（DL<2:0>）的状态信息
- 选择固定或可变中断延时（仅限 dsPIC33E）

2.17.2 PIC24E 和 dsPIC33E 的特定位

提供了指示堆栈帧是否激活的状态位（SFA）。

注： PIC24E 和 dsPIC33E 器件不具有 PSV 控制位，它为 SFA 位所替代。

2.18 影子寄存器

影子寄存器用作暂存寄存器。在某些事件发生时，影子寄存器与其相关主寄存器之间能够实现存储内容的传送。编程模型中的一些寄存器具有影子寄存器。在 DO、POP.S 或 PUSH.S 指令的执行过程中，将会使用到这些影子寄存器。表 2-2 给出了影子寄存器的使用。

注： DO 指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

表 2-2: 自动影子寄存器使用

| 存储单元 | DO ⁽¹⁾ | POP.S/PUSH.S |
|--------------------------|-------------------|--------------|
| DCOUNT ⁽¹⁾ | 使用 | — |
| DOSTART ⁽¹⁾ | 使用 | — |
| DOEND ⁽¹⁾ | 使用 | — |
| 状态寄存器 —— DC、N、OV、Z 和 C 位 | — | 使用 |
| W0-W3 | — | 使用 |

注 1： DO 指令使用的影子寄存器仅在 dsPIC30F 和 dsPIC33F 器件中可用。

对于 dsPIC30F 和 dsPIC33F 器件，由于 DCOUNT、DOSTART 和 DOEND 寄存器都具有影子寄存器，因此可在没有额外开销的情况下实现嵌套 DO 循环的功能。所有影子寄存器的深度皆为一个寄存器，因此可实现一层 DO 循环嵌套。在 CORCON 寄存器中 DO 循环嵌套层级状态位 DL<2:0>（CORCON<10:8>）的支持下，在程序中可实现更多层 DO 循环嵌套。

注： 所有影子寄存器的深度皆为一个寄存器，且不能对其进行直接访问。更多层影子寄存器深度可通过软件堆栈用软件实现。

2.19 DO 堆栈（dsPIC33E 器件）

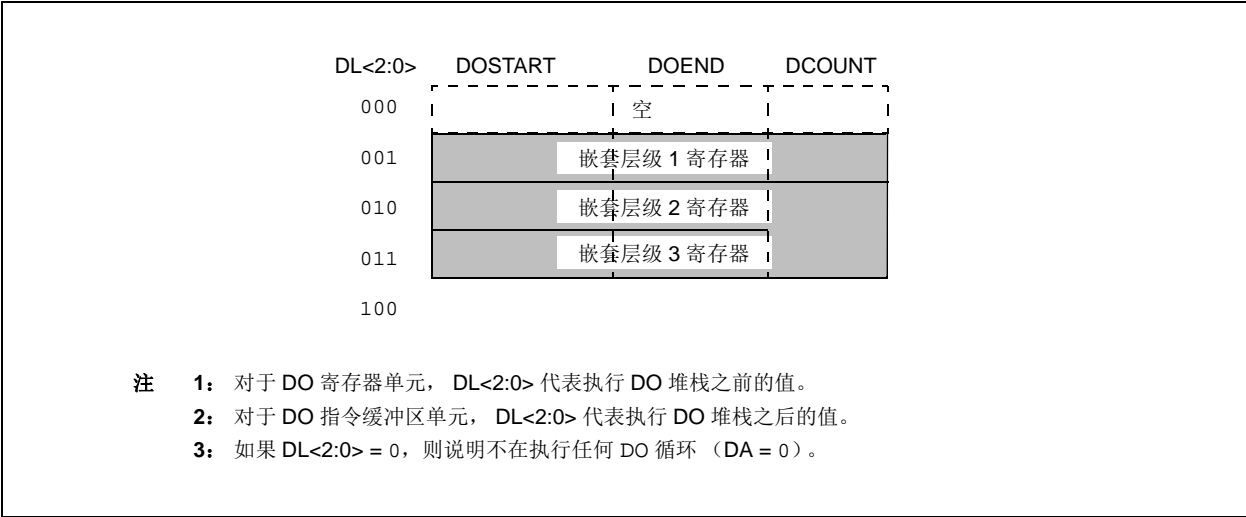
DO堆栈用于在遇到另一个DO循环（即，嵌套的DO循环）时，保存与当前DO循环关联的以下元素。

- DOSTART 寄存器的值
- DOEND 寄存器的值
- DCOUNT 寄存器的值

请注意，DO 嵌套层级状态位域（DL<2:0>）还用作一个用于寻址 DO 堆栈的指针。执行 DO 指令之后，DO 嵌套层级状态位域（DL<2:0>）会指向下一个可用堆栈单元。

DOSTART、DOEND 和 DCOUNT 寄存器各自具有关联的硬件堆栈，使 DO 循环硬件可以最多支持三层嵌套。图 2-5 给出了 DO 循环堆栈的概念表示图。

图 2-5: DO 循环堆栈概念图



16 位 MCU 和 DSC 程序员参考手册

寄存器 2-1: SR: CPU 状态寄存器 (PIC24H、PIC24F 和 PIC24E 器件)

| | | | | | | | |
|--------|-----|-----|-----|-----|-----|-----|-------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
| — | — | — | — | — | — | — | DC |
| bit 15 | | | | | | | bit 8 |

| | | | | | | | |
|-----------------------|-----------------------|-----------------------|-----|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| IPL2 ^(1,2) | IPL1 ^(1,2) | IPL0 ^(1,2) | RA | N | OV | Z | C |
| bit 7 | | | | | | | bit 0 |

图注:

R = 可读位

W = 可写位

U = 未实现位, 读为 0

C = 可清零位

-n = POR 时的值

1 = 置 1

0 = 清零

x = 未知

bit 15-9 **未实现:** 读为 0

bit 8 **DC:** MCU ALU 半进位 / 借位位

1 = 结果的第 4 个低位 (对于字节大小的数据) 或第 8 个低位 (对于字大小的数据) 发生进位

0 = 结果的第 4 个低位 (对于字节大小的数据) 或第 8 个低位 (对于字大小的数据) 未发生进位

bit 7-5 **IPL<2:0>:** CPU 中断优先级状态位 ^(1,2)

111 = CPU 中断优先级是 7 (15)。禁止用户中断

110 = CPU 中断优先级是 6 (14)

101 = CPU 中断优先级是 5 (13)

100 = CPU 中断优先级是 4 (12)

011 = CPU 中断优先级是 3 (11)

010 = CPU 中断优先级是 2 (10)

001 = CPU 中断优先级是 1 (9)

000 = CPU 中断优先级是 0 (8)

bit 4 **RA:** REPEAT 循环激活位

1 = REPEAT 循环在进行

0 = REPEAT 循环不在进行

bit 3 **N:** MCU ALU 负位

1 = 结果为负

0 = 结果非负 (零或正值)

bit 2 **OV:** MCU ALU 溢出位

该位用于有符号的算术运算 (以二进制补码方式进行)。它表示量值上的溢出, 这种溢出将导致符号位改变状态。

1 = 有符号算术运算中发生溢出 (本次算术运算)

0 = 未发生溢出

bit 1 **Z:** MCU ALU 零位

1 = 影响 Z 位的任何运算在过去某时已将该位置 1

0 = 影响 Z 位的最近一次运算已将该位清零 (即运算结果非零)

bit 0 **C:** MCU ALU 进位 / 借位位

1 = 最高位发生进位

0 = 最高位未发生进位

注 1: IPL<2:0> 位与 IPL3 位 (CORCON<3>) 组合形成 CPU 中断优先级。如果 IPL3 = 1, 则括号中的值即指 IPL。当 IPL<3> = 1 时, 禁止用户中断。

2: 当 NSTDIS (INTCON1<15>) = 1 时, IPL<2:0> 状态位是只读位。请参见具体器件系列的系列参考手册, 以查看相关的中断寄存器。

寄存器 2-2: SR: CPU 状态寄存器 (dsPIC30F 和 dsPIC33F 器件)

| | | | | | | | |
|---------------------|---------------------|---------------------|---------------------|-------|------------------------|-------------------|-------|
| R-0 | R-0 | R/C-0 | R/C-0 | R-0 | R/C-0 | R-0 | R/W-0 |
| OA | OB | SA ^(1,2) | SB ^(1,2) | OAB | SAB ^(1,2,3) | DA ⁽⁴⁾ | DC |
| bit 15 | | | | | | | bit 8 |
| R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| IPL2 ⁽⁵⁾ | IPL1 ⁽⁵⁾ | IPL0 ⁽⁵⁾ | RA | N | OV | Z | C |
| bit 7 | | | | | | | bit 0 |

| | | | |
|--------------|---------|----------|--------|
| 图注: | | | |
| R = 可读位 | W = 可写位 | C = 可清零位 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

| | |
|---------|---|
| bit 15 | OA: 累加器 A 溢出位 1 = 累加器 A 溢出 0 = 累加器 A 未溢出 |
| bit 14 | OB: 累加器 B 溢出位 1 = 累加器 B 溢出 0 = 累加器 B 未溢出 |
| bit 13 | SA: 累加器 A 饱和位 ^(1, 2) 1 = 累加器 A 饱和或自该位上一次被清零以来已处于饱和 0 = 累加器 A 未饱和 |
| bit 12 | SB: 累加器 B 饱和位 ^(1, 2) 1 = 累加器 B 饱和或自该位上一次被清零以来已处于饱和 0 = 累加器 B 未饱和 |
| bit 11 | OAB: OA OB 联合累加器溢出位 1 = 累加器 A 或 B 溢出 0 = 累加器 A 或 B 皆未溢出 |
| bit 10 | SAB: SA SB 联合累加器饱和位 ^(1, 2, 3) 1 = 累加器 A 或 B 饱和或自该位上一次被清零以来已处于饱和 0 = 累加器 A 或 B 皆未饱和 |
| bit 9 | DA: DO 循环激活位 ⁽⁴⁾ 1 = DO 循环在进行 0 = DO 循环不在进行 |
| bit 8 | DC: MCU ALU 半进位位 1 = 低半字节最高位发生进位 0 = 低半字节最高位未发生进位 |
| bit 7-5 | IPL<2:0>: 中断优先级位 ⁽⁵⁾ 111 = CPU 中断优先级是 7 (15)。禁止用户中断 110 = CPU 中断优先级是 6 (14) 101 = CPU 中断优先级是 5 (13) 100 = CPU 中断优先级是 4 (12) 011 = CPU 中断优先级是 3 (11) 010 = CPU 中断优先级是 2 (10) 001 = CPU 中断优先级是 1 (9) 000 = CPU 中断优先级是 0 (8) |

- 注 1: 可对该位进行读或清零操作，但不能将其置 1。
- 2: 一旦该位置 1，必须由软件手动清零。
- 3: 对该位进行清零将导致 SA 和 SB 清零。
- 4: 该位是只读位。
- 5: IPL<2:0>位与 IPL3 位 (CORCON<3>) 组合形成 CPU 中断优先级。如果 IPL3 = 1，则括号中的值即指 IPL。

寄存器 2-2: **SR: CPU 状态寄存器** (dsPIC30F 和 dsPIC33F 器件) (续)

| | |
|-------|--|
| bit 4 | RA: REPEAT 循环激活位 1 = REPEAT 循环在进行 0 = REPEAT 循环不在进行 |
| bit 3 | N: MCU ALU 负位 1 = 操作结果为负 0 = 操作结果非负 |
| bit 2 | OV: MCU ALU 溢出位 1 = 发生溢出 0 = 未发生溢出 |
| bit 1 | Z: MCU ALU 零位 1 = 操作结果为零 0 = 操作结果非零 |
| bit 0 | C: MCU ALU 进位 / 借位位 1 = 最高位发生进位 0 = 最高位未发生进位 |

- 注
- 1: 可对该位进行读或清零操作, 但不能将其置 1。
 - 2: 一旦该位置 1, 必须由软件手动清零。
 - 3: 对该位进行清零将导致 SA 和 SB 清零。
 - 4: 该位是只读位。
 - 5: IPL<2:0> 位与 IPL3 位 (CORCON<3>) 组合形成 CPU 中断优先级。如果 IPL3 = 1, 则括号中的值即指 IPL。

寄存器 2-3: SR: CPU 状态寄存器 (dsPIC33E 器件)

| | | | | | | | |
|-----------------------|-----------------------|-----------------------|-------------------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/C-0 | R/C-0 | R -0 | R/W-0 |
| OA | OB | SA ⁽³⁾ | SB ⁽³⁾ | OAB | SAB | DA | DC |
| bit 15 | | | | | | bit 8 | |
| R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| IPL2 ^(1,2) | IPL1 ^(1,2) | IPL0 ^(1,2) | RA | N | OV | Z | C |
| bit 7 | | | | | | bit 0 | |

| | | | | | | | |
|--------------|--|---------|--|----------------|--|--------|--|
| 图注: | | | | U = 未实现位, 读为 0 | | | |
| R = 可读位 | | W = 可写位 | | C = 可清零位 | | | |
| -n = POR 时的值 | | 1 = 置 1 | | 0 = 清零 | | x = 未知 | |

- bit 15
- OA:** 累加器 A 溢出状态位
1 = 累加器 A 溢出
0 = 累加器 A 未溢出
- bit 14
- OB:** 累加器 B 溢出状态位
1 = 累加器 B 溢出
0 = 累加器 B 未溢出
- bit 13
- SA:** 累加器 A 饱和状态位
1 = 累加器 A 饱和或自该位上一次被清零以来已处于饱和
0 = 累加器 A 未饱和
- bit 12
- SB:** 累加器 B 饱和状态位
1 = 累加器 B 饱和或自该位上一次被清零以来已处于饱和
0 = 累加器 B 未饱和
- bit 11
- OAB:** OA || OB 联合累加器溢出状态位
1 = 累加器 A 或 B 溢出
0 = 累加器 A 或 B 皆未溢出
- bit 10
- SAB:** SA || SB 联合累加器饱和状态位
1 = 累加器 A 或 B 饱和或自该位上一次被清零以来已处于饱和
0 = 累加器 A 或 B 皆未饱和
- bit 9
- DA:** DO 循环激活位
1 = DO 循环在进行
0 = DO 循环不在进行
- bit 8
- DC:** MCU ALU 半进位 / 借位位
1 = 结果的第 4 个低位 (对于字节大小的数据) 或第 8 个低位 (对于字大小的数据) 发生进位
0 = 结果的第 4 个低位 (对于字节大小的数据) 或第 8 个低位 (对于字大小的数据) 未发生进位
- 注
- 1: IPL<2:0> 位与 IPL3 位 (CORCON<3>) 组合形成 CPU 中断优先级。如果 IPL3 = 1, 则括号中的值即指 IPL。当 IPL3 = 1 时, 禁止用户中断。

2: 当 NSTDIS (INTCON1<15>) = 1 时, IPL<2:0> 状态位是只读位。请参见具体器件系列的系列参考手册, 以查看相关的中断寄存器。

3: 对 SR 的数据写操作可以修改 SA 或 SB 位, 方法是向 SA 和 SB 写入数据或清零 SAB 位。要避免可能出现的 SA/SB 位写竞争条件, 不要使用位操作来修改 SA 和 SB 位。

寄存器 2-3: SR: CPU 状态寄存器 (dsPIC33E 器件) (续)

| | |
|---------|--|
| bit 7-5 | IPL<2:0> : CPU 中断优先级状态位 ^(1,2) 111 = CPU 中断优先级是 7 (15)。禁止用户中断 110 = CPU 中断优先级是 6 (14) 101 = CPU 中断优先级是 5 (13) 100 = CPU 中断优先级是 4 (12) 011 = CPU 中断优先级是 3 (11) 010 = CPU 中断优先级是 2 (10) 001 = CPU 中断优先级是 1 (9) 000 = CPU 中断优先级是 0 (8) |
| bit 4 | RA : REPEAT 循环激活位 1 = REPEAT 循环在进行 0 = REPEAT 循环不在进行 |
| bit 3 | N : MCU ALU 负位 1 = 结果为负 0 = 结果非负 (零或正值) |
| bit 2 | OV : MCU ALU 溢出位 该位用于有符号的算术运算 (以二进制补码方式进行)。它表示量值上的溢出, 这种溢出将导致符号位改变状态。 1 = 有符号算术运算中发生溢出 (本次算术运算) 0 = 未发生溢出 |
| bit 1 | Z : MCU ALU 零位 1 = 操作结果为零 0 = 操作结果非零 |
| bit 0 | C : MCU ALU 进位 / 借位位 1 = 结果的最高位发生进位 0 = 结果的最高位未发生进位 |

- 注
- 1: IPL<2:0> 位与 IPL3 位 (CORCON<3>) 组合形成 CPU 中断优先级。如果 IPL3 = 1, 则括号中的值即指 IPL。当 IPL3 = 1 时, 禁止用户中断。
 - 2: 当 NSTDIS (INTCON1<15>) = 1 时, IPL<2:0> 状态位是只读位。请参见具体器件系列的系列参考手册, 以查看相关的中断寄存器。
 - 3: 对 SR 的数据写操作可以修改 SA 或 SB 位, 方法是向 SA 和 SB 写入数据或清零 SAB 位。要避免可能出现的 SA/SB 位写竞争条件, 不要使用位操作来修改 SA 和 SB 位。

寄存器 2-4: **CORCON: 内核控制寄存器 (PIC24F 和 PIC24H 器件)**

| | | | | | | | |
|----------------|-----|----------|-----|-----------------------|-------|---------|-------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |
| | | | | | | | |
| U-0 | U-0 | U-0 | U-0 | R/C-0 | R/W-0 | U-0 | U-0 |
| — | — | — | — | IPL3 ^(1,2) | PSV | — | — |
| bit 7 | | | | | | | bit 0 |
| | | | | | | | |
| 图注: | | C = 可清零位 | | R = 可读位 | | W = 可写位 | |
| -n = POR 时的值 | | 1 = 置 1 | | 0 = 清零 | | x = 未知 | |
| U = 未实现位, 读为 0 | | | | | | | |

- bit 15-4未实现: 读为 0
- bit 3

IPL3: 中断优先级 3 状态位 ^(1,2)

1 = CPU 中断优先级为 8 或更高 (激活陷阱异常)

0 = CPU 中断优先级为 7 或更低 (未激活陷阱异常)
- bit 2

PSV: 数据空间中的 PSV 使能位

1 = 使能数据空间中的 PSV

0 = 禁止数据空间中的 PSV
- bit 1-0未实现: 读为 0

- 注 1: 可对该位进行读或清零操作, 但不能将其置 1。
- 2: 该位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

16 位 MCU 和 DSC 程序员参考手册

寄存器 2-5: CORCON: 内核控制寄存器 (PIC24E 器件)

| | | | | | | | |
|--------|-----|-----|-----|-----------------------|-----|-----|-------|
| R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| VAR | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |
| U-0 | U-0 | U-0 | U-0 | R/C-0 | R-0 | U-0 | U-0 |
| — | — | — | — | IPL3 ^(1,2) | SFA | — | — |
| bit 7 | | | | | | | bit 0 |

| | | | |
|--------------|---------|----------------|--------|
| 图注: | | | |
| R = 可读位 | W = 可写位 | U = 未实现位, 读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

| | |
|----------|---|
| bit 15 | VAR: 可变异常处理延时控制位 1 = 可变 (有界确定性) 异常处理延时 0 = 固定 (完全确定性) 异常处理延时 |
| bit 14-4 | 未实现: 读为 0 |
| bit 3 | IPL3: CPU 中断优先级状态位 3 ⁽¹⁾ 1 = CPU 中断优先级大于 7 0 = CPU 中断优先级等于或小于 7 |
| bit 2 | SFA: 堆栈帧激活状态位 1 = 堆栈帧激活。无论 DSRPAG 和 DSWPAG 值如何, W14 和 W15 都寻址 0x0000 至 0xFFFF。 0 = 堆栈帧未激活。W14 和 W15 寻址 EDS 或基本数据空间 |
| bit 1-0 | 未实现: 读为 0 |

- 注 1: 可对该位进行读或清零操作, 但不能将其置 1。
- 2: IPL3 位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

寄存器 2-6: CORCON: 内核控制寄存器 (dsPIC30F 和 dsPIC33F 器件)

| | | | | | | | |
|--------|-----|-----|-------|--------------------|--------------------------|-----|-----|
| U-0 | U-0 | U-0 | R/W-0 | R(0)/W-0 | R-0 | R-0 | R-0 |
| — | — | — | US | EDT ⁽¹⁾ | DL<2:0> ^(2,3) | | |
| bit 15 | | | | | bit 8 | | |

| | | | | | | | |
|-------|-------|-------|--------|-----------------------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/C-0 | R/W-0 | R/W-0 | R/W-0 |
| SATA | SATB | SATDW | ACCSAT | IPL3 ^(4,5) | PSV | RND | IF |
| bit 7 | | | | | | | |
| | | | bit 0 | | | | |

| | | | |
|----------------|----------|---------|---------|
| 图注: | C = 可清零位 | R = 可读位 | W = 可写位 |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |
| U = 未实现位, 读为 0 | | | |

bit 15-13 未实现: 读为 0

bit 12 **US**: 无符号或有符号乘法器模式选择位
 1 = 使能 DSP 乘法运算为无符号模式
 0 = 使能 DSP 乘法运算为有符号模式

bit 11 **EDT**: 提前终止 DO 循环控制位 ⁽¹⁾
 1 = 在当前迭代结束时终止执行 DO 循环
 0 = 无影响

bit 10-8 **DL<2:0>**: DO 循环嵌套层级状态位 ^(2,3)
 111 = DO 循环位于 7 层嵌套
 110 = DO 循环位于 6 层嵌套
 110 = DO 循环位于 5 层嵌套
 110 = DO 循环位于 4 层嵌套
 011 = DO 循环位于 3 层嵌套
 010 = DO 循环位于 2 层嵌套
 001 = DO 循环处于激活状态, 但并未嵌套 (仅是 1 层)
 000 = DO 循环并未激活

bit 7 **SATA**: ACCA 饱和使能位
 1 = 使能累加器 A 饱和
 0 = 禁止累加器 A 饱和

bit 6 **SATB**: ACCB 饱和使能位
 1 = 使能累加器 B 饱和
 0 = 禁止累加器 B 饱和

bit 5 **SATDW**: DSP 引擎对数据空间写饱和使能位
 1 = 使能数据空间写饱和
 0 = 禁止数据空间写饱和

bit 4 **ACCSAT**: 累加器饱和模式选择位
 1 = 9.31 饱和 (超饱和)
 0 = 1.31 饱和 (普通饱和)

bit 3 **IPL3**: 中断优先级 3 状态位 ^(4,5)
 1 = CPU 中断优先级为 8 或更高 (激活陷阱异常)
 0 = CPU 中断优先级为 7 或更低 (未激活陷阱异常)

bit 2 **PSV**: 数据空间中的程序空间可视性使能位
 1 = 数据空间中程序空间可视
 0 = 数据空间中程序空间不可视

注 1: 该位总是读为 0。

2: DL<2:1> 为只读位。

3: DO 循环嵌套的前两层由硬件处理。

4: 可对该位进行读或清零操作, 但不能将其置 1。

5: 该位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

寄存器 2-6: **CORCON:** 内核控制寄存器 (dsPIC30F 和 dsPIC33F 器件) (续)

bit 1 **RND:** 舍入模式选择位
 1 = 使能有偏 (常规) 舍入
 0 = 使能无偏 (收敛) 舍入

bit 0 **IF:** 整数或小数乘法器模式选择位
 1 = 使能整数模式的 DSP 乘法运算
 0 = 使能小数模式的 DSP 乘法运算

- 注 1: 该位总是读为 0。
- 2: DL<2:1> 为只读位。
- 3: DO 循环嵌套的前两层由硬件处理。
- 4: 可对该位进行读或清零操作, 但不能将其置 1。
- 5: 该位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

寄存器 2-7: CORCON: 内核控制寄存器 (dsPIC33E 器件)

| | | | | | | | | |
|--------|-------|---------|--------|-----------------------|---------|-------|-------|-------|
| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-0 | |
| VAR | — | US<1:0> | | EDT ⁽¹⁾ | DL<2:0> | | | |
| bit 15 | | | | | | | | bit 8 |
| | | | | | | | | |
| R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/C-0 | R-0 | R/W-0 | R/W-0 | |
| SATA | SATB | SATDW | ACCSAT | IPL3 ^(2,3) | SFA | RND | IF | |
| bit 7 | | | | | | | | bit 0 |

| | | | |
|--------------|---------|----------------|--------|
| 图注: | | | |
| R = 可读位 | W = 可写位 | U = 未实现位, 读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

- bit 15 **VAR:** 可变异常处理延时控制位
1 = 可变 (有界确定性) 异常处理延时
0 = 固定 (完全确定性) 异常处理延时
- bit 14 **未实现:** 读为 0
- bit 13-12 **US<1:0>:** DSP 乘法无符号 / 有符号控制位
11 = 保留
10 = DSP 引擎执行混合符号乘法运算
01 = DSP 引擎执行无符号乘法运算
00 = DSP 引擎执行有符号乘法运算
- bit 11 **EDT:** 提前终止 DO 循环控制位 ⁽¹⁾
1 = 在当前循环迭代结束时终止执行 DO 循环
0 = 无影响
- bit 10-8 **DL<2:0>:** DO 循环嵌套层级状态位
111 = 正在进行 7 层 DO 循环嵌套
•
•
•
001 = 正在进行 1 层 DO 循环嵌套
000 = 正在进行 0 层 DO 循环嵌套
- bit 7 **SATA:** ACCA 饱和和使能位
1 = 使能累加器 A 饱和
0 = 禁止累加器 A 饱和
- bit 6 **SATB:** ACCB 饱和和使能位
1 = 使能累加器 B 饱和
0 = 禁止累加器 B 饱和
- bit 5 **SATDW:** DSP 引擎对数据空间写饱和和使能位
1 = 使能数据空间写饱和
0 = 禁止数据空间写饱和
- bit 4 **ACCSAT:** 累加器饱和模式选择位
1 = 9.31 饱和 (超饱和)
0 = 1.31 饱和 (普通饱和)
- bit 3 **IPL3:** CPU 中断优先级状态位 3⁽²⁾
1 = CPU 中断优先级大于 7
0 = CPU 中断优先级等于或小于 7

注 1: 该位总是读为 0。
 2: 可对该位进行读或清零操作, 但不能将其置 1。
 3: IPL3 位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

寄存器 2-7: CORCON: 内核控制寄存器 (dsPIC33E 器件) (续)

- bit 2 **SFA:** 堆栈帧激活状态位
1 = 堆栈帧激活。无论 DSRPAG 和 DSWPAG 值如何, W14 和 W15 都寻址 0x0000 至 0xFFFF。
0 = 堆栈帧未激活。W14 和 W15 寻址 EDS 或基本数据空间
- bit 1 **RND:** 舍入模式选择位
1 = 使能有偏 (常规) 舍入
0 = 使能无偏 (收敛) 舍入
- bit 0 **IF:** 整数或小数乘法器模式选择位
1 = 使能整数模式的 DSP 乘法运算
0 = 使能小数模式的 DSP 乘法运算

- 注** **1:** 该位总是读为 0。
2: 可对该位进行读或清零操作, 但不能将其置 1。
3: IPL3 位与 IPL<2:0> 位 (SR<7:5>) 组合形成 CPU 中断优先级。

第 3 章 指令集概述

目录

本章包括以下主要内容：

| | | |
|-----|--------------|----|
| 3.1 | 简介 | 38 |
| 3.2 | 指令集概述 | 38 |
| 3.3 | 指令集汇总表 | 40 |

3.1 简介

16 位 MCU 和 DSC 指令集提供了一整套支持传统单片机应用的指令，以及一类支持数学密集型应用的指令。由于保留了几乎所有 8 位 PIC MCU 指令集的功能，该混合指令集使得已熟悉 PIC 单片机的用户可以很容易地移植到 16 位器件。

3.2 指令集概述

根据不同的器件系列，16 位 MCU 和 DSC 指令集最多包含 84 条指令，如表 3-1 所示可分为不同的功能组。表 1-2 对表 3-2 至表 3-11 指令汇总表中使用的符号进行了定义。这些表对每一条指令的语法、说明、存储以及执行要求进行了定义。存储要求以 24 位指令字的方式来表征，而执行要求以指令周期的方式来表征。

表 3-1: 指令组

| 功能组 | 汇总表 | 页码 |
|-------------------|--------|----|
| 传送指令 | 表 3-2 | 40 |
| 数学指令 | 表 3-3 | 41 |
| 逻辑指令 | 表 3-4 | 43 |
| 循环 / 移位指令 | 表 3-5 | 44 |
| 位操作指令 | 表 3-6 | 45 |
| 比较 / 跳过和比较 / 转移指令 | 表 3-7 | 46 |
| 程序流指令 | 表 3-8 | 47 |
| 影子 / 堆栈指令 | 表 3-9 | 49 |
| 控制指令 | 表 3-10 | 49 |
| DSP 类指令 (1) | 表 3-11 | 50 |

注 1: DSP 类指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件系列中可用。

多数指令具有多种不同的寻址模式以及执行流，这需要不同的指令形式。例如，根据不同的器件系列，最多有 6 种不同的 ADD 指令，而每一种指令形式都具有其自身的指令编码。第 5 章“指令描述”提供了指令格式的说明以及特定指令的操作。除此之外，第 7 章“参考信息”给出了一个以阿拉伯字母顺序排列的指令集总表。

3.2.1 多周期指令

如指令汇总表所示，大多数指令的执行时间为一个指令周期，但存在以下例外：

- 注：** DO 和 DIVF 指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件系列中可用。
- 指令 DO、MOV.D、POP.D、PUSH.D、TBLRDH、TBLRDL、TBLWTH 和 TBLWTL 的执行时间为 2 个指令周期
 - 指令 DIV.S、DIV.U 和 DIVF 为单周期指令，应该作为 REPEAT 指令的目标被连续执行 18 次
 - 改变程序计数器的指令执行时间同样需要 2 个指令周期，但需要额外的一个指令周期来执行 NOP 指令。对于比较 - 跳过指令，当跳过双字指令时需要 3 个指令周期来执行，其中 2 个周期执行 NOP 指令。对于比较 - 转移指令（仅限 dsPIC33E/PIC24E 器件），当发生转移时需要 5 个指令周期来执行
 - RETFIE、RETLW 和 RETURN 是改变程序计数器的特殊指令。这些指令的执行时间为 3 个指令周期。如果有异常等待处理，将执行 2 个周期。

- 注 1：** 对于 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件，通过使用程序空间可视性（PSV）而将程序存储器作为数据访问的指令会产生一或两个周期的延时；而在 dsPIC33E 和 PIC24E 器件中，根据闪存访问时间，使用 PSV 会产生 4 个周期的延时。但是，无论使用何种器件，当 REPEAT 循环的目标指令将程序存储器作为数据访问时，只有第一次执行目标指令时才会产生延时。有关详细信息，请参见具体器件的系列参考手册。
- 2：** 在一些器件系列中，根据闪存访问时间，所有指令都可能会产生额外的延时。例如，PIC24E 和 dsPIC33E 器件的闪存访问时间为 3 个周期。但是，指令流水线可以提高实际的指令执行吞吐率。有关指令时序的详细信息，请参见具体器件系列参考手册的**第 2 章“CPU”**。
- 3：** 在 PIC24E 和 dsPIC33E 器件中，对于非 CPU 特殊功能寄存器（例如，I/O 端口、外设控制或状态寄存器以及中断标志等）的所有读操作和读 - 修改 - 写操作（包括位操作）都需要 2 个指令周期来执行。但是，对 CPU 和非 CPU 特殊功能寄存器的所有写操作以及对 CPU 特殊功能寄存器的所有读操作和读 - 修改 - 写操作，都需要 1 个指令周期。

3.2.2 多字指令

如表 3-2 所定义，几乎所有的指令都占用一个指令字（24 位），但表 3-8 中所列的程序流指令 CALL、DO 和 GOTO 例外。因为这些指令的操作码中包含较大的立即数操作数，因此这些指令需要两个字的存储空间。

3.3 指令集汇总表

表 3-2: 传送指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|--------------------------------------|---------------------------------|----|------------------|---------------------|
| EXCH Wns, Wnd | 交换 Wns 和 Wnd 的内容 | 1 | 1 | 243 |
| MOV f {, WREG} ⁽¹⁾ | 将 f 中内容传送至目标寄存器 | 1 | 1 | 279 |
| MOV WREG, f | 将 WREG 中内容传送至 f | 1 | 1 | 280 |
| MOV f, Wnd | 将 f 中内容传送至 Wnd | 1 | 1 ⁽⁴⁾ | 281 |
| MOV Wns, f | 将 Wns 中内容传送至 f | 1 | 1 | 282 |
| MOV.B #lit8, Wnd | 将 8 位立即数传送至 Wnd | 1 | 1 | 283 |
| MOV #lit16, Wnd | 将 16 位立即数传送至 Wnd | 1 | 1 | 284 |
| MOV [Ws+Slit10], Wnd | 将 [Ws + 有符号 10 位偏移量] 中内容传送至 Wnd | 1 | 1 ⁽⁴⁾ | 285 |
| MOV Wns, [Wd+Slit10] | 将 Wns 中内容传送至 [Wd + 有符号 10 位偏移量] | 1 | 1 | 286 |
| MOV Wso, Wdo | 将 Wso 中内容传送至 Wdo | 1 | 1 ⁽⁴⁾ | 287 |
| MOV.D Ws, Wnd | 双字传送 Ws 内容至 Wnd:Wnd + 1 | 1 | 2 ⁽⁴⁾ | 289 |
| MOV.D Wns, Wd | 双字传送 Wns:Wns + 1 内容至 Wd | 1 | 2 | 289 |
| MOVPAG #lit10, DSRPAG ⁽²⁾ | 将 10 位立即数传送至 DSRPAG | 1 | 1 | 291 |
| MOVPAG #lit9, DSWPAG ⁽²⁾ | 将 9 位立即数传送至 DSWPAG | 1 | 1 | 291 |
| MOVPAG #lit8, TBLPAG ⁽²⁾ | 将 8 位立即数传送至 TBLPAG | 1 | 1 | 291 |
| MOVPAG Wn, DSRPAG ⁽²⁾ | 将 Wn 中内容传送到 DSRPAG | 1 | 1 | 292 |
| MOVPAG Wn, DSWPAG ⁽²⁾ | 将 Wn 中内容传送到 DSWPAG | 1 | 1 | 292 |
| MOVPAG Wn, TBLPAG ⁽²⁾ | 将 Wn 中内容传送到 TBLPAG | 1 | 1 | 292 |
| SWAP Wn | Wn = Wn 内容字节或半字节交换 | 1 | 1 | 426 |
| TBLRDH [Ws], Wd | 将高位程序字读入 Wd | 1 | 2 ⁽³⁾ | 427 |
| TBLRDL [Ws], Wd | 将低位程序字读入 Wd | 1 | 2 ⁽³⁾ | 429 |
| TBLWTH Ws, [Wd] | 将 Ws 中的内容写入高位程序字 | 1 | 2 ⁽⁴⁾ | 431 |
| TBLWTL Ws, [Wd] | 将 Ws 中的内容写入低位程序字 | 1 | 2 ⁽⁴⁾ | 433 |

- 注 1: 当指定了可选的 {, WREG} 操作数时, 指令的目标寄存器为 WREG。当未指定 {, WREG} 时, 指令的目标寄存器为文件寄存器 f。
- 2: MOVPAG 指令仅在 dsPIC33E 和 PIC24E 器件中可用。
- 3: 在 dsPIC33E 和 PIC24E 器件中, 这些指令需要 3 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。
- 4: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

表 3-3: 数学指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|--------------------------------|---|----|-------------------|-----|
| ADD $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f + WREG$ | 1 | 1 ⁽⁵⁾ | 99 |
| ADD $\#lit10, Wn$ | $Wn = lit10 + Wn$ | 1 | 1 | 100 |
| ADD $Wb, \#lit5, Wd$ | $Wd = Wb + lit5$ | 1 | 1 | 101 |
| ADD Wb, Ws, Wd | $Wd = Wb + Ws$ | 1 | 1 ⁽⁵⁾ | 102 |
| ADDC $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f + WREG + (C)$ | 1 | 1 ⁽⁵⁾ | 106 |
| ADDC $\#lit10, Wn$ | $Wn = lit10 + Wn + (C)$ | 1 | 1 | 107 |
| ADDC $Wb, \#lit5, Wd$ | $Wd = Wb + lit5 + (C)$ | 1 | 1 | 108 |
| ADDC Wb, Ws, Wd | $Wd = Wb + Ws + (C)$ | 1 | 1 ⁽⁵⁾ | 110 |
| DAW.B Wn | Wn = 对 Wn 的内容进行十进制调整 | 1 | 1 | 216 |
| DEC $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f - 1$ | 1 | 1 ⁽⁵⁾ | 217 |
| DEC Ws, Wd | $Wd = Ws - 1$ | 1 | 1 ⁽⁵⁾ | 218 |
| DEC2 $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f - 2$ | 1 | 1 ⁽⁵⁾ | 220 |
| DEC2 Ws, Wd | $Wd = Ws - 2$ | 1 | 1 ⁽⁵⁾ | 221 |
| DIV.S Wm, Wn | 有符号 16/16 位整数除法, $Q \rightarrow W0, R \rightarrow W1$ | 1 | 18 ⁽²⁾ | 224 |
| DIV.SD Wm, Wn | 有符号 32/16 位整数除法, $Q \rightarrow W0, R \rightarrow W1$ | 1 | 18 ⁽²⁾ | 224 |
| DIV.U Wm, Wn | 无符号 16/16 位整数除法, $Q \rightarrow W0, R \rightarrow W1$ | 1 | 18 ⁽²⁾ | 226 |
| DIV.UD Wm, Wn | 无符号 32/16 位整数除法, $Q \rightarrow W0, R \rightarrow W1$ | 1 | 18 ⁽²⁾ | 226 |
| DIVF Wm, Wn | 有符号 16/16 位小数除法, $Q \rightarrow W0, R \rightarrow W1$ | 1 | 18 ⁽²⁾ | 228 |
| INC $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f + 1$ | 1 | 1 ⁽⁵⁾ | 254 |
| INC Ws, Wd | $Wd = Ws + 1$ | 1 | 1 ⁽⁵⁾ | 255 |
| INC2 $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f + 2$ | 1 | 1 ⁽⁵⁾ | 257 |
| INC2 Ws, Wd | $Wd = Ws + 2$ | 1 | 1 ⁽⁵⁾ | 258 |
| MUL f | $W3:W2 = f * WREG$ | 1 | 1 ⁽⁵⁾ | 303 |
| MUL.SS Wb, Ws, Wnd | $\{Wnd + 1, Wnd\} =$ 有符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 305 |
| MUL.SS $Wb, Ws, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 307 |
| MUL.SU $Wb, \#lit5, Wnd$ | $\{Wnd + 1, Wnd\} =$ 有符号 (Wb) * 无符号 ($lit5$) | 1 | 1 | 308 |
| MUL.SU Wb, Ws, Wnd | $\{Wnd + 1, Wnd\} =$ 有符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 310 |
| MUL.SU $Wb, Ws, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 312 |
| MUL.SU $Wb, \#lit5, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 无符号 ($lit5$) | 1 | 1 | 314 |
| MUL.US Wb, Ws, Wnd | $\{Wnd + 1, Wnd\} =$ 无符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 315 |
| MUL.US $Wb, Ws, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 317 |
| MUL.UU $Wb, \#lit5, Wnd$ | $\{Wnd + 1, Wnd\} =$ 无符号 (Wb) * 无符号 ($lit5$) | 1 | 1 | 319 |
| MUL.UU Wb, Ws, Wnd | $\{Wnd + 1, Wnd\} =$ 无符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 320 |
| MUL.UU $Wb, Ws, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 322 |
| MUL.UU $Wb, \#lit5, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 无符号 ($lit5$) | 1 | 1 | 323 |
| MULW.SS $Wb, Ws, Wnd^{(3)}$ | $Wnd =$ 有符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 324 |

- 注 1: 当指定了可选的 $\{, WREG\}$ 操作数时, 指令的目标寄存器为 $WREG$ 。当未指定 $\{, WREG\}$ 时, 指令的目标寄存器为文件寄存器 f 。
- 2: 除法指令前必须加上一条 REPEAT #17 指令, 这样除法指令才能连续执行 18 次。
- 3: 这些指令仅在 dsPIC33E 和 PIC24E 器件中可用。
- 4: 这些指令仅在 dsPIC33E 器件中可用。
- 5: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

表 3-3: 数学指令 (续)

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|-------------------------------------|---------------------------------------|----|------------------|---------------------|
| MULW.SU Wb,Ws,Wnd ⁽³⁾ | Wnd = 有符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 326 |
| MULW.SU Wb,#lit5,Wnd ⁽³⁾ | Wnd = 有符号 (Wb) * 无符号 (lit5) | 1 | 1 | 328 |
| MULW.US Wb,Ws,Wnd ⁽³⁾ | Wnd = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 ⁽⁵⁾ | 329 |
| MULW.UU Wb,Ws,Wnd ⁽³⁾ | Wnd = 无符号 (Wb) * 无符号 (Ws) | 1 | 1 ⁽⁵⁾ | 331 |
| MULW.UU Wb,#lit5,Wnd ⁽³⁾ | Wnd = 无符号 (Wb) * 无符号 (lit5) | 1 | 1 | 332 |
| SE Ws,Wnd | Wnd = 符号扩展 Ws 内容 | 1 | 1 ⁽⁵⁾ | 393 |
| SUB f {,WREG} ⁽¹⁾ | 目标寄存器 = f - WREG | 1 | 1 ⁽⁵⁾ | 405 |
| SUB #lit10,Wn | Wn = Wn - lit10 | 1 | 1 | 406 |
| SUB Wb,#lit5,Wd | Wd = Wb - lit5 | 1 | 1 | 407 |
| SUB Wb,Ws,Wd | Wd = Wb - Ws | 1 | 1 ⁽⁵⁾ | 408 |
| SUBB f {,WREG} ⁽¹⁾ | 目标寄存器 = f - WREG - (\overline{C}) | 1 | 1 ⁽⁵⁾ | 411 |
| SUBB #lit10,Wn | Wn = Wn - lit10 - (\overline{C}) | 1 | 1 | 412 |
| SUBB Wb,#lit5,Wd | Wd = Wb - lit5 - (\overline{C}) | 1 | 1 | 413 |
| SUBB Wb,Ws,Wd | Wd = Wb - Ws - (\overline{C}) | 1 | 1 ⁽⁵⁾ | 415 |
| SUBBR f {,WREG} ⁽¹⁾ | 目标寄存器 = WREG - f - (\overline{C}) | 1 | 1 ⁽⁵⁾ | 417 |
| SUBBR Wb,#lit5,Wd | Wd = lit5 - Wb - (\overline{C}) | 1 | 1 | 418 |
| SUBBR Wb,Ws,Wd | Wd = Ws - Wb - (\overline{C}) | 1 | 1 ⁽⁵⁾ | 420 |
| SUBR f {,WREG} ⁽¹⁾ | 目标寄存器 = WREG - f | 1 | 1 ⁽⁵⁾ | 422 |
| SUBR Wb,#lit5,Wd | Wd = lit5 - Wb | 1 | 1 | 423 |
| SUBR Wb,Ws,Wd | Wd = Ws - Wb | 1 | 1 ⁽⁵⁾ | 424 |
| ZE Ws,Wnd | Wnd = 零扩展 Ws | 1 | 1 ⁽⁵⁾ | 442 |

- 注 1: 当指定了可选的 {,WREG} 操作数时, 指令的目标寄存器为 WREG。当未指定 {,WREG} 时, 指令的目标寄存器为文件寄存器 f。
- 2: 除法指令前必须加上一条 REPEAT #17 指令, 这样除法指令才能连续执行 18 次。
- 3: 这些指令仅在 dsPIC33E 和 PIC24E 器件中可用。
- 4: 这些指令仅在 dsPIC33E 器件中可用。
- 5: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

表 3-4: 逻辑指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|--------------------------|------------------------|----|------------------|-----|
| AND $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f .AND. WREG$ | 1 | 1 ⁽²⁾ | 112 |
| AND $\#lit10, Wn$ | $Wn = lit10 .AND. Wn$ | 1 | 1 | 113 |
| AND $Wb, \#lit5, Wd$ | $Wd = Wb .AND. lit5$ | 1 | 1 | 114 |
| AND Wb, Ws, Wd | $Wd = Wb .AND. Ws$ | 1 | 1 ⁽²⁾ | 115 |
| CLR f | $f = 0x0000$ | 1 | 1 | 184 |
| CLR $WREG$ | $WREG = 0x0000$ | 1 | 1 | 184 |
| CLR Wd | $Wd = 0x0000$ | 1 | 1 | 185 |
| COM $f \{, WREG\}^{(1)}$ | 目标寄存器 = \bar{f} | 1 | 1 ⁽²⁾ | 189 |
| COM Ws, Wd | $Wd = \bar{Ws}$ | 1 | 1 ⁽²⁾ | 190 |
| IOR $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f .IOR. WREG$ | 1 | 1 ⁽²⁾ | 260 |
| IOR $\#lit10, Wn$ | $Wn = lit10 .IOR. Wn$ | 1 | 1 | 261 |
| IOR $Wb, \#lit5, Wd$ | $Wd = Wb .IOR. lit5$ | 1 | 1 | 262 |
| IOR Wb, Ws, Wd | $Wd = Wb .IOR. Ws$ | 1 | 1 ⁽²⁾ | 263 |
| NEG $f \{, WREG\}^{(1)}$ | 目标寄存器 = $\bar{f} + 1$ | 1 | 1 ⁽²⁾ | 333 |
| NEG Ws, Wd | $Wd = \bar{Ws} + 1$ | 1 | 1 ⁽²⁾ | 334 |
| SETM f | $f = 0xFFFF$ | 1 | 1 | 395 |
| SETM $WREG$ | $WREG = 0xFFFF$ | 1 | 1 | 395 |
| SETM Wd | $Wd = 0xFFFF$ | 1 | 1 | 396 |
| XOR $f \{, WREG\}^{(1)}$ | 目标寄存器 = $f .XOR. WREG$ | 1 | 1 ⁽²⁾ | 437 |
| XOR $\#lit10, Wn$ | $Wn = lit10 .XOR. Wn$ | 1 | 1 | 438 |
| XOR $Wb, \#lit5, Wd$ | $Wd = Wb .XOR. lit5$ | 1 | 1 | 439 |
| XOR Wb, Ws, Wd | $Wd = Wb .XOR. Ws$ | 1 | 1 ⁽²⁾ | 440 |

- 注 1: 当指定了可选的 $\{, WREG\}$ 操作数时, 指令的目标寄存器为 $WREG$ 。当未指定 $\{, WREG\}$ 时, 指令的目标寄存器为文件寄存器 f 。
- 2: 在 $dsPIC33E$ 和 $PIC24E$ 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 $dsPIC30F$ 、 $dsPIC33F$ 、 $PIC24F$ 和 $PIC24H$ 器件相比。

表 3-5: 循环 / 移位指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|-----------------------------|---|----|-----------|---------------------|
| ASR $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 算术右移 f , $LSb \rightarrow C$ | 1 | $1^{(2)}$ | 117 |
| ASR Ws, Wd | Wd = 算术右移 Ws , $LSb \rightarrow C$ | 1 | $1^{(2)}$ | 119 |
| ASR $Wb, \#lit4, Wnd$ | Wnd = Wb 算术右移 $lit4$ 位, $LSb \rightarrow C$ | 1 | 1 | 121 |
| ASR Wb, Wns, Wnd | Wnd = Wb 算术右移 Wns 位, $LSb \rightarrow C$ | 1 | 1 | 122 |
| LSR $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 逻辑右移 f , $LSb \rightarrow C$ | 1 | $1^{(2)}$ | 269 |
| LSR Ws, Wd | Wd = 逻辑右移 Ws , $LSb \rightarrow C$ | 1 | $1^{(2)}$ | 271 |
| LSR $Wb, \#lit4, Wnd$ | Wnd = Wb 逻辑右移 $lit4$ 位, $LSb \rightarrow C$ | 1 | 1 | 273 |
| LSR Wb, Wns, Wnd | Wnd = Wb 逻辑右移 Wns 位, $LSb \rightarrow C$ | 1 | 1 | 274 |
| RLC $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 带进位位循环左移 f 内容 | 1 | $1^{(2)}$ | 373 |
| RLC Ws, Wd | Wd = 带进位位循环左移 Ws 内容 | 1 | $1^{(2)}$ | 375 |
| RLNC $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 不带进位位循环左移 f 内容 | 1 | $1^{(2)}$ | 377 |
| RLNC Ws, Wd | Wd = 不带进位位循环左移 Ws 内容 | 1 | $1^{(2)}$ | 379 |
| RRC $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 带进位位循环右移 f 内容 | 1 | $1^{(2)}$ | 381 |
| RRC Ws, Wd | Wd = 带进位位循环右移 Ws 内容 | 1 | $1^{(2)}$ | 383 |
| RRNC $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 不带进位位循环右移 f 内容 | 1 | $1^{(2)}$ | 385 |
| RRNC Ws, Wd | Wd = 不带进位位循环右移 Ws 内容 | 1 | $1^{(2)}$ | 387 |
| SL $f \{ , WREG \}^{(1)}$ | 目标寄存器 = 左移 f 内容, $MSb \rightarrow C$ | 1 | $1^{(2)}$ | 399 |
| SL Ws, Wd | Wd = 左移 Ws 内容, $MSb \rightarrow C$ | 1 | $1^{(2)}$ | 401 |
| SL $Wb, \#lit4, Wnd$ | Wnd = Wb 左移 $lit4$ 位, $MSb \rightarrow C$ | 1 | 1 | 403 |
| SL Wb, Wns, Wnd | Wnd = Wb 左移 Wns 位, $MSb \rightarrow C$ | 1 | 1 | 404 |

- 注 1: 当指定了可选的 $\{ , WREG \}$ 操作数时, 指令的目标寄存器为 $WREG$ 。当未指定 $\{ , WREG \}$ 时, 指令的目标寄存器为文件寄存器 f 。
- 2: 在 $dsPIC33E$ 和 $PIC24E$ 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 $dsPIC30F$ 、 $dsPIC33F$ 、 $PIC24F$ 和 $PIC24H$ 器件相比。

表 3-6: 位操作指令

| 汇编语法 | 说明 | 字数 | 周期数 (1) | 页码 |
|----------------------|--|----|---------|---------------------|
| BCLR $f, \#bit4$ | 将 f 中的指定位清零 | 1 | 1 | 123 |
| BCLR $Ws, \#bit4$ | 将 Ws 中的指定位清零 | 1 | 1 | 124 |
| BSET $f, \#bit4$ | 将 f 中的指定位置 1 | 1 | 1 | 152 |
| BSET $Ws, \#bit4$ | 将 Ws 中的指定位置 1 | 1 | 1 | 153 |
| BSW.C Ws, Wb | 将 C 位内容写入 $Ws<Wb>$ | 1 | 1 | 155 |
| BSW.Z Ws, Wb | 将 \bar{Z} 位内容写入 $Ws<Wb>$ | 1 | 1 | 155 |
| BTG $f, \#bit4$ | 将 f 中的指定位翻转 | 1 | 1 | 157 |
| BTG $Ws, \#bit4$ | 将 Ws 中的指定位翻转 | 1 | 1 | 158 |
| BTST $f, \#bit4$ | 测试 f 中的指定位, 并将被测试位的反码存储到全零标志位 Z 中 | 1 | 1 | 168 |
| BTST.C $Ws, \#bit4$ | 测试 Ws 中的指定位, 并将被测试位的值存储到进位标志位 C 中 | 1 | 1 | 169 |
| BTST.Z $Ws, \#bit4$ | 测试 Ws 中的指定位, 并将被测试位的反码存储到全零标志位 Z 中 | 1 | 1 | 169 |
| BTST.C Ws, Wb | 测试 $Ws<Wb>$ 中的指定位, 并将被测试位的值存储到进位标志位 C 中 | 1 | 1 | 171 |
| BTST.Z Ws, Wb | 测试 $Ws<Wb>$ 中的指定位, 并将被测试位的反码存储到全零标志位 Z 中 | 1 | 1 | 171 |
| BTSTS $f, \#bit4$ | 测试 f 中的指定位, 并将被测试位的反码存储到全零标志位 Z 中, 然后将 f 中的该位置 1 | 1 | 1 | 173 |
| BTSTS.C $Ws, \#bit4$ | 测试 Ws 中的指定位, 并将被测试位的值存储到进位标志位 C 中, 然后将 Ws 中的该位置 1 | 1 | 1 | 175 |
| BTSTS.Z $Ws, \#bit4$ | 测试 Ws 中的指定位, 并将被测试位的反码存储到全零标志位 Z 中, 然后将 Ws 中的该位置 1 | 1 | 1 | 175 |
| FBCL Ws, Wnd | 搜索自左 (MSb) 起第一个位变化 | 1 | 1 | 244 |
| FF1L Ws, Wnd | 搜索自左 (MSb) 起第一个 1 | 1 | 1 | 246 |
| FF1R Ws, Wnd | 搜索自右 (LSb) 起第一个 1 | 1 | 1 | 248 |

注 1: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

表 3-7: 比较 / 跳过和比较 / 转移指令

| 汇编语法 | 说明 | 字数 | 周期数 ⁽¹⁾ | 页码 |
|----------------------------|--------------------------------------|----|--------------------------|---------------------|
| BTSC $f, \#bit4$ | 测试 f 中的指定位, 为零则跳过 | 1 | 1 (2 或 3) ⁽⁵⁾ | 160 |
| BTSC $Ws, \#bit4$ | 测试 Ws 中的指定位, 为零则跳过 | 1 | 1 (2 或 3) ⁽⁵⁾ | 162 |
| BTSS $f, \#bit4$ | 测试 f 中的指定位, 为 1 则跳过 | 1 | 1 (2 或 3) ⁽⁵⁾ | 164 |
| BTSS $Ws, \#bit4$ | 测试 Ws 中的指定位, 为 1 则跳过 | 1 | 1 (2 或 3) ⁽⁵⁾ | 166 |
| CP f | 比较 ($f - WREG$) | 1 | 1 ⁽⁵⁾ | 191 |
| CP $Wb, \#lit5^{(2)}$ | 比较 ($Wb - lit5$) | 1 | 1 | 192 |
| CP $Wb, \#lit8^{(3)}$ | 比较 ($Wb - lit8$) | 1 | 1 | 193 |
| CP Wb, Ws | 比较 ($Wb - Ws$) | 1 | 1 ⁽⁵⁾ | 194 |
| CP0 f | 比较 ($f - 0x0000$) | 1 | 1 ⁽⁵⁾ | 196 |
| CP0 Ws | 比较 ($Ws - 0x0000$) | 1 | 1 ⁽⁵⁾ | 197 |
| CPB f | 带借位比较 ($f - WREG - \overline{C}$) | 1 | 1 ⁽⁵⁾ | 198 |
| CPB $Wb, \#lit5^{(2)}$ | 带借位比较 ($Wb - lit5 - \overline{C}$) | 1 | 1 | 199 |
| CPB $Wb, \#lit8^{(3)}$ | 带借位比较 ($Wb - lit8 - \overline{C}$) | 1 | 1 | 200 |
| CPB Wb, Ws | 带借位比较 ($Wb - Ws - \overline{C}$) | 1 | 1 ⁽⁵⁾ | 201 |
| CPBEQ $Wb, Wn, Expr^{(3)}$ | 比较 Wb 和 Wn , 如果相等则转移 | 1 | 1 (5) ⁽⁴⁾ | 203 |
| CPBGT $Wb, Wn, Expr^{(3)}$ | 有符号比较 Wb 和 Wn , 如果大于则转移 | 1 | 1 (5) ⁽⁴⁾ | 204 |
| CPBLT $Wb, Wn, Expr^{(3)}$ | 有符号比较 Wb 和 Wn , 如果小于则转移 | 1 | 1 (5) ⁽⁴⁾ | 205 |
| CPBNE $Wb, Wn, Expr^{(3)}$ | 比较 Wb 和 Wn , 如果不等于则转移 | 1 | 1 (5) ⁽⁴⁾ | 206 |
| CPSEQ Wb, Wn | 比较 ($Wb - Wn$), 如果相等则跳过 | 1 | 1 (2 或 3) | 207 |
| CPSGT Wb, Wn | 有符号比较 ($Wb - Wn$), 如果大于则跳过 | 1 | 1 (2 或 3) | 211 |
| CPSLT Wb, Wn | 有符号比较 ($Wb - Wn$), 如果小于则跳过 | 1 | 1 (2 或 3) | 212 |
| CPSNE Wb, Wn | 比较 ($Wb - Wn$), 如果不等于则跳过 | 1 | 1 (2 或 3) | 214 |

- 注 1: 如果未发生跳过, 则条件跳过指令的执行时间为 1 个周期。如果跳过的是单字指令, 则执行时间为 2 个周期; 而如果跳过的是双字指令, 则执行时间为 3 个周期。
- 2: 该指令仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
- 3: 该指令仅在 dsPIC33E 和 PIC24E 器件中可用。
- 4: 在 dsPIC33E/PIC24E 器件中, 如果比较 - 转移指令未发生转移, 则执行时间为 1 个周期; 而如果发生转移, 则执行时间为 5 个周期。
- 5: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

表 3-8: 程序流指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|--------------------------------|--|----|------------------------|-----|
| BRA Expr | 无条件转移 | 1 | 2 ⁽⁸⁾ | 126 |
| BRA Wn | 计算转移 | 1 | 2 ⁽⁸⁾ | 128 |
| BRA C, Expr | 如果进位（无借位）则转移 | 1 | 1 (2) ^(1,8) | 130 |
| BRA GE, Expr | 如果大于或等于则转移 | 1 | 1 (2) ^(1,8) | 132 |
| BRA GEU, Expr | 如果无符号大于或等于则转移 | 1 | 1 (2) ^(1,8) | 134 |
| BRA GT, Expr | 如果大于则转移 | 1 | 1 (2) ^(1,8) | 135 |
| BRA GTU, Expr | 如果无符号大于则转移 | 1 | 1 (2) ^(1,8) | 136 |
| BRA LE, Expr | 如果小于或等于则转移 | 1 | 1 (2) ^(1,8) | 137 |
| BRA LEU, Expr | 如果无符号小于或等于则转移 | 1 | 1 (2) ^(1,8) | 138 |
| BRA LT, Expr | 如果小于则转移 | 1 | 1 (2) ^(1,8) | 139 |
| BRA LTU, Expr | 如果无符号小于则转移 | 1 | 1 (2) ^(1,8) | 140 |
| BRA N, Expr | 如果为负则转移 | 1 | 1 (2) ^(1,8) | 141 |
| BRA NC, Expr | 如果无进位（有借位）则转移 | 1 | 1 (2) ^(1,8) | 142 |
| BRA NN, Expr | 如果非负则转移 | 1 | 1 (2) ^(1,8) | 143 |
| BRA NOV, Expr | 如果未溢出则转移 | 1 | 1 (2) ^(1,8) | 144 |
| BRA NZ, Expr | 如果非零则转移 | 1 | 1 (2) ^(1,8) | 145 |
| BRA OA, Expr | 如果累加器 A 溢出则转移 | 1 | 1 (2) ^(1,8) | 146 |
| BRA OB, Expr | 如果累加器 B 溢出则转移 | 1 | 1 (2) ^(1,8) | 147 |
| BRA OV, Expr | 如果溢出则转移 | 1 | 1 (2) ^(1,8) | 148 |
| BRA SA, Expr | 如果累加器 A 饱和则转移 | 1 | 1 (2) ^(1,8) | 149 |
| BRA SB, Expr | 如果累加器 B 饱和则转移 | 1 | 1 (2) ^(1,8) | 150 |
| BRA Z, Expr | 如果为零则转移 | 1 | 1 (2) ^(1,8) | 151 |
| CALL Expr | 调用子程序 | 2 | 2 ⁽⁸⁾ | 177 |
| CALL Wn | 间接调用子程序 | 1 | 2 ⁽⁸⁾ | 180 |
| CALL.L Wn ⁽⁴⁾ | 间接调用子程序（长地址） | 1 | 4 | 183 |
| DO #lit14, Expr ⁽⁶⁾ | 执行 DO 循环代码到地址 PC + Expr, 执行次数为 (lit14 + 1) 次 | 2 | 2 | 230 |
| DO #lit15, Expr ⁽⁷⁾ | 执行 DO 循环代码到地址 PC + Expr, 执行次数为 (lit15 + 1) 次 | 2 | 2 | 233 |
| DO Wn, Expr ⁽³⁾ | 执行 DO 循环代码到地址 PC + Expr, 执行次数为 (Wn + 1) 次 | 2 | 2 | 235 |
| GOTO Expr | 跳转到地址 | 2 | 2 ⁽⁸⁾ | 250 |
| GOTO Wn | 间接跳转到地址 | 1 | 2 ⁽⁸⁾ | 251 |
| GOTO.L Wn ⁽⁴⁾ | 间接跳转到地址（长地址） | 1 | 4 | 253 |
| RCALL Expr | 相对调用 | 1 | 2 ⁽⁸⁾ | 347 |
| RCALL Wn | 计算调用 | 1 | 2 ⁽⁸⁾ | 351 |
| REPEAT #lit14 ⁽⁵⁾ | 重复执行下一条指令 (lit14 + 1) 次 | 1 | 1 | 355 |

- 注 1: 如果未发生转移, 则条件转移指令的执行时间为 1 个周期; 而如果发生转移, 则为 2 个周期。
- 2: RETURN 指令的执行时间为 3 个周期, 但如果有一个异常等待处理, 则执行时间为 2 个周期。
- 3: 该指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
- 4: 该指令仅在 dsPIC33E 和 PIC24E 器件中可用。
- 5: 该指令仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
- 6: 该指令仅在 dsPIC30F 和 dsPIC33F 器件中可用。
- 7: 该指令仅在 dsPIC33E 器件中可用。
- 8: 在 dsPIC33E 和 PIC24E 器件中, 当发生转移时, 这些指令需要 2 个额外的周期（总共 4 个周期）。
- 9: 在 dsPIC33E 和 PIC24E 器件中, 这些指令需要 3 个额外的周期。

表 3-8: 程序流指令 (续)

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|------------------------------|-------------------------|----|------------------------|---------------------|
| REPEAT #lit15 ⁽⁴⁾ | 重复执行下一条指令 (lit15 + 1) 次 | 1 | 1 | 357 |
| REPEAT Wn | 重复执行下一条指令 (Wn + 1) 次 | 1 | 1 | 359 |
| RETFIE | 从中断返回 | 1 | 3 (2) ^(2,9) | 365 |
| RETLW #lit10,Wn | 从子程序返回并将 lit10 存入 Wn | 1 | 3 (2) ^(2,9) | 367 |
| RETURN | 从子程序返回 | 1 | 3 (2) ^(2,9) | 371 |

- 注
- 1: 如果未发生转移, 则条件转移指令的执行时间为 1 个周期; 而如果发生转移, 则为 2 个周期。
 - 2: RETURN 指令的执行时间为 3 个周期, 但如果有一个异常等待处理, 则执行时间为 2 个周期。
 - 3: 该指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 4: 该指令仅在 dsPIC33E 和 PIC24E 器件中可用。
 - 5: 该指令仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令仅在 dsPIC30F 和 dsPIC33F 器件中可用。
 - 7: 该指令仅在 dsPIC33E 器件中可用。
 - 8: 在 dsPIC33E 和 PIC24E 器件中, 当发生转移时, 这些指令需要 2 个额外的周期 (总共 4 个周期)。
 - 9: 在 dsPIC33E 和 PIC24E 器件中, 这些指令需要 3 个额外的周期。

表 3-9: 影子 / 堆栈指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|------------|------------------------|----|------------------|---------------------|
| LNK #lit14 | 分配堆栈帧 | 1 | 1 | 267 |
| POP f | 栈顶内容弹出至 f | 1 | 1 | 337 |
| POP Wdo | 栈顶内容弹出至 Wdo | 1 | 1 | 338 |
| POP.D Wnd | 将栈顶内容双字弹出至 Wnd:Wnd + 1 | 1 | 2 | 339 |
| POP.S | 弹出影子寄存器内容 | 1 | 1 | 340 |
| PUSH f | 将 f 内容压入栈顶 | 1 | 1 ⁽¹⁾ | 341 |
| PUSH Wso | 将 Wso 内容压入栈顶 | 1 | 1 ⁽¹⁾ | 342 |
| PUSH.D Wns | 将 Wns:Wns + 1 内容双字压入栈顶 | 1 | 2 | 343 |
| PUSH.S | 压入影子寄存器 | 1 | 1 | 345 |
| ULNK | 释放堆栈帧 | 1 | 1 | 435 |

注 1: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 相比。

表 3-10: 控制指令

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|---------------|-------------------------------|----|-----|---------------------|
| CLRWDT | 清零看门狗定时器 | 1 | 1 | 188 |
| DISI #lit14 | 禁止中断, 持续时间为 (lit14 + 1) 个指令周期 | 1 | 1 | 223 |
| NOP | 空操作 | 1 | 1 | 336 |
| NOPR | 空操作 | 1 | 1 | 336 |
| PWRSABV #lit1 | 进入低功耗模式 | 1 | 1 | 346 |
| RESET | 软件器件复位 | 1 | 1 | 363 |

16 位 MCU 和 DSC 程序员参考手册

表 3-11: DSP 类指令 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| 汇编语法 | 说明 | 字数 | 周期数 | 页码 |
|---|-----------------------------------|----|------------------|-----|
| ADD Acc | 累加器相加 | 1 | 1 | 103 |
| ADD Wso, #Slit4, Acc | 16 位有符号数与累加器相加 | 1 | 1 ⁽¹⁾ | 104 |
| CLR Acc, [Wx], Wxd, [Wy], Wyd, AWB | 清零累加器 | 1 | 1 | 186 |
| ED Wm*Wm, Acc, [Wx], [Wy], Wxd | 求取欧几里德距离 (无累加) | 1 | 1 | 239 |
| EDAC Wm*Wm, Acc, [Wx], [Wy], Wxd | 求取欧几里德距离 | 1 | 1 | 241 |
| LAC Wso, #Slit4, Acc | 装载累加器 | 1 | 1 ⁽¹⁾ | 265 |
| MAC Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd, AWB | 相乘并累加 | 1 | 1 | 275 |
| MAC Wm*Wm, Acc, [Wx], Wxd, [Wy], Wyd | 平方并累加 | 1 | 1 | 277 |
| MOVSAC Acc, [Wx], Wxd, [Wy], Wyd, AWB | 将 Wx 中内容传送到 Wxd, 并将 Wy 中内容传送到 Wyd | 1 | 1 | 293 |
| MPY Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd | Wn 和 Wm 相乘, 结果存入累加器 | 1 | 1 | 295 |
| MPY Wm*Wm, Acc, [Wx], Wxd, [Wy], Wyd | 求平方, 结果存入累加器 | 1 | 1 | 297 |
| MPY.N Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd | Wn 和 Wm 相乘并取反, 结果存入累加器 | 1 | 1 | 299 |
| MSC Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd, AWB | 相乘并从累加器中减去乘积 | 1 | 1 | 301 |
| NEG Acc | 对累加器内容求补 | 1 | 1 | 335 |
| SAC Acc, #Slit4, Wdo | 保存累加器内容 | 1 | 1 | 389 |
| SAC.R Acc, #Slit4, Wdo | 保存舍入后的累加器内容 | 1 | 1 | 391 |
| SFTAC Acc, #Slit6 | 算术移位累加器内容 Slit6 位 | 1 | 1 | 397 |
| SFTAC Acc, Wn | 算术移位累加器内容 (Wn) 位 | 1 | 1 | 398 |
| SUB Acc | 累加器相减 | 1 | 1 | 410 |

注 1: 在 dsPIC33E 和 PIC24E 器件中, 对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作, 都需要 1 个额外周期 —— 与 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件相比。

第 4 章 指令集详解

目录

本章包括以下主要内容：

| | | |
|------|---|----|
| 4.1 | 数据空间寻址模式 | 52 |
| 4.2 | 程序空间寻址模式 | 61 |
| 4.3 | 指令停顿 | 62 |
| 4.4 | 字节操作 | 64 |
| 4.5 | 字传送操作 | 66 |
| 4.6 | 使用 10 位立即数操作数 | 69 |
| 4.7 | 软件堆栈指针和帧指针 | 70 |
| 4.8 | 条件转移指令 | 76 |
| 4.9 | Z 状态位 | 77 |
| 4.10 | 规定的工作寄存器用法 | 78 |
| 4.11 | DSP 数据格式（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 81 |
| 4.12 | 累加器的使用（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 83 |
| 4.13 | 累加器访问（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 84 |
| 4.14 | DSP MAC 类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 84 |
| 4.15 | DSP 累加器类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 88 |
| 4.16 | 使用 FBCL 指令换算数据（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 88 |
| 4.17 | 使用 FBCL 指令将累加器中内容归一化（dsPIC30F、dsPIC33F 和 dsPIC33E 器件） | 90 |

4.1 数据空间寻址模式

16 位 MCU 和 DSC 器件对数据存储器的访问，除几种立即数寻址形式外，还支持三种固有寻址模式。通过文件寄存器寻址、寄存器直接寻址或寄存器间接寻址模式实现对数据存储器的访问，而立即数寻址使得指令可以使用固定值作为操作数。

通过文件寄存器寻址可实现对存放在数据存储空间低 8 KB 存储区（Near RAM）中的数据进行操作，且可实现工作寄存器和整个 64 KB 数据空间之间的数据传送。寄存器直接寻址模式用来对 16 个存储区映射的工作寄存器 W0:W15 进行访问。寄存器间接寻址模式通过使用工作寄存器中内容作为有效地址，可对存储在整个 64 KB 数据存储空间（以及扩展数据空间，如果是 dsPIC33E/PIC24E）中的数据进行高效操作。立即数寻址模式并不直接访问数据存储区，但可使用常量值作为指令操作数。表 4-1 对所有模式的寻址范围进行了汇总。

表 4-1: 16 位 MCU 和 DSC 的寻址模式

| 寻址模式 | 寻址范围 |
|---------|-------------------------------|
| 文件寄存器寻址 | 0x0000–0x1FFF ⁽¹⁾ |
| 寄存器直接寻址 | 0x0000–0x001F（工作寄存器阵列 W0:W15） |
| 寄存器间接寻址 | 0x0000–0xFFFF |
| 立即数寻址 | N/A（常量值） |

注 1：文件寄存器 MOV 的寻址范围是 0x0000–0xFFFE。

4.1.1 文件寄存器寻址

文件寄存器寻址模式由使用预先确定的数据地址作为操作数的指令使用。大多数支持文件寄存器寻址模式的指令都可对数据存储空间的低 8 KB 存储区进行访问，该存储区又称为 Near RAM。然而，MOV 指令可使用文件寄存器寻址模式对所有 64 KB 存储空间进行访问。这将允许把数据存储区中任何地址单元中的数据装载到任一工作寄存器，以及将任一工作寄存器的内容存放到数据存储区的任何地址单元中。应注意，文件寄存器寻址模式支持以字节和字两种方式对数据存储空间进行访问，而 MOV 指令则是例外，它以字方式对所有 64 KB 存储空间进行访问。例 4-1 中给出了文件寄存器寻址的指令示例。

支持文件寄存器寻址的大多数指令以指定文件寄存器和默认工作寄存器 WREG 为操作对象（见第 2.4 节“默认工作寄存器（WREG）”）。如果指令中只提供一个操作数，WREG 将是一个隐含的操作数，且操作结果将存回文件寄存器。此时，该指令实际上是一条读 - 修改 - 写指令。然而，当指令中同时指定文件寄存器和 WREG 寄存器时，操作结果将存放在 WREG 寄存器中，而文件寄存器中的内容将不发生改变。例 4-2 中给出了文件寄存器和 WREG 寄存器之间相互作用的指令示例。

注：如第 3 章“指令集概述”中的指令汇总表所示，支持文件寄存器寻址的指令使用 f 作为操作数。

例 4-1: 文件寄存器寻址

DEC0x1000; 对存储于 0x1000 的数据进行递减

指令执行前:

数据存储区 0x1000 = 0x5555

指令执行后:

数据存储区 0x1000 = 0x5554

MOV0x27FE, W0; 将存储于 0x27FE 的数据传送至 W0

指令执行前:

W0 = 0x5555

数据存储区 0x27FE = 0x1234

指令执行后:

W0 = 0x1234

数据存储区 0x27FE = 0x1234

例 4-2: 文件寄存器寻址和 WREG

AND0x1000; 将 0x1000 和 WREG 的内容相与，结果存入 0x1000

指令执行前:

W0 (WREG) = 0x332C

数据存储区 0x1000 = 0x5555

指令执行后:

W0 (WREG) = 0x332C

数据存储区 0x1000 = 0x1104

AND0x1000, WREG; 将 0x1000 和 WREG 的内容相与，结果存入 WREG

指令执行前:

W0 (WREG) = 0x332C

数据存储区 0x1000 = 0x5555

指令执行后:

W0 (WREG) = 0x1104

数据存储区 0x1000 = 0x5555

4.1.2 寄存器直接寻址

采用寄存器直接寻址模式对 16 个工作寄存器（W0:W15）的内容进行访问。寄存器直接寻址模式为全正交模式，允许为使用寄存器直接寻址的指令指定任何工作寄存器，且支持字节和字访问方式。采用寄存器直接寻址的指令使用指定工作寄存器中的内容作为数据来执行指令，因此只有当数据已存入工作寄存器内时才可使用该寻址模式。例 4-3 中给出了使用寄存器直接寻址模式的指令示例。

寄存器直接寻址的另一特点是它可提供动态流控制能力。由于 DO 和 REPEAT 指令的不同形式支持寄存器直接寻址，因此可使用这些指令实现灵活的循环结构。

注：对于那些必须使用寄存器直接寻址模式的指令，需要用到符号 Wb、Wn、Wns 和 Wnd，详见第 3 章“指令集概述”的汇总表。通常，可使用寄存器间接寻址模式的场合也可使用寄存器直接寻址。使用寄存器间接寻址的指令使用第 3 章“指令集概述”汇总表中的符号 Wd 和 Ws。

例 4-3: 寄存器直接寻址

EXCH W2, W3 ; 将 W2 和 W3 中的内容进行交换

指令执行前:

W2 = 0x3499

W3 = 0x003D

指令执行后:

W2 = 0x003D

W3 = 0x3499

IOR #0x44, W0 ; 将 0x44 和 W0 的内容相或

指令执行前:

W0 = 0x9C2E

指令执行后:

W0 = 0x9C6E

SL W6, W7, W8 ; 将 W6 内容左移 W7 位, 并将结果存入 W8

指令执行前:

W6 = 0x000C

W7 = 0x0008

W8 = 0x1234

指令执行后:

W6 = 0x000C

W7 = 0x0008

W8 = 0x0C00

4.1.3 寄存器间接寻址

寄存器间接寻址模式通过将工作寄存器中内容作为数据存储区有效地址（Effective Address, EA），可实现对数据空间中任何存储单元的访问。实质上，工作寄存器中的内容变为一个指向指令要访问的数据存储单元的指针。

因为该寻址模式允许在进行数据访问之前或之后，通过对 EA 进行递增或递减来修改工作寄存器中的内容，因此可实现强大的功能。通过在访问操作执行的同一周期内对 EA 中的内容进行修改，寄存器间接寻址可对顺序存储在存储区中的数据进行高效处理。表 4-2 给出了 16 位 MCU 和 DSC 器件所支持的间接寻址模式。

表 4-2: 间接寻址模式

| 间接模式 | 语法 | 功能 (字节指令) | 功能 (字指令) | 说明 |
|--------|---------|-----------------|-----------------|-----------------------------------|
| 无修改 | [Wn] | EA = [Wn] | EA = [Wn] | Wn 中的内容形成 EA。 |
| 执行前递增 | [++Wn] | EA = [Wn + = 1] | EA = [Wn + = 2] | 对 Wn 中内容进行递增以形成 EA。 |
| 执行前递减 | [--Wn] | EA = [Wn - = 1] | EA = [Wn - = 2] | 对 Wn 中内容进行递减以形成 EA。 |
| 执行后递增 | [Wn++] | EA = [Wn]+ = 1 | EA = [Wn]+ = 2 | Wn 中内容形成 EA, 然后 Wn 中内容进行递增。 |
| 执行后递减 | [Wn--] | EA = [Wn] - = 1 | EA = [Wn] - = 2 | Wn 中内容形成 EA, 然后 Wn 中内容进行递减。 |
| 寄存器偏移量 | [Wn+Wb] | EA = [Wn + Wb] | EA = [Wn + Wb] | Wn 和 Wb 的和形成 EA。Wn 和 Wb 中的内容未被修改。 |

表 4-2 显示了四种寻址模式对指令中使用的 EA 进行修改，且允许对工作寄存器进行以下更新：执行后递增、执行后递减、执行前递增以及执行前递减。由于所有 EA 必须赋以字节地址，因此 EA 将以 2 为单位进行更新以支持字模式指令。也就是说，在字模式中指令执行前 / 后递减将对存储于工作寄存器中的 EA 减 2，而指令执行前 / 后递增将对 EA 进行加 2。该功能确保在 EA 修改之后，EA 将指向存储区中下一个相邻的字。例 4-4 说明了如何使用间接寻址对 EA 进行更新。

表 4-2 也阐明了使用寄存器偏移量模式寻址的情况，该寻址模式对距离一个工作寄存器中的基准 EA 某个偏移量的数据进行寻址。该模式使用另外一個工作寄存器的内容，通过将两个指定的工作寄存器相加来形成 EA。该寻址模式不适用于字模式指令，但其提供了整个 64 KB 偏移量范围。注意，用来形成 EA 的工作寄存器中内容将不会被修改。例 4-5 说明了如何使用寄存器偏移量间接寻址对数据存储区进行访问。

注： 带偏移量的 MOV 指令（见第 285 页和第 286 页）提供了在间接寻址时可使用的立即数偏移量寻址功能。在这些指令中，通过将工作寄存器的内容加到一个 10 位有符号立即数以形成 EA。例 4-6 显示了如何利用这些指令实现从 / 向工作寄存器阵列传送数据。

例 4-4: 带有效地址更新的间接寻址

```
MOV.B    [W0++], [W13--]      ; 字节传送 [W0] 内容至 [W13]
                                   ; 指令执行后递增 W0，指令执行后递减 W13

指令执行前:
W0 = 0x2300
W13 = 0x2708
数据存储区 0x2300 = 0x7783
数据存储区 0x2708 = 0x904E

指令执行后:
W0 = 0x2301
W13 = 0x2707
数据存储区 0x2300 = 0x7783
数据存储区 0x2708 = 0x9083

ADD      W1, [--W5], [++W8]    ; 指令执行前递减 W5，指令执行前递增 W8
                                   ; 将 W1 内容与 [W5] 相加，结果存入 [W8]

指令执行前:
W1 = 0x0800
W5 = 0x2200
W8 = 0x2400
数据存储区 0x21FE = 0x7783
数据存储区 0x2402 = 0xAACC

指令执行后:
W1 = 0x0800
W5 = 0x21FE
W8 = 0x2402
数据存储区 0x21FE = 0x7783
数据存储区 0x2402 = 0x7F83
```

例 4-5: 寄存器偏移量间接寻址

```
MOV.B  [W0+W1], [W7++]      ; 字节传送 [W0+W1] 内容至 W7, 指令执行后递增 W7
```

指令执行前:

```
W0 = 0x2300
W1 = 0x01FE
W7 = 0x1000
数据存储区 0x24FE = 0x7783
数据存储区 0x1000 = 0x11DC
```

指令执行后:

```
W0 = 0x2300
W1 = 0x01FE
W7 = 0x1001
数据存储区 0x24FE = 0x7783
数据存储区 0x1000 = 0x1183
```

```
LAC      [W0+W8], A          ; 将 [W0+W8] 内容装入 ACCA
                                ; (符号扩展和零回填)
```

指令执行前:

```
W0 = 0x2344
W8 = 0x0008
ACCA = 0x00 7877 9321
数据存储区 0x234C = 0xE290
```

指令执行后:

```
W0 = 0x2344
W8 = 0x0008
ACCA = 0xFF E290 0000
数据存储区 0x234C = 0xE290
```

例 4-6: 用立即数偏移量进行传送的指令

```
MOV      [W0+0x20], W1      ; 将 [W0+0x20] 内容传送至 W1
```

指令执行前:

```
W0 = 0x1200
W1 = 0x01FE
数据存储区 0x1220 = 0xFD27
```

指令执行后:

```
W0 = 0x1200
W1 = 0xFD27
数据存储区 0x1220 = 0xFD27
```

```
MOV      W4, [W8-0x300]     ; 将 W4 内容传送至 [W8-0x300]
```

指令执行前:

```
W4 = 0x3411
W8 = 0x2944
数据存储区 0x2644 = 0xCB98
```

指令执行后:

```
W4 = 0x3411
W8 = 0x2944
数据存储区 0x2644 = 0x3411
```


4.1.3.1 寄存器间接寻址和指令集

表 4-2 中给出的寻址模式展示了 16 位 MCU 和 DSC 器件的间接寻址模式能力。出于操作编码和功能上的考虑，并非所有支持间接寻址的指令都支持表 4-2 中所示的所有模式。使用间接寻址模式的大多数指令支持无修改、执行前递增、执行前递减、执行后递增和执行后递减寻址模式。MOV 指令和几种基于累加器的 DSP 类指令（仅限 dsPIC30F、dsPIC33F 和 dsPIC33E 器件）也具备使用寄存器偏移量寻址模式的能力。

注：使用寄存器间接寻址的指令使用第 3 章“指令集概述”汇总表中的操作数符号 Wd 和 Ws。

4.1.3.2 DSP MAC 间接寻址模式（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DSP MAC 类指令采用一类特殊的间接寻址模式。如后面第 4.14 节“DSP MAC 类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所介绍，DSP MAC 类指令能够使用有效寻址两次从存储区取操作数。由于 DSP 算法经常需要更宽范围的地址更新，DSP MAC 类指令提供的寻址模式可实现更宽的有效地址更新大小范围。表 4-3 显示 X 和 Y 存储区的预取都支持执行后递增和执行后递减寻址模式，且更新方式可为 2、4 和 6 字节。由于 DSP 类指令只以字模式的方式执行，EA 将不会以奇数方式进行更新。

表 4-3: DSP MAC 间接寻址模式

| 寻址模式 | X 存储区 | Y 存储区 |
|---------------|-----------------|------------------|
| 无修改的间接寻址 | EA = [Wx] | EA = [Wy] |
| 执行后递增 2 的间接寻址 | EA = [Wx] + = 2 | EA = [Wy] + = 2 |
| 执行后递增 4 的间接寻址 | EA = [Wx] + = 4 | EA = [Wy] + = 4 |
| 执行后递增 6 的间接寻址 | EA = [Wx] + = 6 | EA = [Wy] + = 6 |
| 执行后递减 2 的间接寻址 | EA = [Wx] - = 2 | EA = [Wy] - = 2 |
| 执行后递减 4 的间接寻址 | EA = [Wx] - = 4 | EA = [Wy] - = 4 |
| 执行后递减 6 的间接寻址 | EA = [Wx] - = 6 | EA = [Wy] - = 6 |
| 寄存器偏移量间接寻址 | EA = [W9 + W12] | EA = [W11 + W12] |

注：如第 4.14 节“DSP MAC 类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所介绍，只有 W8 和 W9 才可用来访问 X 存储区，而只有 W10 和 W11 才可用来访问 Y 存储区。

4.1.3.3 模寻址和位反转寻址模式（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

16 位 DSC 架构支持两种通常用来实现 DSP 算法的特殊寄存器间接寻址模式。模寻址（又称循环寻址）提供了一种自动支持 X 和 / 或 Y 存储区中循环数据缓冲区的方法。模缓冲区使得软件不再需要进行地址边界检查，这将改善某些算法的性能。类似地，位反转寻址可以实现以一种非线性方式对缓冲区中的单元进行访问。这种寻址模式简化了用于基为 2 的 FFT 算法的数据重新排序，并大大减少了 FFT 的处理时间。

这两种寻址模式体现了 dsPIC30F、dsPIC33F 和 dsPIC33E 架构的强大功能。使用间接寻址的任何指令都可利用这两种寻址模式。有关使用模寻址和位反转寻址的详细信息，请参见具体器件的系列参考手册。

4.1.4 立即数寻址

在立即数寻址模式中，指令编码包含一个指令使用的预先定义常量操作数。该寻址模式可以独立使用，但通常是同文件寄存器、直接和间接寻址模式联合使用。根据指令类型的不同，可以使用长度不同的立即数操作数。常量的长度可以是 1 位（#lit1）、4 位（#bit4、#lit4 和 #Slit4）、5 位（#lit5）、6 位（#Slit6）、8 位（#lit8）、10 位（#lit10 和 #Slit10）、14 位（#lit14）以及 16 位（#lit16）。常量可以是有符号或无符号的，符号 #Slit4、#Slit6 和 #Slit10 指定为有符号常量，而所有其他立即数常量都是无符号的。表 4-4 给出了每种立即数操作数在指令集中的使用。

注： 6 位（#Slit6）操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

表 4-4: 指令集中的立即数操作数

| 操作数 | 使用的指令 |
|-----------------------|---|
| #lit1 | PWRSV |
| #bit4 | BCLR, BSET, BTG, BTSC, BTSS, BTST, BTST.C, BTST.Z, BTSTS, BTSTS.C, BTSTS.Z |
| #lit4 | ASR, LSR, SL |
| #Slit4 | ADD, LAC, SAC, SAC.R |
| #lit5 | ADD, ADDC, AND, CP ⁽⁵⁾ , CPB ⁽⁵⁾ , IOR, MUL.SU, MUL.UU, SUB, SUBB, SUBBR, SUBR, XOR |
| #Slit6 ⁽¹⁾ | SFTAC |
| #lit8 | MOV.B, CP ⁽⁴⁾ , CPB ⁽⁴⁾ |
| #lit10 | ADD, ADDC, AND, CP, CPB, IOR, RETLW, SUB, SUBB, XOR |
| #Slit10 | MOV |
| #lit14 | DISI, DO ⁽²⁾ , LNK, REPEAT ⁽⁵⁾ |
| #lit15 | DO ⁽³⁾ , REPEAT ⁽⁴⁾ |
| #lit16 | MOV |

- 注** 1: 该操作数或指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
2: 该操作数或指令仅在 dsPIC30F 和 dsPIC33F 器件中可用。
3: 该操作数或指令仅在 dsPIC33E 器件中可用。
4: 该操作数或指令仅在 dsPIC33E 和 PIC24E 器件中可用。
5: 该操作数或指令仅在 dsPIC30F、dsPIC33F、PIC24F 和 PIC24H 器件中可用。

立即数寻址的语法规则要求必须将数值符号（#）放在常量操作数之前且紧靠常量操作数。“#”符号向汇编器表明该量为常量。如果指令中使用的常量超出范围，汇编器将产生一个错误。例 4-7 中给出了几个立即数寻址的示例。

例 4-7：立即数寻址

PWRSAB #1; 进入空闲模式

ADD.B #0x10, W0; 将 0x10 加到 W0（字节模式）

指令执行前:
W0 = 0x12A9
指令执行后:
W0 = 0x12B9

XOR W0, #1, [W1++]; 将 W0 和 0x1 异或
; 结果存入 [W1]
; 指令执行后, 递增 W1

指令执行前:
W0 = 0xFFFF
W1 = 0x0890
数据存储区 0x0890 = 0x0032
指令执行后:
W0 = 0xFFFF
W1 = 0x0892
数据存储区 0x0890 = 0xFFFFE

4.1.5 数据空间寻址模式树

图 4-1 对 PIC24F、PIC24H 和 PIC24E 系列的数据空间寻址模式进行了汇总。

图 4-1：数据空间寻址模式树（PIC24F、PIC24H 和 PIC24E 器件）

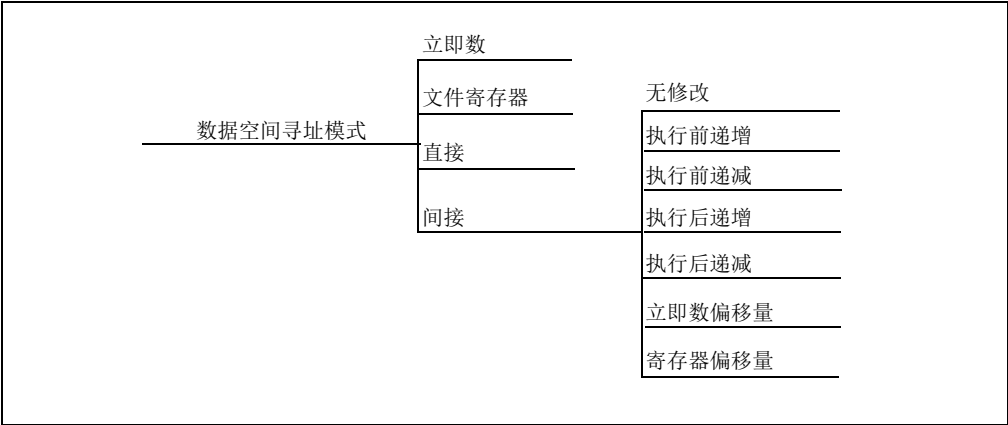
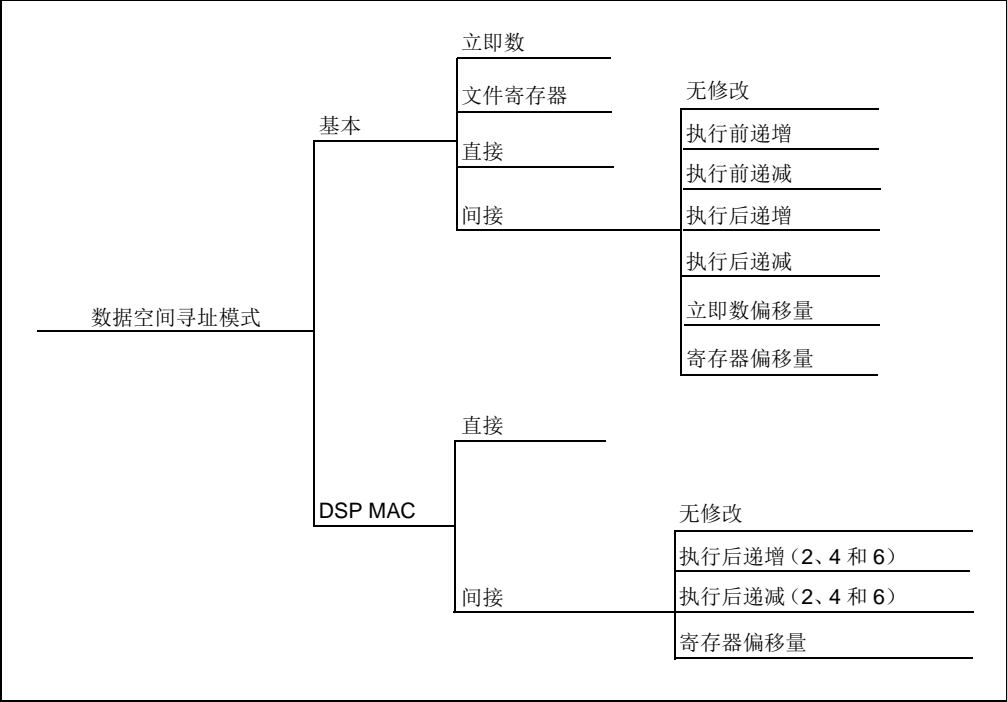


图 4-2 对 dsPIC30F、dsPIC33F 和 dsPIC33E 的数据空间寻址模式进行了汇总。

图 4-2: 数据空间寻址模式树 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)



4.2 程序空间寻址模式

16 位 MCU 和 DSC 器件拥有一个 23 位程序计数器 (PC)。PC 可对 24 位宽的程序存储空间进行寻址以取出将执行的指令。可通过几种不同方式实现对 PC 的装载。对于与表读以及表写指令的字节兼容性，每一个指令字都在程序存储区中占用两个地址单元。这表明程序顺序执行时，PC 递增的方式将为 $PC + 2$ 。

有几种不同的方法可实现以非顺序方式修改 PC 值进行改动，这些改动可以是绝对或是相对的。PC 值的改变可能来自指令编码中的立即数或包含在工作寄存器中的动态值。在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中，当执行 DO 循环时，在 DOEND 地址中的指令执行结束后，PC 将装载存放于 DOSTART 寄存器中的地址。对于异常处理，PC 将装载异常处理程序的入口地址，该地址存放在中断向量表中。在需要时，可用软件堆栈来保存现场，返回到程序流发生变化处的前台进程。

表 4-5 汇总了修改 PC 值的指令。当进行函数调用时，由于 RCALL 仅占用一个字的程序存储空间，因此建议使用 RCALL 而不使用 CALL 指令。

表 4-5: 改变程序流的方法

| 条件 / 指令 | PC 改变 | 软件堆栈使用情况 |
|--|--|----------------------------|
| 顺序执行 | $PC = PC + 2$ | 无 |
| BRA Expr ⁽¹⁾ (无条件转移) | $PC = PC + 2 * \text{Slit16}$ | 无 |
| BRA Condition, Expr ⁽¹⁾ (条件转移) | $PC = PC + 2$ (条件为假) $PC = PC + 2 * \text{Slit16}$ (条件为真) | 无 |
| CALL Expr ⁽¹⁾ (调用子程序) | $PC = \text{lit23}$ | PC + 4 压入堆栈 ⁽²⁾ |
| CALL Wn (间接调用子程序) | $PC = Wn$ | PC + 2 压入堆栈 ⁽²⁾ |
| CALL.L Wn ⁽⁵⁾ (间接长调用子程序) | $PC = \{Wn+1:Wn\}$ | 无 |
| GOTO Expr ⁽¹⁾ (无条件跳转) | $PC = \text{lit23}$ | 无 |
| GOTO Wn (无条件间接跳转) | $PC = Wn$ | 无 |
| GOTO.L Wn ⁽⁵⁾ (无条件间接长跳转) | $PC = \{Wn+1:Wn\}$ | 无 |
| RCALL Expr ⁽¹⁾ (相对调用) | $PC = PC + 2 * \text{Slit16}$ | PC + 2 压入堆栈 ⁽²⁾ |
| RCALL Wn (计算相对调用) | $PC = PC + 2 * Wn$ | PC + 2 压入堆栈 ⁽²⁾ |
| 异常处理 | PC = 异常处理程序入口地址 (从向量表中读出) | PC + 2 压入堆栈 ⁽³⁾ |
| PC = 目标 REPEAT 指令 (REPEAT 循环) | PC 未改动 (如果 REPEAT 激活) | 无 |
| PC = DOEND 地址 ⁽⁴⁾ (DO 循环) | PC = DOSTART (如果 DO 激活) | 无 |

注 1: 对于 BRA、CALL 以及 GOTO 指令，Expr 可以是标号、绝对地址或表达式。Expr 将由链接器解析为一个 16 位或 23 位值 (Slit16 或 lit23)。有关详细信息，请参见第 5 章“指令描述”。

2: 在执行 CALL 或 RCALL 指令后，RETURN 或 RETLW 将把栈顶内容弹回 PC。

3: 在异常处理结束后，RETFIE 将把栈顶内容弹回 PC。

4: 该条件 / 指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

5: 该条件 / 指令仅在 dsPIC33E 和 PIC24E 器件中可用。

4.3 指令停顿

为尽可能缩短数据空间 EA 计算和取操作数的时间，部分 X 数据空间读和写访问采用流水线方式进行。这种流水线方式的一个可能后果是，在使用共用的寄存器进行连续的读写操作可能导致地址寄存器数据的相关性。

“写后读”（RAW）相关性问题发生在跨指令边界时，由硬件检测。RAW 相关性的一个示例是在修改 W5 的写操作结束后，使用 W5 作为地址指针的读操作。在前次写操作结束之前，W5 中的内容将不能作为读操作的有效数据。这个问题可通过对指令执行停顿一个周期来解决，这将允许在下一读操作启动之前结束写操作。

4.3.1 RAW 相关性检测

在指令预译码期间，内核将确定地址寄存器相关性是否即将跨指令边界。停顿检测逻辑将用于当前执行指令的目标 EA 的 W 寄存器（如果有）和用于预取指令源 EA（如果有）的 W 寄存器进行比较。当发现目标寄存器和源寄存器相匹配时，将应用一组规则来判定是否将对指令执行停顿一个周期。表 4-6 列出了导致指令执行停顿的各种 RAW 条件。

表 4-6: RAW 相关性规则（由硬件检测）

| 使用 Wn 的目标地址模式 | 使用 Wn 的源地址模式 | 需要停顿吗? | 示例 (2) (Wn = W2) |
|-----------------|-----------------|--------|--|
| 直接 | 直接 | 无停顿 | ADD.W W0, W1, W2 MOV.W W2, W3 |
| 间接 | 直接 | 无停顿 | ADD.W W0, W1, [W2] MOV.W W2, W3 |
| 间接 | 间接 | 无停顿 | ADD.W W0, W1, [W2] MOV.W [W2], W3 |
| 间接 | 带执行前 / 后修改的间接地址 | 无停顿 | ADD.W W0, W1, [W2] MOV.W [W2++], W3 |
| 带执行前 / 后修改的间接地址 | 直接 | 无停顿 | ADD.W W0, W1, [W2++] MOV.W W2, W3 |
| 直接 | 间接 | 停顿 (1) | ADD.W W0, W1, W2 MOV.W [W2], W3 |
| 直接 | 带执行前 / 后修改的间接地址 | 停顿 (1) | ADD.W W0, W1, W2 MOV.W [W2++], W3 |
| 间接 | 间接 | 停顿 (1) | ADD.W W0, W1, [W2](2) MOV.W [W2], W3(2) |
| 间接 | 带执行前 / 后修改的间接地址 | 停顿 (1) | ADD.W W0, W1, [W2](2) MOV.W [W2++], W3(2) |
| 带执行前 / 后修改的间接地址 | 间接 | 停顿 (1) | ADD.W W0, W1, [W2++] MOV.W [W2], W3 |
| 带执行前 / 后修改的间接地址 | 带执行前 / 后修改的间接地址 | 停顿 (1) | ADD.W W0, W1, [W2++] MOV.W [W2++], W3 |

注 1: 当检测到停顿时，指令执行时间将增加 1 个周期。

2: 对于这些示例，W2 的内容 = W2 的映射地址（0x0004）。

4.3.2 指令停顿和异常

为保持确定的操作，将允许指令停顿发生，即使在停顿后紧接着异常处理。

4.3.3 指令停顿和改变程序流的指令

CALL 和 RCALL 使用 W15 来写堆栈，因此如果下一条指令的读源操作数操作使用 W15 寄存器则可能发生指令停顿。

GOTO、RETFIE 和 RETURN 指令的执行过程中将不会发生指令停顿的现象，因为它们不执行写入工作寄存器的操作。

4.3.4 指令停顿和 DO/REPEAT 循环

如同其他任何指令，运行于 DO 或 REPEAT 循环中的指令也易发生指令停顿的现象。停顿可能发生在循环入口、出口以及在循环处理过程中。

注： DO 循环仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

4.3.5 指令停顿和 PSV

如同其他任何指令，运行于 PSV 地址空间的指令易发生指令停顿的现象。如果紧接 PSV 数据访问之后的指令执行中检测到数据相关性的问题，指令的第二个周期将启动一个停顿。如果紧靠 PSV 数据访问之前的指令执行中检测到数据相关性的问题，前条指令的最后一个周期将启动一个停顿。

注： 有关 RAW 指令停顿更为详细的信息，请参见具体器件的系列参考手册。

4.4 字节操作

由于数据存储区可字节寻址，大多数基本指令可运行在字节或字两种模式。当这些指令运行于字节模式时，以下规则将适用：

- 所有直接工作寄存器访问使用 16 位工作寄存器的低位字节，而高位字节（MSB）将不会被改动
- 所有间接工作寄存器访问使用由存放在工作寄存器中的 16 位地址指定的数据字节
- 所有文件寄存器访问使用由字节地址指定的数据字节
- 状态寄存器内容将被更新以反映字节操作的结果

应注意，数据地址总是以**字节**地址来表示。此外，固有的数据存放格式小尾数法（little-endian），即将低位字节存放在较低地址中，而将高位字节存放在相邻的较高地址中（如图 4-3 所示）。例 4-8 给出了字节传送操作的示例，而例 4-9 给出了字节数学操作的示例。

注： 工作于字节模式的指令必须使用 “.b” 或 “.B” 指令扩展以指定字节指令。例如，以下两条指令为字节清零操作的有效形式：

- CLR.b W0
- CLR.B W0

例 4-8: 字节传送操作示例

```
MOV.B #0x30, W0      ; 将立即数字节 0x30 传送至 W0
指令执行前:
W0 = 0x5555
指令执行后:
W0 = 0x5530

MOV.B 0x1000, W0      ; 将 0x1000 中的字节传送至 W0
指令执行前:
W0 = 0x5555
数据存储区 0x1000 = 0x1234
指令执行后:
W0 = 0x5534
数据存储区 0x1000 = 0x1234

MOV.B W0, 0x1001      ; 将 W0 中字节传送至地址 0x1001
指令执行前:
W0 = 0x1234
数据存储区 0x1000 = 0x5555
指令执行后:
W0 = 0x1234
数据存储区 0x1000 = 0x3455

MOV.B W0, [W1++]      ; 将 W0 中字节传送至 [W1]，指令执行后递增 W1
指令执行前:
W0 = 0x1234
W1 = 0x1001
数据存储区 0x1000 = 0x5555
指令执行后:
W0 = 0x1234
W1 = 0x1002
数据存储区 0x1000 = 0x3455
```


例 4-9: 字节数学操作示例

CLR.B [W6--] ; 字节清零 [W6], 指令执行后递减 W6

指令执行前:
W6 = 0x1001
数据存储区 0x1000 = 0x5555

指令执行后:
W6 = 0x1000
数据存储区 0x1000 = 0x0055

SUB.B W0, #0x10, W1 ; 从 W0 中字节减立即数 0x10
; 并将结果存入 W1

指令执行前:
W0 = 0x1234
W1 = 0xFFFF

指令执行后:
W0 = 0x1234
W1 = 0xFF24

ADD.B W0, W1, [W2++] ; 将 W0 和 W1 进行字节加, 结果存入 [W2]
; 指令执行后, 递增 W2

指令执行前:
W0 = 0x1234
W1 = 0x5678
W2 = 0x1000
数据存储区 0x1000 = 0x5555

指令执行后:
W0 = 0x1234
W1 = 0x5678
W2 = 0x1001
数据存储区 0x1000 = 0x55AC

4.5 字传送操作

即使数据空间采用字节寻址模式，所有以字模式进行的传送操作也必须满足字对齐的原则。这表明对于所有源操作数和目标操作数，最低地址位必须是 0。如果发生字传送操作的一方是奇数地址，那将产生一个地址出错异常。类似地，所有双字传送操作也必须满足字对齐原则。图 4-3 说明了如何实现字节和字在数据存储区的对齐。例 4-10 给出了几种合法的字传送操作示例。

当由于未对齐访问导致异常产生时，将在指令执行后进行异常处理。如果在数据读操作期间发生非法访问，该操作将被允许完成，但是源地址的最低位将被清零以强制字对齐。如果在数据写操作期间发生非法访问，该写操作将被禁止。例 4-11 中包含了几种非法字传送操作。

图 4-3: 存储区中数据对齐方式

| | | | |
|--------|----|----|--------|
| 0x1001 | | b0 | 0x1000 |
| 0x1003 | b1 | | 0x1002 |
| 0x1005 | b3 | b2 | 0x1004 |
| 0x1007 | b5 | b4 | 0x1006 |
| 0x1009 | b7 | b6 | 0x1008 |
| 0x100B | | b8 | 0x100A |

图注:

b0—— 存放于 0x1000 中的字节

b1—— 存放于 0x1003 中的字节

b3:b2—— 存放于 0x1005:1004 中的字（b2 为 LSB）

b7:b4—— 存放于 0x1009:0x1006 中的双字（b4 为 LSB）

b8—— 存放于 0x100A 中的字节

注： 工作于字模式的指令不需要使用指令扩展。但如果需要可对其指定可选的 “.w” 或 “.W” 扩展。例如，以下指令为字清零操作的有效形式：

- CLR W0
- CLR.w W0
- CLR.W W0

例 4-10: 合法的字传送操作

MOV #0x30, W0 ; 将立即数字 0x30 传送至 W0

指令执行前:

W0 = 0x5555

指令执行后:

W0 = 0x0030

MOV 0x1000, W0 ; 将 0x1000 中字传送至 W0

指令执行前:

W0 = 0x5555

数据存储区 0x1000 = 0x1234

指令执行后:

W0 = 0x1234

数据存储区 0x1000 = 0x1234

MOV [W0], [W1++] ; 字传送 [W0] 至 [W1]

; 指令执行后, 递增 W1

指令执行前:

W0 = 0x1234

W1 = 0x1000

数据存储区 0x1000 = 0x5555

数据存储区 0x1234 = 0xAAAA

指令执行后:

W0 = 0x1234

W1 = 0x1002

数据存储区 0x1000 = 0xAAAA

数据存储区 0x1234 = 0xAAAA

例 4-11: 非法的字传送操作

MOV 0x1001, W0 ; 将 0x1001 中的字传送至 W0

指令执行前:

W0 = 0x5555

数据存储区 0x1000 = 0x1234

数据存储区 0x1002 = 0x5678

指令执行后:

W0 = 0x1234

数据存储区 0x1000 = 0x1234

数据存储区 0x1002 = 0x5678

产生地址出错陷阱

(源地址未对齐, 因此执行 MOV)

MOV W0, 0x1001 ; 将 W0 传送至 0x1001

指令执行前:

W0 = 0x1234

数据存储区 0x1000 = 0x5555

数据存储区 0x1002 = 0x6666

指令执行后:

W0 = 0x1234

数据存储区 0x1000 = 0x5555

数据存储区 0x1002 = 0x6666

产生地址出错陷阱

(目标地址未对齐, 因此不执行 MOV)

MOV [W0], [W1++] ; 字传送 [W0] 至 [W1]

; 指令执行后, 递增 W1

指令执行前:

W0 = 0x1235

W1 = 0x1000

数据存储区 0x1000 = 0x1234

数据存储区 0x1234 = 0xAAAA

数据存储区 0x1236 = 0xBBBB

指令执行后:

W0 = 0x1235

W1 = 0x1002

数据存储区 0x1000 = 0xAAAA

数据存储区 0x1234 = 0xAAAA

数据存储区 0x1236 = 0xBBBB

产生地址出错陷阱

(源地址未对齐, 因此执行 MOV)

4.6 使用 10 位立即数操作数

支持字节和字模式的几条指令具有 10 位操作数。对于字节指令，10 位立即数太大而不能使用。因此在字节模式下使用 10 位立即数时，操作数的范围必须减至 8 位，否则汇编器将产生一个错误。表 4-7 说明，在字模式下 10 位立即数的范围是 0:1023，而在字节模式下则为 0:255。

在字节和字模式下使用 10 位立即数的指令为：ADD、ADDC、AND、IOR、RETLW、SUB、SUBB 和 XOR。例 4-12 以 ADD 指令为例说明了如何在字节模式下使用正或负的立即数。

表 4-7: 10 位立即数编码

| 立即数值 | 字模式 kk kkkk kkkk | 字节模式 kkkk kkkk |
|------|---------------------|-------------------|
| 0 | 00 0000 0000 | 0000 0000 |
| 1 | 00 0000 0001 | 0000 0001 |
| 2 | 00 0000 0010 | 0000 0010 |
| 127 | 00 0111 1111 | 0111 1111 |
| 128 | 00 1000 0000 | 1000 0000 |
| 255 | 00 1111 1111 | 1111 1111 |
| 256 | 01 0000 0000 | N/A |
| 512 | 10 0000 0000 | N/A |
| 1023 | 11 1111 1111 | N/A |

例 4-12: 使用 10 位立即数作为字节操作数

| | | |
|-------|------------|------------------------|
| ADD.B | #0x80, W0 | ; 将 128 (或 -128) 加到 W0 |
| ADD.B | #0x380, W0 | ; 出错 ... 字节模式的非法语法 |
| ADD.B | #0xFF, W0 | ; 将 255 (或 -1) 加到 W0 |
| ADD.B | #0x3FF, W0 | ; 出错 ... 字节模式的非法语法 |
| ADD.B | #0xF, W0 | ; 将 15 加到 W0 |
| ADD.B | #0x7F, W0 | ; 将 127 加到 W0 |
| ADD.B | #0x100, W0 | ; 出错 ... 字节模式的非法语法 |

注: 由于字节最高位被置 1，在字节模式下使用大于 127 的立即数在功能上与使用负的二进制补码值等效。当工作于字节模式时，汇编器将接受正或负的立即数值（例如，#-10）。

4.7 软件堆栈指针和帧指针

4.7.1 软件堆栈指针

16 位 MCU 和 DSC 器件都具有一个软件堆栈，有助于函数调用以及异常处理。W15 为默认的堆栈指针（Stack Pointer，SP），在复位之后，它被初始化为 0x0800（对于 PIC24E 和 dsPIC33E 器件，初始化为 0x1000）。这将确保 SP 将指向有效的 RAM 单元并允许异常处理可以使用堆栈，这可能发生在用户软件对 SP 进行设定之前。在初始化时，用户可重新编程设定 SP 为数据空间内的任何地址单元。

SP 总是指向堆栈顶部第一个可供使用的字，并按照地址从低到高的原则对软件堆栈进行填充。SP 在弹出堆栈（读）前递减而在压入堆栈（写）后递增。

使用 PUSH 和 POP 指令对软件堆栈进行操作。PUSH 和 POP 指令等同于将 W15 用作目标指针的 MOV 指令。例如，可通过以下指令将 W0 中内容压入栈顶（Top-of-Stack，TOS）：

```
PUSH W0

该语法等同于：

MOV W0,[W15++]
```

通过以下指令可将 TOS 中内容返回 W0：

```
POP W0

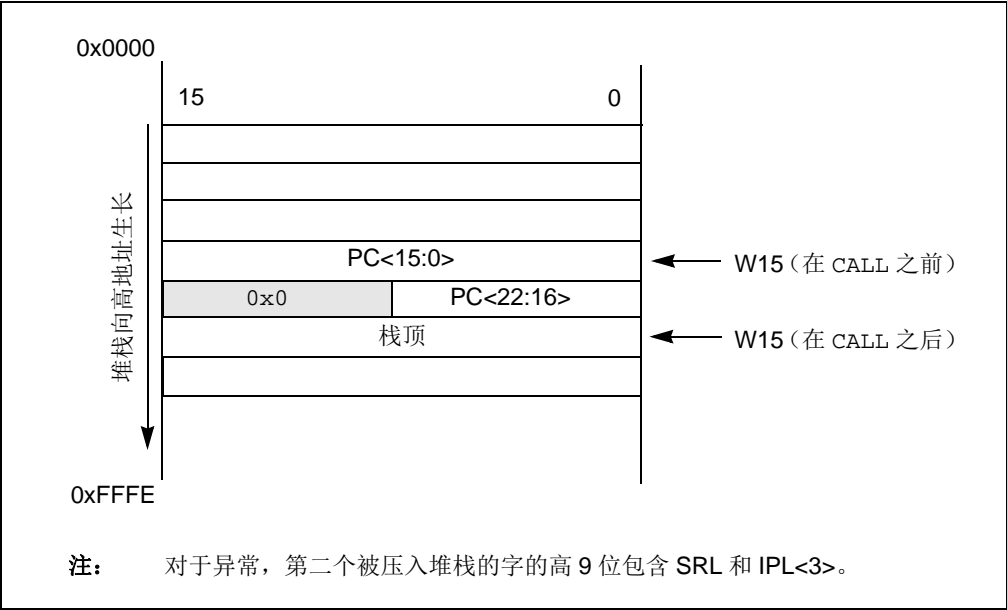
该语法等同于：

MOV [--W15],W0
```

在执行任何 CALL 指令过程中，PC 的内容将被压入堆栈，因此当执行完子程序时程序流将从正确地址继续执行。当将 PC 压入堆栈时，PC<15:0> 被压入第一个可用的堆栈字，然后将 PC<22:16> 压入堆栈。如图 4-4 所示，当 PC<22:16> 被压入堆栈时，在压栈前 PC 的高 7 位将进行零扩展。在异常处理过程中，PC 的高 7 位将与状态寄存器的低位字节（SRL）和 IPL<3>（CORCON<3>）组合起来，这将使得在中断过程中主要的状态寄存器内容和 CPU 中断优先级被自动保存。

注： 为防止未对齐的堆栈访问操作发生，W15<0> 总是清零。

图 4-4: CALL 指令执行时的堆栈操作



4.7.1.1 堆栈指针示例

图 4-5 至图 4-8 说明了例 4-13 的代码段是如何修改软件堆栈的。图 4-5 显示了第一条 PUSH 指令执行前软件堆栈的情况。注意，SP 已被初始化为 0x0800。此外，示例代码将 0x5A5A 和 0x3636 分别装入 W0 和 W1。在图 4-6 中第一次执行压入堆栈的操作，W0 中包含的值被复制到 TOS。W15 被自动更新以指向下一个可用的堆栈地址单元，且新的 TOS 为 0x0802。在图 4-7 中，W1 中内容被压入堆栈，此时新的 TOS 变为 0x0804。在图 4-8 中，堆栈被弹出，将最后一次压入堆栈的值（W1 内容）复制到 W3。在弹出操作过程中，SP 将递减，且在示例代码执行结束时 TOS 最终值为 0x0802。

例 4-13: 堆栈指针的使用

| | | |
|------|-------------|----------------------------|
| MOV | #0x5A5A, W0 | ; 将 0x5A5A 装入 W0 |
| MOV | #0x3636, W1 | ; 将 0x3636 装入 W1 |
| PUSH | W0 | ; 将 W0 中内容压入 TOS (见图 4-5) |
| PUSH | W1 | ; 将 W1 中内容压入 TOS (见图 4-7) |
| POP | W3 | ; 将 TOS 中内容弹出到 W3 (见图 4-8) |

图 4-5: 第一次执行 PUSH 指令之前的堆栈指针

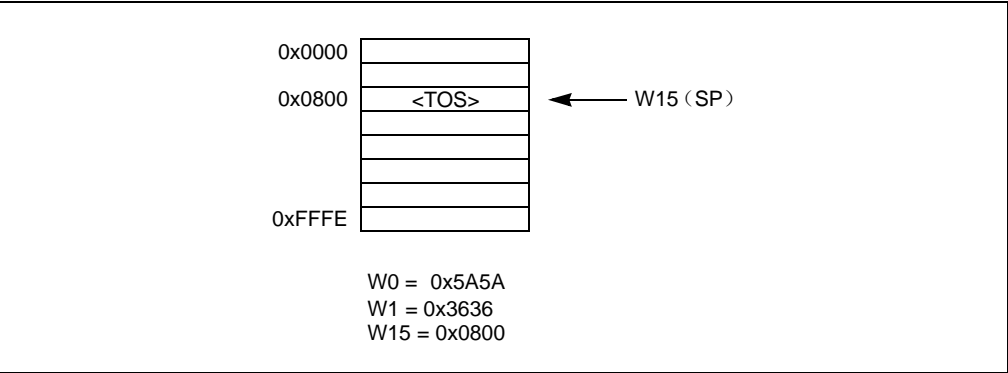


图 4-6: 执行 “PUSH W0” 指令后的堆栈指针

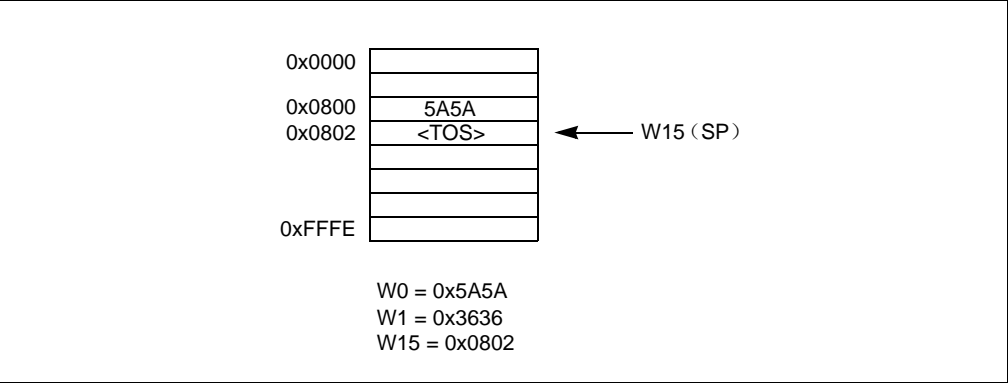


图 4-7: 执行 “PUSH W1” 指令后的堆栈指针

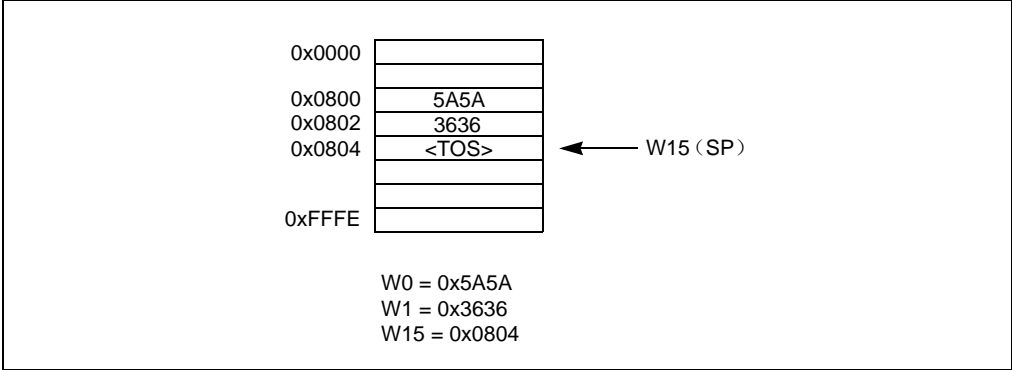
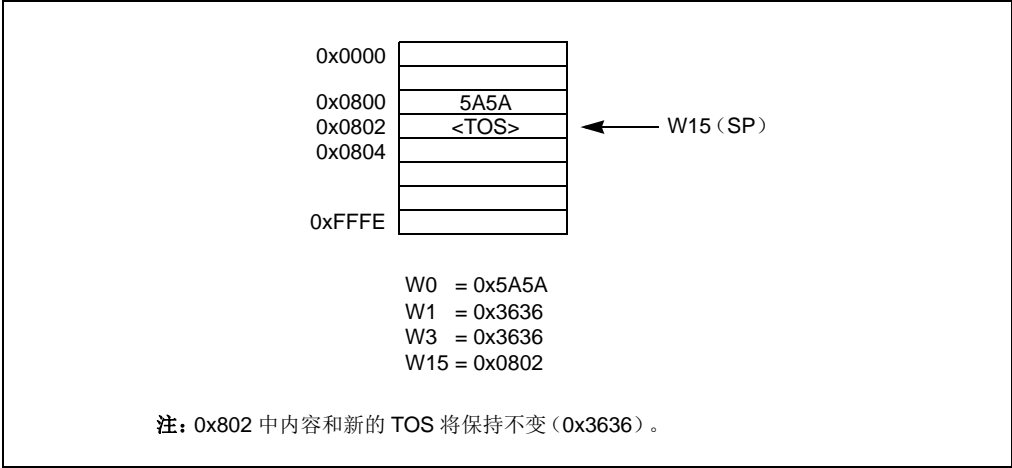


图 4-8: 执行 “POP W3” 指令后的堆栈指针



4.7.2 软件堆栈帧指针

堆栈帧是用户定义的驻留在软件堆栈中的存储区。它用来为函数使用的临时变量分配存储区，可为每个函数创建一个堆栈帧。W14 是默认的堆栈帧指针（Frame Pointer, FP），它在任何复位时被初始化为 0x0000。如果未使用堆栈帧指针，W14 可被当作普通的工作寄存器使用。

分配堆栈帧（LNK）和释放堆栈帧（ULNK）指令提供了堆栈帧功能。LNK 指令用来创建一个堆栈帧，在调用过程中用来调整 SP，这样堆栈可用来存放被调用函数使用的临时变量。在函数执行结束后，ULNK 指令用来移除 LNK 指令创建的堆栈帧。LNK 和 ULNK 指令必须总是一起使用以防止堆栈溢出。

4.7.2.1 堆栈帧指针示例

图 4-9 至图 4-11 说明了在例 4-14 的代码段中是如何创建及移除堆栈帧的。该程序展示了堆栈帧是如何操作的而并不表示编译器生成的代码。图 4-9 显示了程序开始时的堆栈状态，此时所有寄存器尚未被压入堆栈。这里，W15 指向第一个可用的堆栈地址单元（TOS），W14 指向分配给当前正在执行的子程序的堆栈存储区。

在调用子程序“COMPUTE”之前，子程序参数（W0、W1 和 W2）被压入堆栈。在“CALL COMPUTE”指令执行之后，PC 变为“COMPUTE”的地址，且子程序“TASKA”的返回地址被压入堆栈（图 4-10）。子程序“COMPUTE”随后使用“LNK #4”指令将调用子程序的帧指针值压入堆栈，且新的帧指针将被设定指向当前堆栈指针。随后，立即数 4 被加到 W15 中的堆栈指针地址，这将为临时数据保留两个字的存储区（图 4-11）。

在子程序“COMPUTE”中，FP 用来访问子程序参数以及临时（局部）变量。[W14 + n] 将用来对子程序使用的临时变量进行访问，而 [W14 - n] 用来对参数进行访问。在子程序出口，ULNK 指令被用来将帧指针地址复制到堆栈指针，然后将调用子程序的帧指针弹回 W14 寄存器。ULNK 指令将使堆栈返回至图 4-10 中所示的状态。

RETURN 指令将使程序执行返回至调用子程序的代码。调用代码负责将参数从堆栈中移除。RETURN 和 POP 指令将堆栈恢复至图 4-9 中所示的状态。

例 4-14: 帧指针用法

```
TASKA:
    ...
    PUSH    W0          ; 将参数 1 压入堆栈
    PUSH    W1          ; 将参数 2 压入堆栈
    PUSH    W2          ; 将参数 3 压入堆栈
    CALL    COMPUTE     ; 调用 COMPUTE 函数
    POP     W2          ; 弹出参数 3
    POP     W1          ; 弹出参数 2
    POP     W0          ; 弹出参数 1
    ...

COMPUTE:
    LNK     #4          ; FP 入栈，为局部变量分配 4 个字节
    ...
    ULNK                    ; 释放分配的存储区，恢复原先的 FP
    RETURN              ; 返回到 TASKA
```

图 4-9: 在例 4-14 开始时的堆栈情况

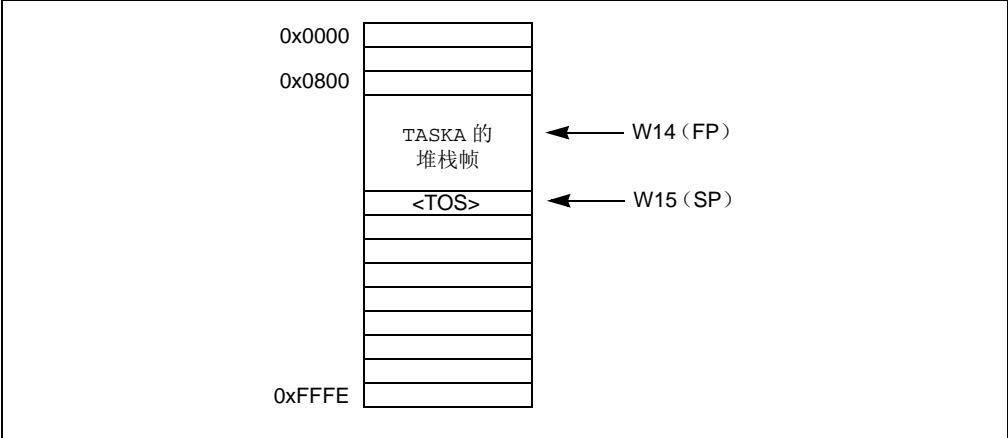


图 4-10: “CALL COMPUTE” 指令执行后的堆栈

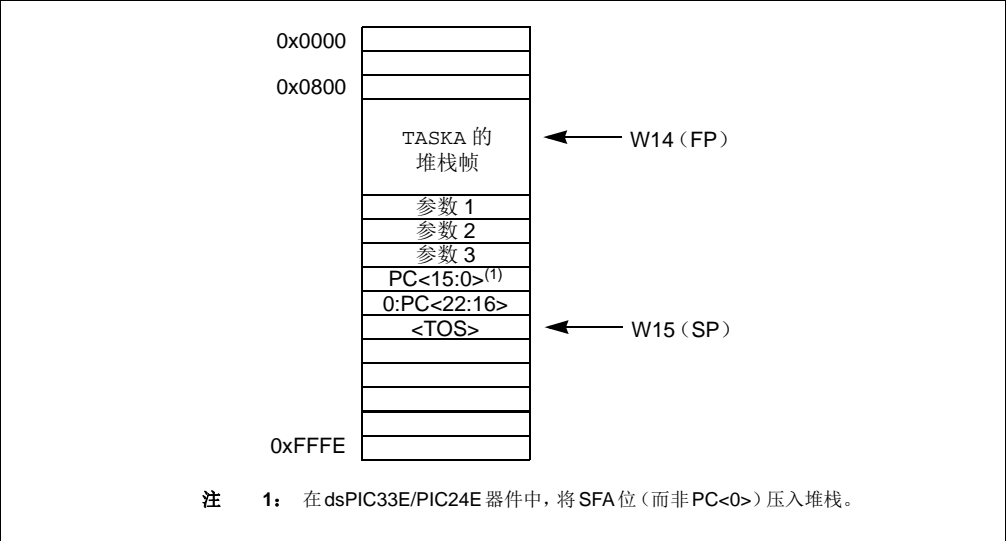
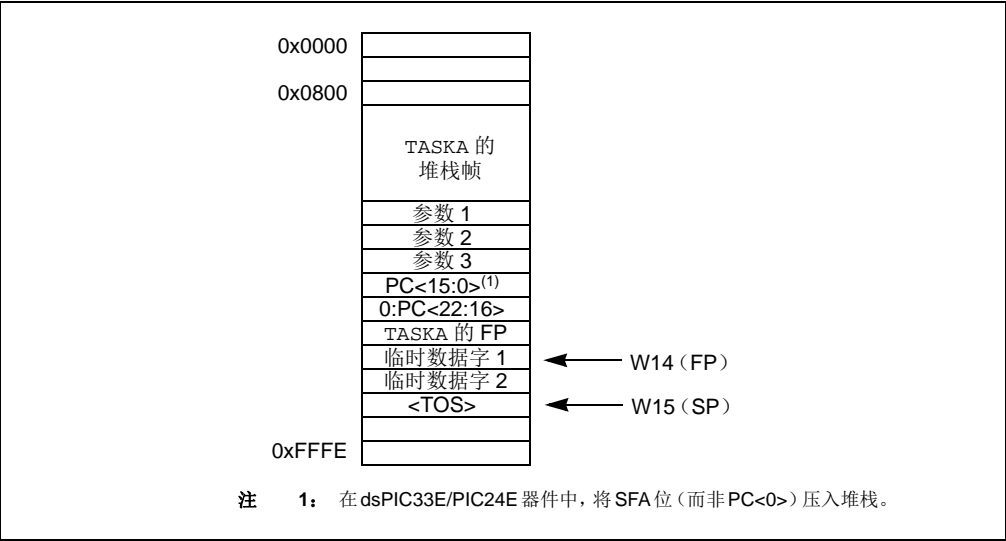


图 4-11: “LNK #4” 指令执行后的堆栈



4.7.3 堆栈指针上溢

SPLIM 是与堆栈指针相关联的堆栈限制寄存器, 在复位时设定为 0x0000。SPLIM 为 16 位寄存器, 但 SPLIM<0> 固定为 0, 这是由于所有堆栈操作必须符合字对齐的原则。

在向 SPLIM 写入字之前, 不使能堆栈上溢检查。在该功能使能之后, 只有器件复位才能将其禁止。所有使用 W15 作为源寄存器或目标寄存器生成的有效地址将与 SPLIM 中的值进行比较。如果有效地址大于 SPLIM 中内容, 将产生堆栈出错陷阱。

如果已使能堆栈上溢检查, 当 W15 有效地址计算越过了数据存储空间末端 (0xFFFF) 时, 也将产生堆栈出错陷阱。

有关堆栈出错陷阱的更多信息, 请参见具体器件的系列参考手册。

4.7.4 堆栈指针下溢

在复位时，堆栈指针初始化为 0x0800（对于 PIC24E 和 dsPIC33E 器件，初始化为 0x1000）。如果堆栈指针地址小于 0x0800（对于 PIC24E 和 dsPIC33E 器件为 0x1000），将会启动一个堆栈出错陷阱。

注： 位于数据存储空间 0x0000 和 0x07FF（对于 PIC24E 和 dsPIC33E 器件为 0x0FFF）之间的地址单元通常保留用于内核和外设特殊功能寄存器（Special Function Register，SFR）。

4.7.5 堆栈帧激活（SFA）控制（dsPIC33E 和 PIC24E 器件）

W15 永远不会受分页影响，因此被限制在地址范围 0x000000 至 0x00FFFF。但对于任何用户软件函数，当堆栈帧指针（W14）所寻址的堆栈帧激活时（即，执行 LNK 指令之后），W14 将仅用于该函数。因此，用户需要能够在用作通用 W 寄存器和用作堆栈帧指针之间动态地切换 W14。SFA 状态位（CORCON<2>）用于实现该功能，同时不会增加任何软件开销。

当 SFA 位清零时，W14 可与任意页寄存器一起使用。当 SFA 置 1 时，W14 不会受分页影响，将被锁定到与 W15 相同的地址范围（0x000000 至 0x00FFFF）。SFA 寄存器锁定操作如下：

- LNK 指令会将 SFA 置 1（并创建堆栈帧）
- ULNK 指令会将 SFA 清零（并删除堆栈帧）
- CALL、CALL.L 和 RCALL 指令还会将 SFA 位压入堆栈（将它放入已入栈 PC 的 LSb），并在堆栈操作完成之后清零 SFA 位。被调用过程现在可以自由地使用 W14 作为通用寄存器，也可以通过使用 LNK 指令来创建另一个堆栈帧。
- RETURN、RETLW 和 RETFIE 指令全部都会从先前的入栈值中恢复 SFA 位

SFA 位是只读位。它只能通过执行 LNK 指令置 1，通过执行 ULNK、CALL、CALL.L 和 RCALL 指令清零。

注： 在 dsPIC33E 和 PIC24E 器件中，会将 SFA 位压入堆栈，而不是 PC<0>。

4.8 条件转移指令

条件转移指令用于根据状态寄存器的内容对程序流进行控制。这些指令通常与比较类指令联合使用，但可在任何改变状态寄存器内容的操作之后有效使用这些指令。

比较指令 CP、CP0 和 CPB 执行减法操作（被减数 - 减数），但实际上并不保存减法操作的结果。而是仅对状态寄存器中的标志位进行更新，因此随后的条件转移指令可通过对更新后的状态寄存器内容进行测试以对程序流进行改变。如果状态寄存器的结果测试为真，将进行转移。如果状态寄存器的结果测试为假，将不转移。

表 4-8 列出了 dsPIC30F、dsPIC33F 和 dsPIC33E 器件所支持的条件转移指令。该表指明了状态寄存器中何种条件为真将触发转移操作。在某些情况下，指令只对单个状态位进行测试（如在 BRA C 指令中），而在其他情况下，则会执行复杂的逻辑运算（如在 BRA GT 指令中）。对于 dsPIC30F、dsPIC33F 和 dsPIC33E 器件，值得注意的是，通过 OA、OB、SA 和 SB 条件助记符，DSP 算法支持有符号和无符号的条件测试。

表 4-8: 条件转移指令

| 条件助记符 (1) | 说明 | 状态测试 |
|--------------------|----------|---|
| C | 进位（无借位） | C |
| GE | 有符号大于或等于 | $(\overline{N} \& \overline{OV}) \parallel (N \& OV)$ |
| GEU ⁽²⁾ | 无符号大于或等于 | C |
| GT | 有符号大于 | $(\overline{Z} \& \overline{N} \& \overline{OV}) \parallel (\overline{Z} \& N \& OV)$ |
| GTU | 无符号大于 | $C \& \overline{Z}$ |
| LE | 有符号小于或等于 | $Z \parallel (\overline{N} \& \overline{OV}) \parallel (N \& \overline{OV})$ |
| LEU | 无符号小于或等于 | $\overline{C} \parallel Z$ |
| LT | 有符号小于 | $(\overline{N} \& \overline{OV}) \parallel (N \& \overline{OV})$ |
| LTU ⁽³⁾ | 无符号小于 | C |
| N | 负 | N |
| NC | 无进位（有借位） | C |
| NN | 非负 | N |
| NOV | 无溢出 | OV |
| NZ | 非零 | Z |
| OA ⁽⁴⁾ | 累加器 A 溢出 | OA |
| OB ⁽⁴⁾ | 累加器 B 溢出 | OB |
| OV | 溢出 | OV |
| SA ⁽⁴⁾ | 累加器 A 饱和 | SA |
| SB ⁽⁴⁾ | 累加器 B 饱和 | SB |
| Z | 零 | Z |

- 注
- 1: 指令形式为: BRA 助记符, Expr。
 - 2: GEU 等同于 C 且反汇编为 BRA C, Expr。
 - 3: LTU 等同于 NC 且反汇编为 BRA NC, Expr。
 - 4: 该条件仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

注: 比较和跳过指令（CPBEQ、CPBGT、CPBLT、CPBNE、CPSEQ、CPSGT、CPSLT 和 CPSNE）不改变状态寄存器内容。

4.9 Z 状态位

Z 状态位是一个特殊的零状态位，有助于实现扩展精度的算术运算。除那些使用进位 / 借位输入的指令（ADDC、CPB、SUBB 和 SUBBR）外，在所有指令中 Z 位都用作普通的零标志位。对于 ADDC、CPB、SUBB 和 SUBBR 指令，Z 位只能被清零而不能被置 1。如果其中一条指令的结果是非零，无论后续 ADDC、CPB、SUBB 或 SUBBR 指令操作结果为何，Z 位将被清零并将保持该状态。这将允许使用 Z 位对一系列扩展精度运算的结果进行简便的零状态检查。

对多精度数据进行操作的一个指令序列（该序列的起始指令无进位 / 借位输入）将自动对连续的零状态测试结果进行逻辑与操作。只有所有结果必须为零，Z 标志才会在操作序列结束时保持置 1。如果 ADDC、CPB、SUBB 或 SUBBR 指令的执行结果为非零，Z 位将被清零且在所有后续的 ADDC、CPB、SUBB 或 SUBBR 指令执行过程中都将保持清零状态。例 4-15 说明了 32 位加法中 Z 位是如何操作的。例中将显示在使用一个 ADD/ADDC 指令序列实现 32 位加法的过程中将会对 Z 位产生何种影响。第一个示例产生了高位字为零的结果，而第二个示例产生了低位字和高位字都为零的结果。

例 4-15: 32 位加法的 Z 状态位操作

```
; 将两个双字 (W0:W1 和 W2:W3) 相加
; 结果存入 W5:W4
ADD    W0, W2, W4      ; LSWord 相加并将结果存入 W4
ADDC   W1, W3, W5      ; MSWord 相加并将结果存入 W5
```

在 32 位加法执行前（结果的高位字为零）：

```
W0 = 0x2342
W1 = 0xFFFF0
W2 = 0x39AA
W3 = 0x0010
W4 = 0x0000
W5 = 0x0000
SR = 0x0000
```

在 32 位加法执行后：

```
W0 = 0x2342
W1 = 0xFFFF0
W2 = 0x39AA
W3 = 0x0010
W4 = 0x5CEC
W5 = 0x0000
SR = 0x0201 (DC,C=1)
```

在 32 位加法执行前（结果的低位字和高位字都为零）：

```
W0 = 0xB76E
W1 = 0xFB7B
W2 = 0x4892
W3 = 0x0484
W4 = 0x0000
W5 = 0x0000
SR = 0x0000
```

在 32 位加法执行后：

```
W0 = 0xB76E
W1 = 0xFB7B
W2 = 0x4892
W3 = 0x0485
W4 = 0x0000
W5 = 0x0000
SR = 0x0103 (DC,Z,C=1)
```

4.10 规定的工作寄存器用法

16 位 MCU 和 DSC 器件的 16 个工作寄存器组成的大寄存器阵列有助于高效的代码生成和算法实现。为使指令集能够实现高级功能、稳定的运行环境以及同早期 Microchip 处理器内核的向下兼容性，一些工作寄存器具有预先指定的用途。表 4-9 对这些工作寄存器的用途进行了汇总。对于 dsPIC30F、dsPIC33F 和 dsPIC33E，在后续的第 4.10.1 节“隐含的 DSP 操作数（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”至第 4.10.3 节“与 PIC® 单片机的兼容性”中提供了更多详细信息。

表 4-9: 特殊工作寄存器用途

| 寄存器 | 特殊用途 |
|-----|--------------------------------|
| W0 | 默认 WREG，除法商 |
| W1 | 除法余数 |
| W2 | “MUL f”乘积的低位字 |
| W3 | “MUL f”乘积的高位字 |
| W4 | MAC 操作数 ⁽¹⁾ |
| W5 | MAC 操作数 ⁽¹⁾ |
| W6 | MAC 操作数 ⁽¹⁾ |
| W7 | MAC 操作数 ⁽¹⁾ |
| W8 | MAC 预取地址（X 存储区） ⁽¹⁾ |
| W9 | MAC 预取地址（X 存储区） ⁽¹⁾ |
| W10 | MAC 预取地址（Y 存储区） ⁽¹⁾ |
| W11 | MAC 预取地址（Y 存储区） ⁽¹⁾ |
| W12 | MAC 预取偏移量 ⁽¹⁾ |
| W13 | MAC 回写目标地址 ⁽¹⁾ |
| W14 | 帧指针 |
| W15 | 堆栈指针 |

注 1: 该用途仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

4.10.1 隐含的 DSP 操作数（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

一些工作寄存器具有预先指定的功能，这有助于实现指令编码以及保持 DSP 类指令的统一性。所有具备预取能力的 DSP 类指令必须遵守以下 10 个寄存器指定用途的规定：

- W4-W7 用于算术操作数
- W8-W11 用于预取地址（指针）
- W12 用于预取寄存器偏移量
- W13 用于累加器回写目标地址

这些限制仅适用于 DSP MAC 类指令，这类指令使用工作寄存器且具有预取能力（如第 4.15 节“DSP 累加器类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所介绍）。这些指令包括 CLR、ED、EDAC、MAC、MOVSAC、MPY、MPY.N 和 MSC。

在 dsPIC33E 器件中，支持混合符号 DSP 乘法运算，而无需动态修改 US<1:0> 位。在该模式下（US<1:0> = 10），基于操作数所使用的寄存器将每个输入操作数视为有符号或无符号。W4 和 W6 总是为无符号操作数，而 W5 和 W7 总是为有符号操作数。该特性可用于高效执行扩展精度 DSP 乘法。

DSP 累加器类指令（如第 4.15 节“DSP 累加器类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所介绍）无须遵守表 4-9 中列出的工作寄存器用途规定且可在需要时不受限制地使用任何工作寄存器。

4.10.2 隐含的帧指针和堆栈指针

为满足软件堆栈使用的需要，W14 是隐含的帧指针（用于 LNK 和 ULNK 指令），而 W15 是隐含的堆栈指针（用于 CALL、LNK、POP、PUSH、RCALL、RETFIE、RETLW、RETURN、TRAP 和 ULNK 指令）。尽管 W14 和 W15 具有隐含的用途，其仍可在任何指令中用作一般的操作数，但第 4.10.1 节“隐含的 DSP 操作数（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所列出的情况除外。如果 W14 和 W15 必须被用作其他用途（我们强烈建议仍将其保留作为帧指针和堆栈指针），用户必须特别注意防止运行时环境被破坏。

4.10.3 与 PIC® 单片机的兼容性

4.10.3.1 默认工作寄存器 WREG

为使 Microchip 8 位 PIC MCU 系列的用户能够方便地移植到 16 位 MCU 和 DSC 器件，该款器件的指令集尽可能保持了与 PIC MCU 指令集的一致性。16 位 MCU 和 DSC 与 8 位 PIC MCU 处理器的一个主要不同之处在于提供的工作寄存器数量。8 位 PIC MCU 系列仅提供一个 8 位工作寄存器，而 16 位 MCU 和 DSC 系列则提供 16 个 16 位工作寄存器。为与 8 位 PIC MCU 单个工作寄存器的架构相适应，16 位 MCU 和 DSC 器件的指令集为所有传统的文件寄存器指令指定了一个工作寄存器作为默认工作寄存器。默认的工作寄存器设定为 W0，该寄存器为所有采用文件寄存器寻址模式的指令所使用。

此外，16 位 MCU 和 DSC 器件的汇编器用来指定默认工作寄存器的语法与 8 位 PIC MCU 汇编器相似。如第 5 章“指令描述”中所介绍，“WREG”必须用于指定默认工作寄存器。例 4-16 给出了几种使用 WREG 的指令。

例 4-16: 使用默认工作寄存器 WREG

| | | |
|-------|--------------|----------------------------------|
| ADD | RAM100 | ; 将 RAM100 和 WREG 相加，结果存入 RAM100 |
| ASR | RAM100, WREG | ; 右移 RAM100，结果存入 WREG |
| CLR.B | WREG | ; 将 WREG 低位字节清零 |
| DEC | RAM100, WREG | ; 将 RAM100 递减，结果存入 WREG |
| MOV | WREG, RAM100 | ; 将 WREG 中内容传送至 RAM100 |
| SETM | WREG | ; 将 WREG 中所有位置 1 |
| XOR | RAM100 | ; 将 RAM100 和 WREG 异或，结果存入 RAM100 |

4.10.3.2 PRODH:PRODL 寄存器对

Microchip 8 位 PIC MCU 与 16 位 MCU 和 DSC 架构之间的另一个显著差异是乘法器。一些 PIC MCU 系列支持一个 8 位 x 8 位乘法器，将乘积存放在 PRODH:PRODL 寄存器对中。16 位 MCU 和 DSC 器件具有一个 17 位 x 17 位乘法器，可将结果存放在任何两个连续的工作寄存器中（从偶数编号寄存器开始存放），或一个累加器中。

尽管存在这一架构上的差异，16 位 MCU 和 DSC 器件仍然通过“MUL{.B} f”指令（在第 303 页中介绍）支持传统的文件寄存器乘法指令（MULWF）。通过将 PRODH:PRODL 寄存器映射到工作寄存器对 W3:W2 可以实现对传统 MULWF 指令的支持。这表明当“MUL{.B} f”工作于字模式时，乘法操作将产生一个 32 位乘积并存入 W3:W2 工作寄存器对，其中 W3 用于存放乘积的高位字而 W2 则用于存放低位字。当“MUL{.B} f”工作于字节模式时，16 位乘积存入 W2 而 W3 将不受影响。该指令的示例如例 4-17 所示。

例 4-17: 无符号 f 和 WREG 相乘（传统 MULWF 指令）

| |
|---|
| <pre>MUL.B 0x100 ; (0x100)*WREG (字节模式), 结果存入 W2</pre> |
| 指令执行前: |
| <pre>W0 (WREG) = 0x7705 W2 = 0x1235 W3 = 0x1000 数据存储区 0x0100 = 0x1255</pre> |
| 指令执行后: |
| <pre>W0 (WREG) = 0x7705 W2 = 0x01A9 W3 = 0x1000 数据存储区 0x0100 = 0x1255</pre> |
| <pre>MUL 0x100 ; (0x100)*WREG (字模式), 结果存入 W3:W2</pre> |
| 指令执行前: |
| <pre>W0 (WREG) = 0x7705 W2 = 0x1235 W3 = 0x1000 数据存储区 0x0100 = 0x1255</pre> |
| 指令执行后: |
| <pre>W0 (WREG) = 0x7705 W2 = 0xDEA9 W3 = 0x0885 数据存储区 0x0100 = 0x1255</pre> |

4.10.3.3 使用 WREG 传送数据

“MOV{.B} f {,WREG}”指令（在第 279 页中介绍）和“MOV{.B} WREG, f”指令（在第 280 页中介绍）允许在文件寄存器和 WREG（工作寄存器 W0）之间进行字节或字数据的传送。这些指令提供了与 Microchip PIC MCU 传统指令 MOVF 和 MOVWF 指令等同的功能。

在所有 MOV 指令中，只有“MOV{.B} f {,WREG}”和“MOV{.B} WREG, f”指令支持文件寄存器之间字节数据的传送。例 4-18 给出了几种使用 WREG 的 MOV 指令示例。

注： 当在文件寄存器和工作寄存器阵列之间进行数据字的传送时，“MOV Wns, f”和“MOV f, Wnd”指令允许任何工作寄存器（W0:W15）用作源寄存器或目标寄存器，而不只是 WREG 寄存器。

例 4-18: 使用 WREG 传送数据

| |
|--|
| <pre>MOV.B 0x1001, WREG ; 将存放在地址单元 0x1001 中的字节传送至 W0</pre> |
| <pre>MOV 0x1000, WREG ; 将存放在地址单元 0x1000 中的字传送至 W0</pre> |
| <pre>MOV.B WREG, TBLPAG ; 将存放在 W0 中的字节传送至 TBLPAG 寄存器</pre> |
| <pre>MOV WREG, 0x804 ; 将存放在 W0 中的字传送至地址单元 0x804</pre> |

4.11 DSP 数据格式（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

4.11.1 整数和小数数据

dsPIC30F、dsPIC33F 和 dsPIC33E 器件支持整数和小数数据类型。整型数据固有表示形式为有符号的二进制补码值，其中最高位定义为符号位。通常来说，N 位二进制补码整数的范围为 -2^{N-1} 至 $2^{N-1} - 1$ 。对于 16 位整数，数据范围为 -32768 (0x8000) 至 32767 (0x7FFF)，包括 0。对于 32 位整数，数据范围为 -2,147,483,648 (0x8000 0000) 至 2,147,483,647 (0x7FFF FFFF)。

小数数据表示为二进制补码数，其中最高位定义为符号位，小数点隐含于符号位之后。这种格式通常被称为 1.15（或 Q15）格式，其中 1 是用来表示数据的整数部分的位数，而 15 是用来表示小数部分的位数。采用这种隐含小数点的 N 位二进制补码表示的小数范围为 -1.0 至 $(1 - 2^{1-N})$ 。对于 16 位小数，1.15 数据格式的范围为 -1.0 (0x8000) 至 0.999969482 (0x7FFF)，包括 0.0 且精度为 3.05176×10^{-5} 。在普通饱和模式下，32 位累加器使用 1.31 数据格式，可将精度提高至 4.6566×10^{-10} 。

通过使用累加器最高字节寄存器 (ACCxU) 的 8 个位作为警戒位扩展了累加器的动态范围。如果存放在累加器中的值发生溢出超出了 32 位的限制，此时将借助警戒位实现 DSP 算法。当 ACCSAT 位 (CORCON<4>) 置 1 时，将使能该模式并将累加器扩展为 40 位。当禁止累加器饱和时也可使用警戒位。此时，累加器可支持的整数范围为 -5.498×10^{11} (0x80 0000 0000) 至 5.498×10^{11} (0x7F FFFF FFFF)。在小数模式下，累加器的警戒位不会改变小数点的位置且 40 位累加器使用 9.31 小数格式。注意，所有小数运算的结果都存入 40 位累加器，并采用 1.31 小数点格式进行对齐。在整数模式下，警戒位仅增加了累加器的动态范围。9.31 小数格式的范围为 -256.0 (0x80 0000 0000) 至 $(256.0 - 4.65661 \times 10^{-10})$ (0x7F FFFF FFFF)。表 4-10 指明了对于 16 位、32 位和 40 位寄存器，dsPIC30F/33F/33E 器件中整数和小数的范围和精度。

应注意，除 DSP 乘法操作之外，ALU 对于整数和小数数据的操作是相同的。也就是说，两个整数相加将产生与两个小数相加相同的结果（二进制数）。唯一的差别在于用户如何对该结果进行解释。然而，DSP 操作执行的乘法是不同的。在这些指令中，数据格式的选择是通过设定 IF 位 (CORCON<0>) 来进行的。因此必须对其进行设定，将其设定为 0 选择小数模式，设定为 1 选择整数模式。因为 dsPIC30F/33F/33E 小数使用隐含的小数点，因此需要进行上述设定。在整数模式下，两个 16 位整数相乘将产生一个 32 位整数结果。然而，两个 1.15 格式的值相乘将产生一个 2.30 格式的结果。由于 dsPIC30F、dsPIC33F 和 dsPIC33E 器件的累加器采用 1.31 数据格式，小数模式的 DSP 乘法执行过程也包括一个左移一位的操作以保持小数点能够正确对齐。该特点将使 DSP 乘法器的精度降低至 2^{-30} ，但对于计算将不会产生其他影响（例如， $0.5 \times 0.5 = 0.25$ ）。

表 4-10: dsPIC30F/33F/33E 的数据范围

| 寄存器宽度 | 整数范围 | 小数范围 | 小数精度 |
|-------|------------------------------------|------------------------------|-------------------------|
| 16 位 | -32768 至 32767 | -1.0 至 $(1.0 - 2^{-15})$ | 3.052×10^{-5} |
| 32 位 | -2,147,483,648 至 2,147,483,647 | -1.0 至 $(1.0 - 2^{-31})$ | 4.657×10^{-10} |
| 40 位 | -549,755,813,888 至 549,755,813,887 | -256.0 至 $(256.0 - 2^{-31})$ | 4.657×10^{-10} |

4.11.2 整数和小数数据的表示

掌握 dsPIC30F、dsPIC33F 和 dsPIC33E 中整数和小数数据的表示方法是利用这些器件实现控制应用的基础。无论整数还是小数数据都将最高位作为符号位，且随着数据位的位置向最低位递进一位，其二进制指数将递减 1。N 位整数的二进制指数从 (N-1) 开始（最高位），最后为 0（最低位）。对于 N 位小数，二进制指数从 0 开始（最高位），最后为 (1-N)（于最低位）。图 4-12 和图 4-13 分别显示了正数和负数的表示方法。

整数和小数表示法之间的转换可通过使用简单的除法和乘法来实现。将整数值除以 2^{N-1} 可实现从 N 位整数至小数的转换。类似地，将小数值乘以 2^{N-1} 可实现从 N 位小数至整数的转换。

图 4-12: 0x4001 的不同表示

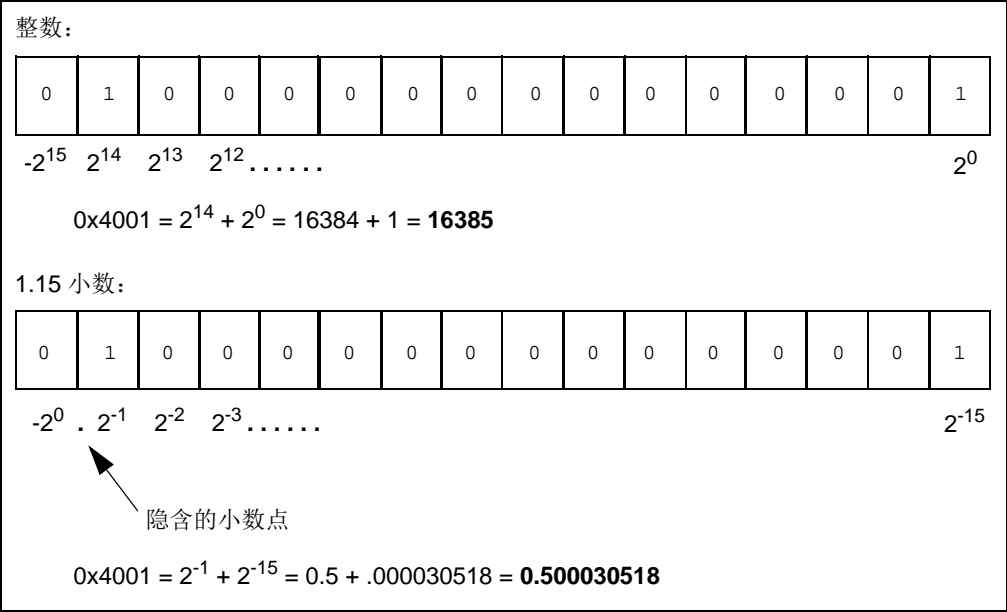
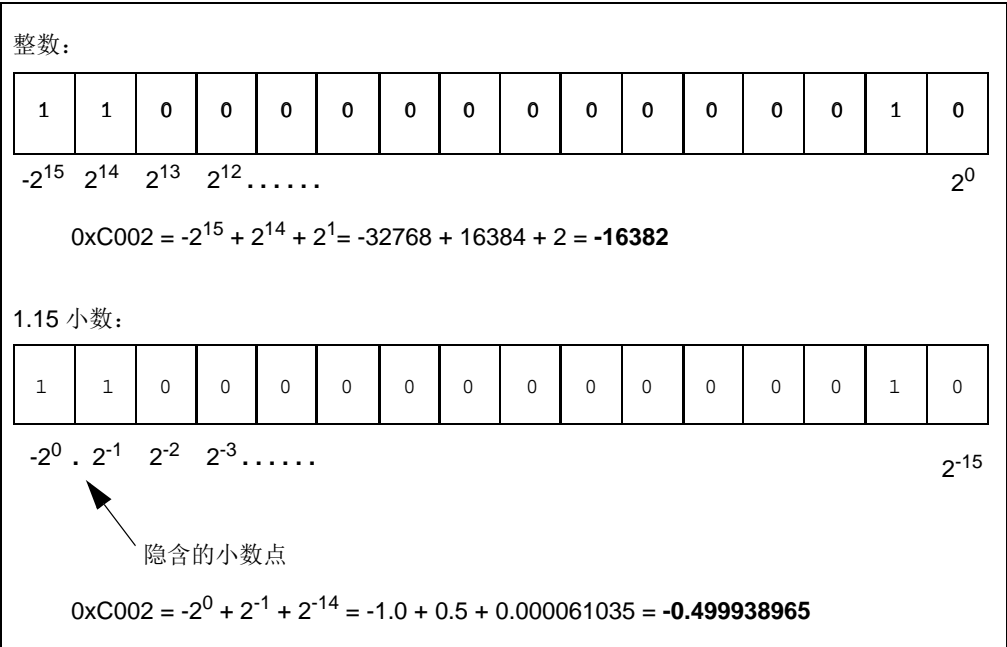


图 4-13: 0xC002 的不同表示



4.12 累加器的使用（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

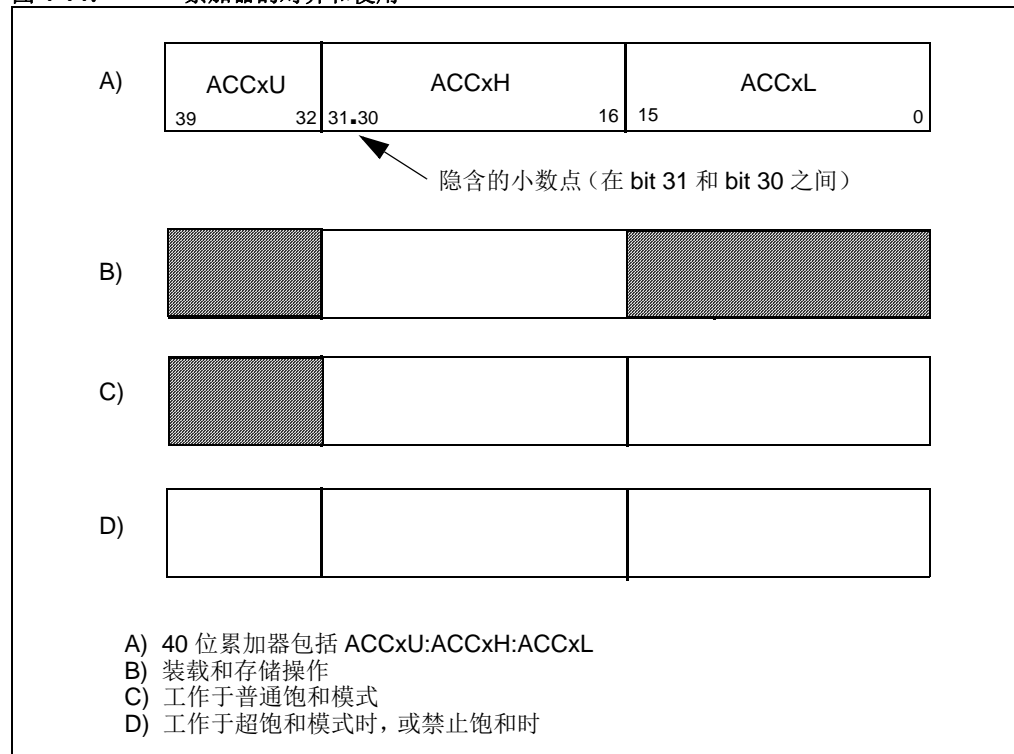
DSP 类指令使用累加器 A 和 B 来执行数学和移位操作。由于累加器为 40 位宽而 X 和 Y 数据宽度只有 16 位，因此用户必须了解累加器执行装载和存储操作的方法。

图 4-14 中 A 项显示每一个 40 位累加器（ACCA 和 ACCB）包含一个 8 位最高位寄存器（ACCxU），一个 16 位高位寄存器（ACCxH）和一个 16 位低位寄存器（ACCxL）。为满足总线对齐要求并提供实现 1.31 数学格式的能力，将 ACCxH 用作累加器装载操作的目标寄存器（使用 LAC 指令），以及累加器存储操作的源寄存器（使用 SAC.R 指令）。图 4-14 中的 B 项显示了该情形，其中累加器的最高字节和低位字部分采用阴影表示。实际上在累加器装载过程中，将对 ACCxL 进行零回填且对 ACCxU 进行符号扩展以表示装入 ACCxH 中值的符号。

当使能普通饱和模式（31 位）时，DSP 操作（例如 ADD、MAC 和 MSC 等）仅利用 ACCxH:ACCxL（如图 4-14 中 C 项所示）而 ACCxU 仅用来保持存入 ACCxH:ACCxL 中值的符号。例如，当执行 MPY 指令时，结果将存入 ACCxH:ACCxL，而结果的符号通过 ACCxU 进行扩展。

当使能超饱和模式时，或禁止饱和时，可利用累加器中的所有寄存器（如图 4-14 中 D 项所示）且 DSP 操作结果将存入 ACCxU:ACCxH:ACCxL。如第 4.11.1 节“整数和小数数据”中所介绍，ACCxU 的作用是使得累加器的动态范围得到扩展。有关普通和超饱和模式下累加器中存储值范围的介绍，请参见表 4-10。

图 4-14: 累加器的对齐和使用



4.13 累加器访问（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

同任何其他特殊功能寄存器一样，构成累加器 A 和累加器 B 的 6 个寄存器皆为存储器映射的。此特点使其支持文件寄存器寻址和间接寻址模式，任何支持此两种寻址模式之一的指令均可对其进行访问。然而，建议使用 DSP 类指令 LAC、SAC 和 SAC.R 对累加器进行装载和存储操作，因为这些指令具备符号扩展、移位以及舍入能力。LAC、SAC 和 SAC.R 指令的细节将在第 5 章“指令描述”给出。

注 1: 为方便起见，将 ACCAU 和 ACCBU 符号扩展至 16 位。这为通过字节模式或字模式访问这些寄存器提供了灵活性（当使用文件寄存器寻址或间接寻址模式时）。
2: OA、OB、SA 或 SB 位不能通过使用 MOV 指令将溢出值写入存储器映射的累加器而置 1，因为这些状态位仅受 DSP 操作影响。

4.14 DSP MAC 类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DSP 乘 - 累加（MAC）操作是一组最高效利用 dsPIC30F、dsPIC33F 和 dsPIC33E 架构的特殊指令。表 4-11 中所示的 DSP MAC 类指令使用 CPU 内核的 X 和 Y 数据线，这将使得这些指令可在一个周期内执行以下操作：

- 使用预取工作寄存器两次读数据存储区（MAC 预取）
- 对预取工作寄存器进行两次更新（MAC 预取寄存器更新）
- 使用一个累加器执行一次数学运算（MAC 操作）

此外，10 条 DSP MAC 类指令中的 4 条也能对一个累加器执行操作，并存储另一个累加器舍入后的内容。该功能称为累加器回写（Write Back，WB），可为软件开发人员提供较高的灵活性。例如，累加器回写可用于同时运行两个算法，或高效处理复数以及用于其他的用途。

表 4-11: DSP MAC 类指令

| 指令 | 说明 | 具有累加器回写功能？ |
|--------|----------------|------------|
| CLR | 清零累加器 | 有 |
| ED | 求取欧几里德距离（无累加） | 无 |
| EDAC | 求取欧几里德距离 | 无 |
| MAC | 相乘并累加 | 有 |
| MAC | 平方并累加 | 无 |
| MOVSAC | X 总线和 Y 总线数据传送 | 有 |
| MPY | 相乘，结果存入累加器 | 无 |
| MPY | 求平方，结果存入累加器 | 无 |
| MPY.N | 相乘取反，结果存入累加器 | 无 |
| MSC | 相乘并从累加器中减去乘积 | 有 |

4.14.1 MAC 预取

预取（或读数据）是通过存放在工作寄存器中的有效地址来实现的。对数据存储区的两次预取操作必须使用表 4-9 中所示的工作寄存器分配进行指定。其中一次读必须使用 W8 或 W9 对 X 数据总线操作，另一次读必须使用 W10 或 W11 对 Y 数据总线操作。用于两种预取操作的允许目标寄存器是 W4-W7。

如表 4-3 中所示，MAC 类指令使用一种特殊的寻址模式。该模式为寄存器偏移量寻址模式，在寻址过程中使用 W12 寄存器。在该模式中，预取操作是通过使用指定工作寄存器中的有效地址加上存放在 W12 中的 16 位有符号值来实现的。寄存器偏移量寻址只可通过 W9 寄存器在 X 数据存储空间中使用时以及 W11 寄存器在 Y 数据存储空间中使用时。

4.14.2 MAC 预取寄存器更新

在 MAC 预取执行之后，可对存放在每一个预取工作寄存器中的有效地址进行更改。这个特点可对顺序存储在 X 和 Y 存储区中的数据实现高效的单周期处理。由于所有 DSP 类指令都工作于字模式，只可对存放在工作寄存器中的有效地址进行偶数更新。每一个预取寄存器可允许的地址更改是 -6、-4、-2、0（无更新）、+2、+4 和 +6。这表明可在每一个方向最多进行 3 个字的有效地址更新。

当使用寄存器偏移量寻址模式时，将不对基本预取寄存器（W9 或 W11）或偏移量寄存器（W12）进行更新。

4.14.3 MAC 操作

DSP MAC 类指令执行的数学运算主要包括将两个工作寄存器中内容相乘并将结果加到或存入累加器 A 或累加器 B。执行这些操作的指令包括 MAC、MPY、MPY.N 和 MSC。表 4-9 表明在 MAC 类指令中 W4-W7 必须用作数据源操作数。W4-W7 可以任何形式进行组合，当同一工作寄存器同时被指定为源操作数和目标操作数时，将执行平方或平方并累加操作。

基于欧几里德距离运算的定义，对于 ED 和 EDAC 指令，指令必须指定相同的被乘数操作数。该指令的另一个独特特征是，从 X 和 Y 存储区预取出的值并非实际存放在 W4-W7 中的值。实际上，W4-W7 只存放预取数据字的差值。

其余的两个 MAC 类指令 CLR 和 MOVSAAC 用来启动或完成一系列 MAC 或 EDAC 指令而不使用乘法器。CLR 可用来对累加器 A 或 B 进行清零，从数据存储区预取两个值且存储另一个累加器的内容。类似地，MOVSAAC 可用来从数据存储区预取两个值并存储任一累加器中的内容。

4.14.4 MAC 回写

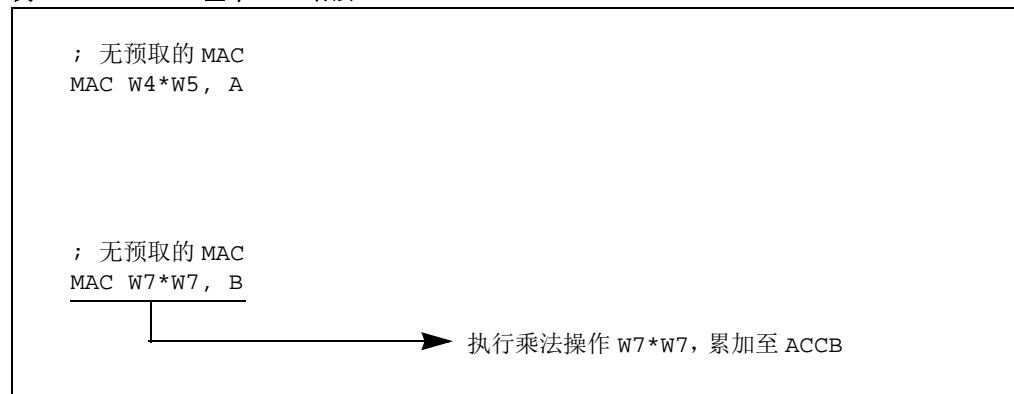
DSP MAC 类指令具有的回写能力有利于实现算法的高效处理。这个特点使得在同一个周期内，可使用一个累加器执行一个数学运算并将另一个累加器舍入后的内容进行存储。如表 4-9 中所示，寄存器 W13 被指定用来执行回写且支持两种寻址模式：直接寻址和执行后递增的间接寻址。

CLR、MOVSAAC 和 MSC 指令支持累加器回写，而 ED、EDAC、MPY 和 MPY.N 指令并不支持累加器回写。将两个不同的工作寄存器内容相乘的 MAC 类指令也支持累加器回写。然而，平方并累加的 MAC 类指令不支持累加器回写（见表 4-11）。

4.14.5 MAC 语法

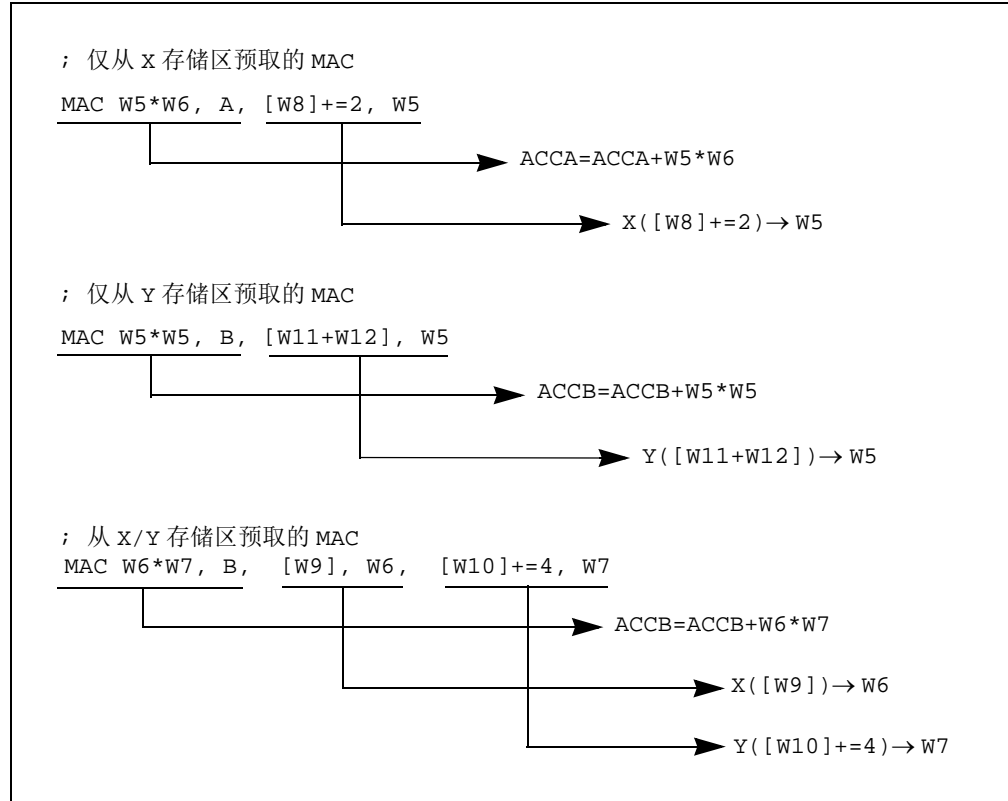
关于预取和累加器回写，根据指令类型和执行的的操作，MAC 类指令的语法具有几种格式。除了 CLR 和 MOVSAAC 指令，所有 MAC 类指令必须指定目标累加器以及两个乘数，如例 4-19 所示。

例 4-19: 基本 MAC 语法



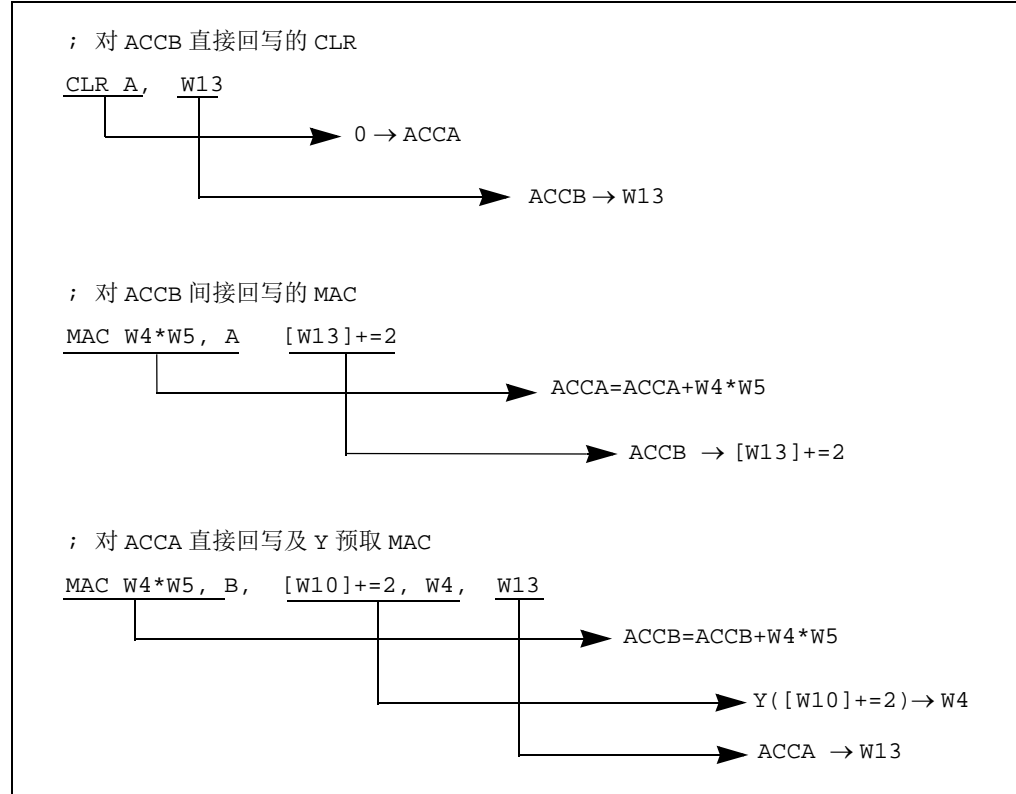
如果指令中使用预取，汇编器能够根据用于有效地址的寄存器区分出 X 还是 Y 存储区数据预取。
[W8] 或 [W9] 用于指定 X 存储区预取而 [W10] 或 [W11] 用于指定 Y 存储区预取。根据语法要求，需要在工作寄存器两边加方括号，表明通过间接寻址实现预取。当使用地址修改时，必须使用减 - 等于或加 + 等于的 C 语言类似语法（例如，“[W8] - = 2”或 “[W8] + = 6”）对其进行指定。当寄存器偏移量寻址用于预取时，W12 被置于方括号之内（[W9 + W12] 用于 X 存储区预取，[W11 + W12] 用于 Y 存储区预取）。每个预取操作也必须指定一个预取目标寄存器（W4-W7）。在指令语法中，目标寄存器出现在预取寄存器之前。预取的合法形式如例 4-20 所示。

例 4-20: MAC 预取语法



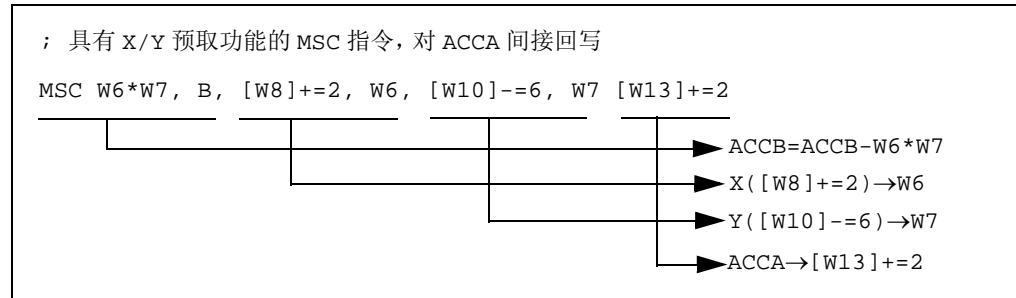
如果指令中使用累加器回写，将在最后对其进行指定。回写操作必须使用 W13 寄存器，而允许的回写形式包括 “W13” 以及 “[W13] + = 2”，分别用于直接寻址以及执行后递增的间接寻址。根据定义，将对数学运算中未使用的累加器的内容进行存储，因此指令中不指定回写累加器。累加器回写（WB）的合法形式如例 4-21 所示。

例 4-21: MAC 累加器回写语法



综上所述，例 4-22 中给出了实现两次预取和一次回写的一条 MSC 指令。

例 4-22: 执行两次预取和一次累加器回写的 msc 指令



4.15 DSP 累加器类指令（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

DSP 累加器类指令不具有预取功能或累加器回写能力，但提供了加法、求补、移位、装载以及将任一 40 位累加器内容进行存储的能力。此外，ADD 和 SUB 指令允许对两个累加器进行相加或相减。DSP 累加器类指令如表 4-12 所示，有关指令的详细介绍在第 5 章“指令描述”给出。

表 4-12: DSP 累加器类指令

| 指令 | 说明 | 具有累加器回写功能？ |
|-------|--------------------------|------------|
| ADD | 累加器相加 | 无 |
| ADD | 16 位有符号数与累加器相加 | 无 |
| LAC | 装载累加器 | 无 |
| NEG | 对累加器内容求补 | 无 |
| SAC | 保存累加器内容 | 无 |
| SAC.R | 保存舍入后的累加器内容 | 无 |
| SFTAC | 对累加器进行算术移位，移位位数由立即数指定 | 无 |
| SFTAC | 对累加器进行算术移位，移位位数由 (Wn) 指定 | 无 |
| SUB | 累加器相减 | 无 |

4.16 使用 FBCL 指令换算数据（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

为尽可能缩小与使用 DSP 类指令进行的数据处理相关的量化误差，充分利用运算得到的全部数值结果很重要。这可能需要将数据按比例放大以避免出现下溢（例如，处理来自 12 位 ADC 的数据时），或将数据按比例缩小以避免出现上溢（例如，将数据发送至 10 位 DAC 时）。必须对数据进行换算处理以将量化误差最小化。换算比例的确定取决于被操作输入数据的动态范围以及所需的输出数据的动态范围。有时，这些条件是事先知晓的，因而可采用固定的换算系数。在其他一些场合，换算条件可能不是固定或已知的，这样就必须使用动态换算方法对数据进行处理。

FBCL 指令（检测从左开始首先发生变化的数据位）可用来高效实现动态换算，因为它可确定一个值的指数。一个定点或整数值的指数表征该值在出现溢出前可进行移位的位数。这个信息是很有用的，因为它可用来实现数据的“满量程”换算，即其换算后的数值表示可利用存放该数值的寄存器中的所有位。

FBCL 指令通过按照从值的符号位至 LSb 的顺序检测第一个发生变化的数据位来确定数据字的指数。由于 dsPIC DSC 器件的桶形移位寄存器使用负值指定左移操作，FBCL 指令返回值的指数的相反数。如果对值进行按比例放大，这将允许立即执行随后的移位操作，而移位位数由 FBCL 指令的返回值确定。此外，由于 FBCL 指令只对有符号值进行操作，因此 FBCL 产生的结果处于 -15:0 范围内。当 FBCL 指令返回 0 时，表明该值已处于满量程。当指令返回 -15 时，表明不能对该值进行换算（对于 0x0 以及 0xFFFF）。表 4-13 显示不同动态范围的字数数据，它们的指数以及每一数据经过换算实现动态范围最大化后的值。例 4-23 说明了如何利用 FBCL 指令进行数据块处理。

表 4-13: 换算示例

| 字值 | 指数 | 满量程值 (字值 << 指数) |
|--------|----|--------------------|
| 0x0001 | 14 | 0x4000 |
| 0x0002 | 13 | 0x4000 |
| 0x0004 | 12 | 0x4000 |
| 0x0100 | 6 | 0x4000 |
| 0x01FF | 6 | 0x7FC0 |
| 0x0806 | 3 | 0x4030 |
| 0x2007 | 1 | 0x400E |
| 0x4800 | 0 | 0x4800 |
| 0x7000 | 0 | 0x7000 |
| 0x8000 | 0 | 0x8000 |
| 0x900A | 0 | 0x900A |
| 0xE001 | 2 | 0x8004 |
| 0xFF07 | 7 | 0x8380 |

注：对于字值 0x0000 和 0xFFFF，FBCL 指令返回 -15。

作为一个实际的例子，假定以数据块处理的方式对一个数据序列进行处理。该数据系列以 1.15 小数格式存储且动态范围很小。为使量化误差最小，可对数据按比例放大以避免在对其进行处理时出现量化丢失。可通过对序列中具有最大幅值的样本数据执行 FBCL 指令以确定用于处理数据的最佳换算系数值。注意，通过将数据左移可使数据按比例增大。下面的代码段说明了这一点。

例 4-23: 使用 FBCL 进行换算

```

; 假定 W0 包含数据块中的最大绝对值
; 假定 W4 指向数据块的开始处
; 假定数据块包含 BLOCK_SIZE 个字

; 确定用于换算操作的指数
FBCL    W0, W2          ; 将指数存入 W2

; 在数据处理前对整个数据块进行换算
DO      #(BLOCK_SIZE-1), SCALE
LAC     [W4], A          ; 将下一个数据样本传送至 ACCA
SFTAC   A, W2            ; 对 ACCA 进行移位操作，移位位数由 W2 中内容确定
SCALE:
SAC     A, [W4++]        ; 存储换算后的输入（将原来数据覆盖）

; 现在处理数据
; （处理模块放在这里）
```

4.17 使用 FBCL 指令将累加器中内容归一化（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）

对量化后的值进行换算以实现最大动态范围的过程称为归一化（表 4-13 中的第三列数据包含了归一化的数据）。累加器归一化作为一项技术，用来确保在存储累加器中数据之前累加器是正确对齐的，而 FBCL 指令有利于实现该功能。

每一个 40 位累加器都具有 ACCxU 寄存器的 8 个警戒位，这使得当工作于超饱和模式时累加器的动态范围从 1.31 扩展至 9.31（见第 4.11.1 节“整数和小数数据”）。然而，即使在超饱和模式，用于存储舍入后的累加器中内容的指令（SAC.R）只存储 ACCxH 中的 16 位数据（以 1.15 格式），如第 4.12 节“累加器的使用（dsPIC30F、dsPIC33F 和 dsPIC33E 器件）”中所介绍。在某些情况下，这可能产生问题。

通过下述方法可实现正确的数据对齐以存储累加器中的内容：如果 ACCxU 正被使用，则将累加器内容按比例缩小；如果 ACCxH 中所有位都未被使用，则将累加器内容按比例放大。FBCL 指令必须对 ACCxU 字节和 ACCxH 字进行操作以实现上述的换算。如果需要进行移位，将采用 ALU 的 40 位移位寄存器，使用 SFTAC 指令来执行换算操作。例 4-24 包含实现累加器归一化的代码段。

例 4-24: 使用 FBCL 实现归一化

```

; 假定 ACCA 中的操作刚结束（SR 未改变）
; 假定处理器处于超饱和模式
; 假定 ACCAH 定义为 ACCAH 的地址（0x24）

MOV    #ACCAH, W5          ; W5 指向 ACCAH
BRA    OA, FBCL_GUARD      ; 如果溢出，则右移
FBCL_HI:
FBCL    [W5], W0            ; 求出用于左移操作的指数
BRA     SHIFT_ACC          ; 转移到移位处理
FBCL_GUARD:
FBCL    [++W5], W0          ; 求出用于右移操作的指数
ADD.B   W0, #15, W0         ; 调整用于右移操作的符号
SHIFT_ACC:
SFTAC   A, W0              ; 对 ACCA 进行移位操作以实现归一化
```

4.18 使用混合符号乘法的扩展精度算术运算（仅限 dsPIC33E 器件）

许多 DSP 算法利用扩展精度算术运算（带 32 位或 64 位操作数和结果的运算）来提高计算的分辨率和精度。这可使用 16 位有符号或无符号乘法实现；但是，这将需要一些额外的处理并将数据进行移位以获得正确的结果。为加快这种扩展精度算法的计算速度，dsPIC33E 器件支持可选的隐含混合符号乘法模式，通过设置 US<1:0>（CORCON<13:12>）= 10 选择此模式。

在此模式下，混合符号（无符号 x 有符号和有符号 x 无符号）乘法的执行无需因操作数格式不同而动态重新配置 US<1:0> 位和移位数据。此外，有符号 x 有符号和无符号 x 无符号乘法的执行也无需更改乘法模式。如果用于指定操作数的工作寄存器为 W4 或 W6，将每个输入操作数隐含视为无符号数。类似地，如果使用的寄存器为 W5 或 W7，则将操作数视为有符号数。DSP 引擎基于所使用的操作数寄存器选择要执行的乘法的类型，因此无需用户软件修改 US<1:0> 位。

下面的代码示例说明了隐含混合符号乘法特性缩短了执行时间。在该示例中，当使能混合符号乘法模式时，执行 32 位乘法的指令周期数从 7 个周期缩短到了 4 个周期。

例 4-25: 使用隐含混合符号模式的 32 位有符号乘法

情形 A——不使能混合符号乘法模式

```
MUL.SU W5, W6, W0; Word1（有符号）x Word2（无符号）
MUL.US W4, W7, W2; Word0（无符号）x Word3（有符号）
CLR    B           ; 清零累加器 B
ADD    W1, B
ADD    W3, B
SFTAC  B, #15      ; 右移 15 位以对齐到 Q31 格式
MAC    W5*W7, B; Word1（有符号）x Word 3（有符号）
```

情形 B——使能混合符号乘法模式

```
MPY    W5*W6, B; Word1（有符号）x Word2（无符号）
MAC    W4*W7, B; Word0（无符号）x Word3（有符号）
SFTAC  B, #15      ; 右移 15 位以对齐到 Q31 格式
MAC    W5*W7, B; Word1（有符号）x Word 3（有符号）
```

除 DSP 类指令外，MCU 类乘法（MUL）指令还可利用累加器 A 或累加器 B 作为结果的目标寄存器，这可加快扩展精度算术运算的执行，即使在不使用 DSP 乘法指令（如 MPY 或 MAC）时。

注:

第 5 章 指令描述

目录

本章包括以下主要内容：

| | | |
|-----|-------------------|----|
| 5.1 | 指令符号..... | 94 |
| 5.2 | 指令编码字段描述符介绍 | 94 |
| 5.3 | 指令描述示例 | 98 |
| 5.4 | 指令描述..... | 99 |

16 位 MCU 和 DSC 程序员参考手册

5.1 指令符号

表 1-2 列出了第 5.4 节“指令描述”中使用的所有符号。

5.2 指令编码字段描述符介绍

表 5-2 至表 5-12 显示了第 5.4 节“指令描述”中使用的所有指令编码字段描述符。

表 5-1: 指令编码字段描述符

| 字段 | 描述 |
|---------------------------------|---|
| A ⁽¹⁾ | 累加器选择位: 0 = ACCA ; 1 = ACCB |
| aa ⁽¹⁾ | 累加器回写模式 (见表 5-12) |
| B | 字节模式选择位: 0 = 字操作; 1 = 字节操作 |
| bbbb | 4 位位置选择: 0000 = LSb ; 1111 = MSb |
| D | 目标地址位: 0 = 结果存入 WREG ; 1 = 结果存入文件寄存器 |
| dddd | Wd 目标寄存器选择: 0000 = W0 ; 1111 = W15 |
| f ffff ffff ffff | 13 位文件寄存器地址 (0x0000 至 0x1FFF) |
| fff ffff ffff ffff | 15 位文件寄存器地址 (隐含为 0 的 LSb) (0x0000 至 0xFFFF) |
| ffff ffff ffff ffff | 16 位文件寄存器地址 (0x0000 至 0xFFFF) |
| ggg | 用于 Ws 源寄存器的寄存器偏移量寻址模式 (见表 5-4) |
| hhh | 用于 Wd 目标寄存器的寄存器偏移量寻址模式 (见表 5-5) |
| iii ⁽¹⁾ | X 数据空间预取操作 (见表 5-6) |
| jjj ⁽¹⁾ | Y 数据空间预取操作 (见表 5-8) |
| k | 1 位立即数字段, 常数或表达式 |
| kkkk | 4 位立即数字段, 常数或表达式 |
| kk kkkk | 6 位立即数字段, 常数或表达式 |
| kkkk kkkk | 8 位立即数字段, 常数或表达式 |
| kk kkkk kkkk | 10 位立即数字段, 常数或表达式 |
| kk kkkk kkkk kkkk | 14 位立即数字段, 常数或表达式 |
| kkkk kkkk kkkk kkkk | 16 位立即数字段, 常数或表达式 |
| mm | 使用相同工作寄存器时的乘法源操作数选择 (见表 5-10) |
| mmm | 使用不同工作寄存器时的乘法源操作数选择 (见表 5-11) |
| nnnn nnnn nnnn nnn0 nnn nnnn | 用于 CALL 和 GOTO 指令的 23 位程序地址 |
| nnnn nnnn nnnn nnnn | 用于相对转移 / 调用指令的 16 位程序偏移量字段 |
| ppp | 用于 Ws 源寄存器的寻址模式 (见表 5-2) |
| qqq | 用于 Wd 目标寄存器的寻址模式 (见表 5-3) |
| rrrr | 桶形移位计数 |
| ssss | Ws 源寄存器选择: 0000 = W0 ; 1111 = W15 |
| tttt | 被除数选择, 高位字 |
| vvvv | 被除数选择, 低位字 |
| W | 双字模式选择位: 0 = 字操作; 1 = 双字操作 |
| www | Wb 基准寄存器选择: 0000 = W0 ; 1111 = W15 |
| xx ⁽¹⁾ | X 数据空间预取目标寄存器 (见表 5-7) |
| xxxx xxxx xxxx xxxx | 16 位未用字段 (无关) |
| yy ⁽¹⁾ | Y 数据空间预取目标寄存器 (见表 5-9) |
| z | 位测试目标: 0 = C 标志位; 1 = Z 标志位 |

注 1: 该位域仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。

表 5-2: 用于 Ws 源寄存器的寻址模式

| PPP | 寻址模式 | 源操作数 |
|-----|----------|--------|
| 000 | 寄存器直接 | Ws |
| 001 | 间接 | [Ws] |
| 010 | 执行后递减的间接 | [Ws--] |
| 011 | 执行后递增的间接 | [Ws++] |
| 100 | 执行前递减的间接 | [--Ws] |
| 101 | 执行前递增的间接 | [++Ws] |
| 11x | 未用 | |

表 5-3: 用于 Wd 目标寄存器的寻址模式

| qqq | 寻址模式 | 目标操作数 |
|-----|---------------------------|--------|
| 000 | 寄存器直接 | Wd |
| 001 | 间接 | [Wd] |
| 010 | 执行后递减的间接 | [Wd--] |
| 011 | 执行后递增的间接 | [Wd++] |
| 100 | 执行前递减的间接 | [--Wd] |
| 101 | 执行前递增的间接 | [++Wd] |
| 11x | 未用（使用该寻址模式将强制执行 RESET 指令） | |

表 5-4: 用于 Ws 源寄存器的偏移量寻址模式（带寄存器偏移量）

| ggg | 寻址模式 | 源操作数 |
|-----|----------|---------|
| 000 | 寄存器直接 | Ws |
| 001 | 间接 | [Ws] |
| 010 | 执行后递减的间接 | [Ws--] |
| 011 | 执行后递增的间接 | [Ws++] |
| 100 | 执行前递减的间接 | [--Ws] |
| 101 | 执行前递增的间接 | [++Ws] |
| 11x | 寄存器偏移量间接 | [Ws+Wb] |

表 5-5: 用于 Wd 目标寄存器的偏移量寻址模式（带寄存器偏移量）

| hhh | 寻址模式 | 源操作数 |
|-----|----------|---------|
| 000 | 寄存器直接 | Wd |
| 001 | 间接 | [Wd] |
| 010 | 执行后递减的间接 | [Wd--] |
| 011 | 执行后递增的间接 | [Wd++] |
| 100 | 执行前递减的间接 | [--Wd] |
| 101 | 执行前递增的间接 | [++Wd] |
| 11x | 寄存器偏移量间接 | [Wd+Wb] |

表 5-6: X 数据空间预取操作 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| iiii | 操作 |
|------|-------------------------|
| 0000 | Wxd = [W8] |
| 0001 | Wxd = [W8], W8 = W8 + 2 |
| 0010 | Wxd = [W8], W8 = W8 + 4 |
| 0011 | Wxd = [W8], W8 = W8 + 6 |
| 0100 | 对 X 数据空间无预取操作 |
| 0101 | Wxd = [W8], W8 = W8 - 6 |
| 0110 | Wxd = [W8], W8 = W8 - 4 |
| 0111 | Wxd = [W8], W8 = W8 - 2 |
| 1000 | Wxd = [W9] |
| 1001 | Wxd = [W9], W9 = W9 + 2 |
| 1010 | Wxd = [W9], W9 = W9 + 4 |
| 1011 | Wxd = [W9], W9 = W9 + 6 |
| 1100 | Wxd = [W9 + W12] |
| 1101 | Wxd = [W9], W9 = W9 - 6 |
| 1110 | Wxd = [W9], W9 = W9 - 4 |
| 1111 | Wxd = [W9], W9 = W9 - 2 |

表 5-7: X 数据空间预取目标寄存器 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| xx | Wxd |
|----|-----|
| 00 | W4 |
| 01 | W5 |
| 10 | W6 |
| 11 | W7 |

表 5-8: Y 数据空间预取操作 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| jjjj | 操作 |
|------|----------------------------|
| 0000 | Wyd = [W10] |
| 0001 | Wyd = [W10], W10 = W10 + 2 |
| 0010 | Wyd = [W10], W10 = W10 + 4 |
| 0011 | Wyd = [W10], W10 = W10 + 6 |
| 0100 | 对 Y 数据空间无预取操作 |
| 0101 | Wyd = [W10], W10 = W10 - 6 |
| 0110 | Wyd = [W10], W10 = W10 - 4 |
| 0111 | Wyd = [W10], W10 = W10 - 2 |
| 1000 | Wyd = [W11] |
| 1001 | Wyd = [W11], W11 = W11 + 2 |
| 1010 | Wyd = [W11], W11 = W11 + 4 |
| 1011 | Wyd = [W11], W11 = W11 + 6 |
| 1100 | Wyd = [W11 + W12] |
| 1101 | Wyd = [W11], W11 = W11 - 6 |
| 1110 | Wyd = [W11], W11 = W11 - 4 |
| 1111 | Wyd = [W11], W11 = W11 - 2 |

表 5-9: Y 数据空间预取目标寄存器 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| YY | Wyd |
|----|-----|
| 00 | W4 |
| 01 | W5 |
| 10 | W6 |
| 11 | W7 |

表 5-10: MAC 或 MPY 源操作数 (相同工作寄存器) (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| mm | 被乘数 |
|----|---------|
| 00 | W4 * W4 |
| 01 | W5 * W5 |
| 10 | W6 * W6 |
| 11 | W7 * W7 |

表 5-11: MAC 或 MPY 源操作数 (不同工作寄存器) (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| mmm | 被乘数 |
|-----|---------|
| 000 | W4 * W5 |
| 001 | W4 * W6 |
| 010 | W4 * W7 |
| 011 | 无效 |
| 100 | W5 * W6 |
| 101 | W5 * W7 |
| 110 | W6 * W7 |
| 111 | 无效 |

表 5-12: MAC 累加器回写选择 (dsPIC30F、dsPIC33F 和 dsPIC33E 器件)

| aa | 回写选择 |
|----|---------------------------------|
| 00 | W13 = 其他累加器 (直接寻址) |
| 01 | [W13] += 2 = 其他累加器 (执行后递增的间接寻址) |
| 10 | 无回写 |
| 11 | 无效 |

表 5-13: MOVWAG 目标寄存器选择

| PP | 目标页寄存器 |
|----|------------|
| 00 | DSRPAG |
| 01 | DSWPAG |
| 10 | TBLPAG |
| 11 | 保留 —— 不要使用 |

表 5-14: 累加器选择

| A | 目标累加器 |
|---|-------|
| 0 | 累加器 A |
| 1 | 累加器 B |

5.3 指令描述示例

以下示例描述是针对虚构指令 FOO 的，用来说明如何通过表字段（语法、操作数和操作等）对第 5.4 节“指令描述”列出的指令进行描述。

FOO

指令标题字段对指令所执行的操作进行概括

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

标有“X”的单元指示在该器件系列中实现了该指令。

- 语法:

语法字段包括可选的标号、指令助记符、任何可选的指令扩展以及指令的操作数。大多数指令都支持多个操作数的形式以支持各种寻址模式。在这些情况下，所有可能的指令操作数都在下面列出并放在方括号内。
- 操作数:

操作数字段对每个操作数的取值集合进行描述。操作数可以是累加器、文件寄存器、立即数常量（有符号或无符号）或工作寄存器。
- 操作:

该字段概括指令执行的操作。
- 受影响的状态位:

该字段对指令执行将影响状态寄存器中哪一（些）位进行说明。状态位将以降序方式按照位位置顺序列出。
- 指令编码:

该字段说明了指令是如何进行位编码的。描述字段对各个位域进行了描述，完整的编码细节在表 5-1 中给出。
- 描述:

该字段详细描述了指令执行的操作。同时还给出了编码中关键位的信息。
- 指令字数:

该字段包含用于在存储器中存储指令的程序字数。
- 指令周期数:

该字段包含执行指令所需的指令周期数。
- 示例:

该字段包含说明指令如何操作的应用示例。其中给出了指令执行前和执行后寄存器的状态，这使得用户可清楚了解指令执行的操作。

5.4 指令描述

ADD

f 与 WREG 相加

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ADD{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (f) + (WREG) → 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0100 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将默认工作寄存器 WREG 中的内容与文件寄存器中的内容相加，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: ADD.B RAM100 ; 将 WREG 中内容加到 RAM100（字节模式）

| 指令执行前 | | | 指令执行后 | | |
|--------|------|--|--------|------|-------------|
| WREG | CC80 | | WREG | CC80 | |
| RAM100 | FFC0 | | RAM100 | FF40 | |
| SR | 0000 | | SR | 0005 | (OV, C = 1) |

例 2: ADD RAM200, WREG ; 将 RAM200 中内容加到 WREG（字模式）

| 指令执行前 | | | 指令执行后 | | |
|--------|------|--|--------|------|---------|
| WREG | CC80 | | WREG | CC40 | |
| RAM200 | FFC0 | | RAM200 | FFC0 | |
| SR | 0000 | | SR | 0001 | (C = 1) |

ADD

立即数与 Wn 相加

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } ADD{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: lit10 + (Wn) → Wn

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0000 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 10 位无符号立即数操作数与工作寄存器 Wn 中的内容相加, 并将结果存回工作寄存器 Wn。

B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
k 位用于指定立即数操作数。
d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 1

例 1: ADD.B #0xFF, W7 ; 将 -1 加到 W7 (字节模式)

| | | | |
|-------|------|-------|-----------------|
| 指令执行前 | | 指令执行后 | |
| W7 | 12C0 | W7 | 12BF |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: ADD #0xFF, W1 ; 将 255 加到 W1 (字模式)

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W1 | 12C0 | W1 | 13BF |
| SR | 0000 | SR | 0000 |

ADD

Wb 与短立即数相加

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ADD{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: (Wb) + lit5 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0100 | 0www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将基准寄存器 **Wb** 中的内容与 **5** 位无符号短立即数操作数相加，并将结果存放在目标寄存器 **Wd** 中。必须使用寄存器直接寻址对 **Wb** 进行寻址。可使用寄存器直接或间接寻址对 **Wd** 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 **5** 位整数的立即数操作数。

注: 指令中的扩展符 **.B** 指明是字节操作而非字操作。用户可使用 **.W** 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1

例 1: ADD.B W0, #0x1F, W7 ; 将 W0 和 31 相加（字节模式）
; 结果存入 W7

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 2290 | W0 | 2290 |
| W7 | 12C0 | W7 | 12AF |
| SR | 0000 | SR | 0008 (N = 1) |

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节“多周期指令”中的注 3。

例 2: ADD B ; 将 ACCA 加到 ACCB
 ; 假定超饱和模式使能
 ; (ACCSAT = 1, SATA = 1, SATB = 1)

| 指令执行前 | | | 指令执行后 | | |
|-------|--------------|--|-------|--------------|---------------|
| ACCA | 00 E111 2222 | | ACCA | 00 E111 2222 | |
| ACCB | 00 7654 3210 | | ACCB | 01 5765 5432 | |
| SR | 0000 | | SR | 4800 | (OB, OAB = 1) |

ADD 16 位有符号数加到累加器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } ADD Ws, {#Slit4,} Acc
 [Ws],
 [Ws++],
 [Ws--],
 [--Ws],
 [++Ws],
 [Ws+Wb],

操作数: Ws ∈ [W0 ... W15]
 Wb ∈ [W0 ... W15]
 Slit4 ∈ [-8 ... +7]
 Acc ∈ [A,B]

操作: Shift_{Slit4}(扩展 (Ws)) + (Acc) → Acc

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1001 | Awww | wrrr | rggg | ssss |
|-------|------|------|------|------|------|------|

描述: 将源工作寄存器指定的 16 位值加到选择的累加器的高位字。源操作数可指定为工作寄存器或有效地址的内容。在将指定值与累加器最高位字相加之前，将对源操作数进行符号扩展以及零回填。在加法执行前，还可将加到累加器的值进行移位，移位位数由一个 4 位有符号立即数指定。

A 位用于指定目标累加器。
w 位用于指定偏移量寄存器 Wb。
r 位用于对可选的移位进行编码。
g 位用于选择源地址模式。
s 位用于指定源寄存器 Ws。

注: 如果操作数 Slit4 为正，将执行算术右移操作。如果操作数 Slit4 为负，则将执行算术左移操作。Slit4 将不会对源寄存器中的内容产生影响。

ADD16 位有符号数加到累加器

指令字数：1
指令周期数：1(1)

注 1：在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1：ADD W0, #2, A ; 将 W0 右移 2 位后加到 ACCA

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W0 | 8000 | W0 | 8000 |
| ACCA | 00 7000 0000 | ACCA | 00 5000 0000 |
| SR | 0000 | SR | 0000 |

例 2：ADD [W5++], A ; 将 W5 的有效值加到 ACCA
; 指令执行后，递增 W5

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W5 | 2000 | W5 | 2002 |
| ACCA | 00 0067 2345 | ACCA | 00 5067 2345 |
| 数据单元 2000 | 5000 | 数据单元 2000 | 5000 |
| SR | 0000 | SR | 0000 |

| 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|------------------|
| WREG | 5600 | WREG | 8A01 |
| RAM200 | 3400 | RAM200 | 3400 |
| SR | 0001 (C = 1) | SR | 000C (N, OV = 1) |

ADDC

立即数带进位位加到 Wn

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ADDC{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
 对于字操作, lit10 ∈ [0 ... 1023]
 Wn ∈ [W0 ... W15]

操作: lit10 + (Wn) + (C) → Wn

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0000 | 1Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 10 位无符号立即数操作数、工作寄存器 Wn 中的内容与进位位相加，并将结果存回工作寄存器 Wn。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于指定立即数操作数。
d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2: 对于字节操作，立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息，请参见第 4.6 节“使用 10 位立即数操作数”。
- 3: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1
指令周期数: 1

例 1: ADDC.B #0xFF, W7 ; 将 -1 和 C 位加到 W7（字节模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|----------------|
| W7 | 12C0 | W7 | 12BF |
| SR | 0000 (C = 0) | SR | 0009 (N,C = 1) |

例 2: ADDC #0xFF, W1 ; 将 255 和 C 位加到 W1（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|------|
| W1 | 12C0 | W1 | 13C0 |
| SR | 0001 (C = 1) | SR | 0000 |

| | | | | | |
|-----|---------|----------|-----|--------|--------|
| 语法: | { 标号 :} | ADDC{.B} | Wb, | #lit5, | Wd |
| | | | | | [Wd] |
| | | | | | [Wd++] |
| | | | | | [Wd--] |
| | | | | | [++Wd] |
| | | | | | [--Wd] |

2: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令周期数: 1

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W0 | CC80 | W0 | CC80 |
| W7 | 12C0 | W7 | 12C0 |
| 数据单元 12C0 | B000 | 数据单元 12C0 | B09F |
| SR | 0000 (C = 0) | SR | 0008 (N = 1) |

例 2: ADDC W3, #0x6, [--W4] ; 将 W3、6 和 C 位相加（字模式）
 ; 结果存入 [--W4]

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|---------|--|-----------|------|--|--|
| W3 | 6006 | | | W3 | 6006 | | |
| W4 | 1000 | | | W4 | 0FFE | | |
| 数据单元 0FFE | DDEE | | | 数据单元 0FFE | 600D | | |
| 数据单元 1000 | DDEE | | | 数据单元 1000 | DDEE | | |
| SR | 0001 | (C = 1) | | SR | 0000 | | |

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节“多周期指令”中的注 3。

| | | | |
|-------------|--------|--------------------|--|
| <u>例 1:</u> | ADDC.B | W0, [W1++], [W2++] | ; 将 W0、[W1] 和 C 位相加 (字节模式) ; 结果存入 [W2] ; 指令执行后, 递增 W1 和 W2 |
|-------------|--------|--------------------|--|

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|------|
| W0 | CC20 | W0 | CC20 |
| W1 | 0800 | W1 | 0801 |
| W2 | 1000 | W2 | 1001 |
| 数据单元 0800 | AB25 | 数据单元 0800 | AB25 |
| 数据单元 1000 | FFFF | 数据单元 1000 | FF46 |
| SR | 0001 (C = 1) | SR | 0000 |

| | | | |
|-------------|------|--------------------|---|
| <u>例 2:</u> | ADDC | W3, [W2++], [W1++] | ; 将 W3、[W2] 和 C 位相加（字模式） ; 结果存入 [W1] ; 指令执行后，递增 W1 和 W2 |
|-------------|------|--------------------|---|

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W1 | 1000 | W1 | 1002 |
| W2 | 2000 | W2 | 2002 |
| W3 | 0180 | W3 | 0180 |
| 数据单元 1000 | 8000 | 数据单元 1000 | 2681 |
| 数据单元 2000 | 2500 | 数据单元 2000 | 2500 |
| SR | 0001 | SR | 0000 |

(C = 1)

AND

f 和 WREG 逻辑与

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} AND{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (f).AND.(WREG) → 由 D 指定的目标寄存器

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0110 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将默认工作寄存器 WREG 中的内容和文件寄存器中的内容进行逻辑与运算，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: AND.B RAM100 ; 将 WREG 和 RAM100 中内容进行逻辑与（字节模式）
; 结果存放在 RAM100 中

| | | | |
|--------|-------|-------|---------|
| | 指令执行前 | 指令执行后 | |
| WREG | CC80 | CC80 | (N = 1) |
| RAM100 | FFC0 | FF80 | |
| SR | 0000 | 0008 | |

例 2: AND RAM200, WREG ; 将 RAM200 和 WREG 中内容进行逻辑与（字模式）
; 结果存放在 WREG 中

| | | | |
|--------|-------|-------|--|
| | 指令执行前 | 指令执行后 | |
| WREG | CC80 | 0080 | |
| RAM200 | 12C0 | 12C0 | |
| SR | 0000 | 0000 | |

AND

立即数和 Wn 逻辑与

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} AND{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: lit10.AND.(Wn) → Wn

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0010 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 10 位立即数操作数和工作寄存器 Wn 中的内容进行逻辑与运算，并将结果存回工作寄存器 Wn。必须使用寄存器直接寻址对 Wn 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于指定立即数操作数。
d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: 对于字节操作，立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息，请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 1

例 1: AND.B #0x83, W7 ; 将 0x83 和 W7 进行逻辑与（字节模式）
; 结果存放在 W7 中

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W7 | 12C0 | W7 | 1280 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: AND #0x333, W1 ; 将 0x333 和 W1 进行逻辑与（字模式）
; 结果存放在 W1 中

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W1 | 12D0 | W1 | 0210 |
| SR | 0000 | SR | 0000 |

AND

Wb 和短立即数逻辑与

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} AND{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: (Wb).AND.lit5 → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0110 | 0www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将基准寄存器 Wb 中的内容和 5 位立即数进行逻辑与运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: AND.B W0,#0x3,[W1++] ; 将 W0 和 0x3 进行逻辑与（字节模式）
; 结果存入 [W1]
; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 23A5 | W0 | 23A5 |
| W1 | 2211 | W1 | 2212 |
| 数据单元 2210 | 9999 | 数据单元 2210 | 0199 |
| SR | 0000 | SR | 0000 |

例 2: AND W0,#0x1F,W1 ; 将 W0 和 0x1F 进行逻辑与（字模式）
; 结果存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 6723 | W0 | 6723 |
| W1 | 7878 | W1 | 0003 |
| SR | 0000 | SR | 0000 |

AND

Wb 和 Ws 逻辑与

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} AND{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Wb).AND.(Ws) → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0110 | 0www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容和基准寄存器 Wb 中的内容进行逻辑与运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: AND.B W0, W1 [W2++] ; 将 W0 和 W1 进行逻辑与，
 ; 并将结果存入 [W2]（字节模式）
 ; 指令执行后，递增 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | AA55 | W0 | AA55 |
| W1 | 2211 | W1 | 2211 |
| W2 | 1001 | W2 | 1002 |
| 数据单元 1000 | FFFF | 数据单元 1000 | 11FF |
| SR | 0000 | SR | 0000 |

例 2: AND W0, [W1++], W2 ; 将 W0 和 [W1] 进行逻辑与，
 ; 并将结果存入 W2（字模式）
 ; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | AA55 | W0 | AA55 |
| W1 | 1000 | W1 | 1002 |
| W2 | 55AA | W2 | 2214 |
| 数据单元 1000 | 2634 | 数据单元 1000 | 2634 |
| SR | 0000 | SR | 0000 |

ASR

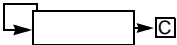
算术右移 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ASR{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
(f<7>) → Dest<7>
(f<7>) → Dest<6>
(f<6:1>) → Dest<5:0>
(f<0>) → C
对于字操作:
(f<15>) → Dest<15>
(f<15>) → Dest<14>
(f<14:1>) → Dest<13:0>
(f<0>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0101 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容右移一位，并将结果存放在目标寄存器中。文件寄存器中最低位移入状态寄存器的进位位。在移位操作执行后，对结果进行符号扩展。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: ASR.B RAM400, WREG ; 将 RAM400 右移一位, 并将结果存入 WREG
; (字节模式)

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|--------------|
| WREG | 0600 | WREG | 0611 |
| RAM400 | 0823 | RAM400 | 0823 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: ASR RAM200 ; 将 RAM200 右移一位 (字模式)

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|-----------------|
| RAM200 | 8009 | RAM200 | C004 |
| SR | 0000 | SR | 0009 (N, C = 1) |

ASR

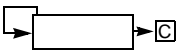
算术右移 *Ws*

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } ASR{.B} *Ws*, *Wd*
[*Ws*], [*Wd*]
[*Ws*++], [*Wd*++]
[*Ws*--], [*Wd*--]
[++*Ws*], [++*Wd*]
[--*Ws*], [--*Wd*]

操作数： *Ws* ∈ [*W0* ... *W15*]
Wd ∈ [*W0* ... *W15*]

操作： 对于字节操作：
(*Ws*<7>) → *Wd*<7>
(*Ws*<7>) → *Wd*<6>
(*Ws*<6:1>) → *Wd*<5:0>
(*Ws*<0>) → *C*
对于字操作：
(*Ws*<15>) → *Wd*<15>
(*Ws*<15>) → *Wd*<14>
(*Ws*<14:1>) → *Wd*<13:0>
(*Ws*<0>) → *C*



受影响的状态位： *N*、*Z* 和 *C*

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1101 | 0001 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述： 将源寄存器 *Ws* 中的内容右移一位，并将结果存放在目标寄存器 *Wd* 中。*Ws* 中的最低位移入状态寄存器的进位位。在移位操作执行后，对结果进行符号扩展。可使用寄存器直接或间接寻址对 *Ws* 和 *Wd* 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注： 指令中的扩展符 *.B* 指明是字节操作而非字操作。用户可使用 *.w* 扩展符指明是字操作，但这并不需要。

指令字数： 1
指令周期数： 1⁽¹⁾

注 1： 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: ASR.B [W0++], [W1++] ; 将 [W0] 右移一位, 并将结果存入 [W1]
(字节模式)
; 指令执行后, 递增 W0 和 W1

| 指令执行前 | | 指令执行后 | |
|----------|------|----------|------|
| W0 | 0600 | W0 | 0601 |
| W1 | 0801 | W1 | 0802 |
| 数据单元 600 | 2366 | 数据单元 600 | 2366 |
| 数据单元 800 | FFC0 | 数据单元 800 | 33C0 |
| SR | 0000 | SR | 0000 |

例 2: ASR W12, W13 ; 将 W12 右移一位, 并将结果存入 W13
(字模式)

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|-----------------|
| W12 | AB01 | W12 | AB01 |
| W13 | 0322 | W13 | D580 |
| SR | 0000 | SR | 0009 (N, C = 1) |

ASR

算术右移（移位位数由短立即数确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } ASR Wb, #lit4, Wnd

操作数： Wb ∈ [W0 ... W15]
lit4 ∈ [0...15]
Wnd ∈ [W0 ... W15]

操作： lit4<3:0> → Shift_Val
Wb<15> → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

受影响的状态位： N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1101 | 1110 | 1www | wddd | d100 | kkkk |
|-------|------|------|------|------|------|------|

描述： 将源寄存器 Wb 中的内容进行算术右移，移位位数由 4 位无符号立即数确定，并将结果存入目标寄存器 Wnd。在移位操作执行后，对结果进行符号扩展。必须使用直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
k 位用于提供立即数操作数。

注： 该指令只能工作于字模式。

指令字数： 1
指令周期数： 1

例 1: ASR W0, #0x4, W1 ; 将 W0 算术右移 4 位，并将结果存入 W1

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 060F | W0 | 060F |
| W1 | 1234 | W1 | 0060 |
| SR | 0000 | SR | 0000 |

例 2: ASR W0, #0x6, W1 ; 将 W0 算术右移 6 位，并将结果存入 W1

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W0 | 80FF | W0 | 80FF |
| W1 | 0060 | W1 | FE03 |
| SR | 0000 | SR | 0008 (N = 1) |

例 3: ASR W0, #0xF, W1 ; 将 W0 算术右移 15 位，并将结果存入 W1

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W0 | 70FF | W0 | 70FF |
| W1 | CC26 | W1 | 0000 |
| SR | 0000 | SR | 0002 (Z = 1) |

ASR 算术右移（移位位数由 Wns 确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ASR Wb, Wns, Wnd

操作数: Wb ∈ [W0 ... W15]
Wns ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作: Wns<3:0> → Shift_Val
Wb<15> → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 1110 | 1www | wddd | d000 | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Wb 中的内容进行算术右移，移位位数由 Wns 的低 4 位确定（最大移位位数为 15 位），并将结果存入目标寄存器 Wnd。在移位操作执行后，对结果进行符号扩展。必须使用直接寻址对 Wb、Wns 和 Wnd 进行寻址。
w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: 当 Wns 大于 15 时，如果 Wb 为正，则 Wnd = 0x0；如果 Wb 为负，则 Wnd = 0xFFFF。

指令字数: 1

指令周期数: 1

例 1: ASR W0, W5, W6 ; 将 W0 算术右移 W5 位，并将结果存入 W6

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 80FF | W0 | 80FF |
| W5 | 0004 | W5 | 0004 |
| W6 | 2633 | W6 | F80F |
| SR | 0000 | SR | 0000 |

例 2: ASR W0, W5, W6 ; 将 W0 算术右移 W5 位，并将结果存入 W6

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 6688 | W0 | 6688 |
| W5 | 000A | W5 | 000A |
| W6 | FF00 | W6 | 0019 |
| SR | 0000 | SR | 0000 |

例 3: ASR W11, W12, W13 ; 将 W11 算术右移 W12 位，并将结果存入 W13

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W11 | 8765 | W11 | 8765 |
| W12 | 88E4 | W12 | 88E4 |
| W13 | A5A5 | W13 | F876 |
| SR | 0000 | SR | 0008 (N = 1) |

BCLR

将 f 中的指定位清零

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BCLR{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 0 → f<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1001 | bbbf | ffff | ffff | fffb |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器 f 中由 bit4 指定的位清零。位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

b 位用于选择指定要被清零的位位置的 bit4 值。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BCLR.B 0x800, #0x7 ; 将 0x800 中的 bit 7 清零

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 0800 | 66EF | 数据单元 0800 | 666F |
| SR | 0000 | SR | 0000 |

例 2: BCLR 0x400, #0x9 ; 将 0x400 中的 bit 9 清零

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 0400 | AA55 | 数据单元 0400 | A855 |
| SR | 0000 | SR | 0000 |

BCLR

将 Ws 中的指定位清零

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BCLR{.B} Ws, #bit4
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 0 → Ws<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0001 | bbbb | 0B00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将寄存器 Ws 中由 bit4 指定的位清零。位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被清零的位位置的 bit4 值。
B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
s 位用于选择源 / 目标寄存器。
p 位用于选择源地址模式。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 源寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。
- 4: 在 dsPIC33E 和 PIC24E 器件中, 该指令使用 DSRPAG 寄存器来间接生成扩展数据空间中的地址。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

BRA 无条件转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号: } BRA Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: (PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0111 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 该指令使程序将相对于下一个 PC 值进行无条件转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。由于 PC 将已递增以取出下一条指令，因此在执行转移后新 PC 值将为 (PC + 2) + 2 * Slit16。n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（程序字数）。

指令字数: 1

指令周期数: 2（PIC24F、PIC24H、dsPIC30F 和 dsPIC33F）
4（PIC24E 和 dsPIC33E）

例 1: 002000 HERE: BRA THERE ; 转移到 THERE
002002 . . .
002004 . . .
002006 . . .
002008 . . .
00200A THERE: . . .
00200C . . .

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200A |
| SR | 0000 | SR | 0000 |

例 2: 002000 HERE: BRA THERE+0x2 ; 转移到 THERE+0x2
002002 . . .
002004 . . .
002006 . . .
002008 . . .
00200A THERE: . . .
00200C . . .

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200C |
| SR | 0000 | SR | 0000 |

例 3:

002000 HERE: BRA 0x1366 ; 转移到 0x1366

002002 . . .

002004 . . .

| | |
|----|---------|
| | 指令执行前 |
| PC | 00 2000 |
| SR | 0000 |

| | |
|----|---------|
| | 指令执行后 |
| PC | 00 1366 |
| SR | 0000 |

BRA

计算转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} BRA Wn

操作数: Wn ∈ [W0 ... W15]
操作: (PC + 2) + (2 * Wn) → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0110 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 该指令使程序将相对于下一个 PC 值进行无条件转移。转移操作的偏移量为一个符号扩展的 17 位值 (2 * Wn)，可支持最多向前或向后转移 32K 个指令字。由于 PC 将已递增以取出下一条指令，因此在执行这条指令后新 PC 值将为 (PC + 2) + 2 * Wn。

s 位用于选择源寄存器。

指令字数: 1

指令周期数: 2

例 1: 002000 HERE: BRA W7 ; 向前转移 (2+2*W7)
002002 . . .
.
.
002108 . . .
00210A TABLE7: . . .
00210C . . .

| | |
|-------|---------|
| 指令执行前 | |
| PC | 00 2000 |
| W7 | 0084 |
| SR | 0000 |

| | |
|-------|---------|
| 指令执行后 | |
| PC | 00 210A |
| W7 | 0084 |
| SR | 0000 |

BRA

计算转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } BRA Wn

操作数: Wn ∈ [W0 ... W15]

操作: (PC + 2) + (2 * Wn) → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0000 | 0110 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 该指令使程序将相对于下一个 PC 值进行无条件转移。转移操作的偏移量为一个符号扩展的 17 位值 (2 * Wn)，可支持最多向前或向后转移 32K 个指令字。由于 PC 将已递增以取出下一条指令，因此在执行这条指令后新 PC 值将为 (PC + 2) + 2 * Wn。

s 位用于选择源寄存器。

指令字数: 1

指令周期数: 4

例 1: 002000 HERE: BRA W7 ; 向前转移 (2+2*W7)
 002002 . . .

 002108 . . .
 00210A TABLE7: . . .
 00210C . . .

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 210A |
| W7 | 0084 | W7 | 0084 |
| SR | 0000 | SR | 0000 |

BRA C

如果进位位为 1 则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA C, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = C
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0001 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果进位标志位为 1, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个 16 位有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA C, CARRY    ; 如果 C 为 1, 转移到 CARRY
002002 NO_C:    . . .          ; 否则继续执行
002004          . . .
002006          GOTO THERE
002008 CARRY:   . . .
00200A          . . .
00200C THERE:   . . .
00200E          . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2008 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

例 2: 002000 HERE: BRA C, CARRY ; 如果 C 为 1, 转移到 CARRY
 002002 NO_C: ... ; 否则继续执行
 002004 ...
 002006 GOTO THERE
 002008 CARRY: ...
 00200A ...
 00200C THERE: ...
 00200E ...

| | 指令执行前 | | 指令执行后 |
|----|---------|----|---------|
| PC | 00 2000 | PC | 00 2002 |
| SR | 0000 | SR | 0000 |

例 3: 006230 HERE: BRA C, CARRY ; 如果 C 为 1, 转移到 CARRY
 006232 NO_C: ... ; 否则继续执行
 006234 ...
 006236 GOTO THERE
 006238 CARRY: ...
 00623A ...
 00623C THERE: ...
 00623E ...

| | 指令执行前 | | 指令执行后 |
|----|--------------|----|--------------|
| PC | 00 6230 | PC | 00 6238 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

例 4: 006230 START: ...
 006232 ...
 006234 CARRY: ...
 006236 ...
 006238 ...
 00623A ...
 00623C HERE: BRA C, CARRY ; 如果 C 为 1, 转移到 CARRY
 00623E ... ; 否则继续执行

| | 指令执行前 | | 指令执行后 |
|----|--------------|----|--------------|
| PC | 00 623C | PC | 00 6234 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA GE

如果有符号大于或等于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA GE, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = (N&&OV)||(!N&&!OV)
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1101 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 (N&&OV)||(!N&&!OV) 为真，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个 16 位有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

注: 汇编器将把指定的标号转换为要使用的偏移量。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
007600 LOOP:      . . .
007602             . . .
007604             . . .
007606             . . .
007608 HERE:      BRA GE, LOOP          ; 如果 GE，转移到 LOOP
00760A NO_GE:     . . .                ; 否则继续执行
```

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 7608 | PC | 00 7600 |
| SR | 0000 | SR | 0000 |

例 2:

007600 LOOP: . . .
007602 . . .
007604 . . .
007606 . . .
007608 HERE: BRA GE, LOOP ; 如果 GE, 转移到 LOOP
00760A NO_GE: . . . ; 否则继续执行

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 00 7608 | PC | 00 760A |
| SR | 0008 (N = 1) | SR | 0008 (N = 1) |

BRA GEU

如果无符号大于或等于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA GEU, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16 偏移量，可支持 [-32768 ... +32767] 程序字的偏移量范围。

操作: 条件 = C
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0001 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果进位标志位为 1，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个 16 位有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

注: 该指令等同于 BRA C, Expr（如果进位位为 1 则转移）指令，且具有相同的编码。对其进行反汇编将得到 BRA C, Slit16。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1: 002000 HERE: BRA GEU, BYPASS ; 如果 C 为 1，转移到 BYPASS
002002 NO_GEU: . . . ; 否则继续执行
002004 . . .
002006 . . .
002008 . . .
00200A GOTO THERE
00200C BYPASS: . . .
00200E . . .

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200C |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA GT

如果有符号大于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA GT, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = (!Z&&N&&OV)||(!Z&&!N&&!OV)
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1100 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 (!Z&&N&&OV)||(!Z&&!N&&!OV) 为真, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个 16 位有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA GT, BYPASS          ; 如果 GT, 转移到 BYPASS
002002 NO_GT:   . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| | | | |
|----|--------------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 200C |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA GTU 如果无符号大于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA GTU, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = (C&&!Z)
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1110 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 (C&&!Z) 为真，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA GTU, BYPASS      ; 如果 GTU，转移到 BYPASS
002002 NO_GTU:  . . .                ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200C |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA LE

如果有符号小于或等于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA LE, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = Z|!(N&&!OV)||(!N&&OV)
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0100 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 (Z|!(N&&!OV)||(!N&&OV)) 为真，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:      BRA LE, BYPASS          ; 如果 LE, 转移到 BYPASS
002002 NO_LE:     . . .                  ; 否则继续执行
002004           . . .
002006           . . .
002008           . . .
00200A           GOTO THERE
00200C BYPASS:    . . .
00200E           . . .
```

| | | | |
|----|--------------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA LEU 如果无符号小于或等于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA LEU, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = !C||Z
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0110 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 (!C||Z) 为真，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:      BRA LEU, BYPASS          ; 如果 LEU, 转移到 BYPASS
002002 NO_LEU:    . . .                    ; 否则继续执行
002004            . . .
002006            . . .
002008            . . .
00200A            GOTO THERE
00200C BYPASS:    . . .
00200E            . . .
```

| | | | |
|----|--------------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 200C |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA LT

如果有符号小于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA LT, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = (N&&!OV)||(!N&&OV)
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0101 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果逻辑表达式 ((N&&!OV)||(!N&&OV)) 为真, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:      BRA LT, BYPASS          ; 如果 LT, 转移到 BYPASS
002002 NO_LT:     . . .                  ; 否则继续执行
002004             . . .
002006             . . .
002008             . . .
00200A             GOTO THERE
00200C BYPASS:    . . .
00200E             . . .
```

| | | | |
|----|--------------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA LTU 如果无符号小于则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA LTU, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = IC
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1001 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果进位标志位为 0, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

注: 该指令等同于 BRA NC, Expr (如果进位位为零则转移) 指令, 且具有相同的编码。对其进行反汇编将得到 BRA NC, Slit16。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA LTU, BYPASS      ; 如果 LTU, 转移到 BYPASS
002002 NO_LTU:  . . .                ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA N

如果为负则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA N, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = N
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0011 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果负标志位为 1, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA N, BYPASS          ; 如果 N 为 1, 转移到 BYPASS
002002 NO_N:    . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| | | | |
|----|--------------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 200C |
| SR | 0008 (N = 1) | SR | 0008 (N = 1) |

BRA NC 如果进位位为零则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA NC, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = !C
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1001 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果进位标志位为 0，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA NC, BYPASS          ; 如果 NC，转移到 BYPASS
002002 NO_NC:   . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

BRA NN

如果非负则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA NN, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = !N
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1011 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果负标志位为 0，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:      BRA NN, BYPASS          ; 如果 NN，转移到 BYPASS
002002 NO_NN:     . . .                    ; 否则继续执行
002004           . . .
002006           . . .
002008           . . .
00200A           GOTO THERE
00200C BYPASS:    . . .
00200E           . . .
```

| | | | |
|----|---------|----|---------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 200C |
| SR | 0000 | SR | 0000 |

BRA NOV 如果未溢出则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA NOV, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = !OV
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1000 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果溢出标志位为 0，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

```
例 1: 002000 HERE:      BRA NOV, BYPASS          ; 如果 NOV，转移到 BYPASS
      002002 NO_NOV: . . .                  ; 否则继续执行
      002004        . . .
      002006        . . .
      002008        . . .
      00200A        GOTO THERE
      00200C BYPASS: . . .
      00200E        . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200C |
| SR | 0008 (N = 1) | SR | 0008 (N = 1) |

BRA NZ

如果非零则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA NZ, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = !Z
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 1010 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果 Z 标志位为 0, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

```
002000 HERE:    BRA NZ, BYPASS          ; 如果 NZ, 转移到 BYPASS
002002 NO_NZ:   . . .                  ; 否则继续执行
002004         . . .
002006         . . .
002008         . . .
00200A        GOTO THERE
00200C BYPASS:  . . .
00200E         . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0002 (Z = 1) | SR | 0002 (Z = 1) |

BRA OA 如果累加器 A 溢出则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } BRA OA, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = OA
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1100 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果累加器 A 溢出标志位为 1, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

注: 汇编器将把指定的标号转换为要使用的偏移量。

指令字数: 1
指令周期数: 1 (如果执行转移操作, 则为 2) ——dsPIC30F 和 dsPIC33F
1 (如果执行转移操作, 则为 4) ——dsPIC33E

例 1:

```
002000 HERE:    BRA OA, BYPASS          ; 如果 OA 为 1, 转移到 BYPASS
002002 NO_OA:   . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| 指令执行前 | | 指令执行后 | |
|---------------|---------|---------------|---------|
| PC | 00 2000 | PC | 00 200C |
| SR | 8800 | SR | 8800 |
| (OA, OAB = 1) | | (OA, OAB = 1) | |

BRA OB

如果累加器 B 溢出则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 ; } BRA OB, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = OB
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1101 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果累加器 B 溢出标志位为 1，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——dsPIC33E

例 1:

```
002000 HERE:    BRA OB, BYPASS          ; 如果 OB 为 1，转移到 BYPASS
002002 NO_OB:   . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| 指令执行前 | | 指令执行后 | |
|-------|--------------------|-------|--------------------|
| PC | 00 2000 | PC | 00 2002 |
| SR | 8800 (OA, OAB = 1) | SR | 8800 (OA, OAB = 1) |

BRA OV 如果溢出则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA OV, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = OV
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0000 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果溢出标志位为 1，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F
1（如果执行转移操作，则为 4）——PIC24E 和 dsPIC33E

```
例 1: 002000 HERE:      BRA OV, BYPASS          ; 如果 OV 为 1，转移到 BYPASS
      002002 NO_OV   . . .                  ; 否则继续执行
      002004         . . .
      002006         . . .
      002008         . . .
      00200A         GOTO THERE
      00200C BYPASS: . . .
      00200E         . . .
```

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0002 (Z = 1) | SR | 0002 (Z = 1) |

BRA SA

如果累加器 A 饱和则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } BRA SA, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = SA
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1110 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果累加器 A 饱和标志位为 1，程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16，可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作，由于 PC 将已递增以取出下一条指令，因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时，执行该指令需要两个指令周期，其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数，用于指定自 (PC + 2) 的偏移量（指令字数）。

指令字数: 1

指令周期数: 1（如果执行转移操作，则为 2）——dsPIC30F 和 dsPIC33F

1（如果执行转移操作，则为 4）——dsPIC33E

例 1:

```
002000 HERE:    BRA SA, BYPASS          ; 如果 SA 为 1，转移到 BYPASS
002002 NO_SA:   . . .                  ; 否则继续执行
002004          . . .
002006          . . .
002008          . . .
00200A          GOTO THERE
00200C BYPASS:  . . .
00200E          . . .
```

| 指令执行前 | | 指令执行后 | |
|---------------|---------|---------------|---------|
| PC | 00 2000 | PC | 00 200C |
| SR | 2400 | SR | 2400 |
| (SA, SAB = 1) | | (SA, SAB = 1) | |

BRA SB 如果累加器 B 饱和则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } BRA SB, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = SB
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1111 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果累加器 B 饱和标志位为 1, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1
指令周期数: 1 (如果执行转移操作, 则为 2) ——dsPIC30F 和 dsPIC33F
1 (如果执行转移操作, 则为 4) ——dsPIC33E

```
例 1: 002000 HERE:    BRA SB, BYPASS      ; 如果 SB 为 1, 转移到 BYPASS
      002002 NO_SB: . . .              ; 否则继续执行
      002004      . . .
      002006      . . .
      002008      . . .
      00200A      GOTO THERE
      00200C BYPASS: . . .
      00200E      . . .
```

| | | | |
|----|---------|----|---------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 2002 |
| SR | 0000 | SR | 0000 |

BRA Z

如果为零则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BRA Z, Expr

操作数: Expr 可以是标号、绝对地址或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 条件 = Z
如果条件为真
(PC + 2) + 2 * Slit16 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0011 | 0010 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 如果零标志位为 1, 程序将相对于下一个 PC 值进行转移。转移操作的偏移量为二进制补码值 2 * Slit16, 可支持最多向前或向后转移 32K 个指令字。Slit16 值由链接器根据提供的标号、绝对地址或表达式进行解析得出。

如果执行转移操作, 由于 PC 将已递增以取出下一条指令, 因此新 PC 值将为 (PC + 2) + 2 * Slit16。此时, 执行该指令需要两个指令周期, 其中第二个指令周期执行一条 NOP 指令。

n 位构成一个有符号立即数, 用于指定自 (PC + 2) 的偏移量 (指令字数)。

指令字数: 1

指令周期数: 1 (如果执行转移操作, 则为 2) ——PIC24F、PIC24H、dsPIC30F 和 dsPIC33F

1 (如果执行转移操作, 则为 4) ——PIC24E 和 dsPIC33E

例 1:

| | | | |
|--------|---------|---------------|------------------------|
| 002000 | HERE: | BRA Z, BYPASS | ; 如果 Z 为 1, 转移到 BYPASS |
| 002002 | NO_Z: | . . . | ; 否则继续执行 |
| 002004 | | . . . | |
| 002006 | | . . . | |
| 002008 | | . . . | |
| 00200A | | GOTO THERE | |
| 00200C | BYPASS: | . . . | |
| 00200E | | . . . | |

| | | | |
|-------|--------------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200C |
| SR | 0002 (Z = 1) | SR | 0002 (Z = 1) |

BSET

将 f 中的指定位置 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 ;} BSET{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 1 → f<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|-------|
| 指令编码: | 1010 | 1000 | bbbf | ffff | ffff | ffffb |
|-------|------|------|------|------|------|-------|

描述: 将文件寄存器 f 中由 bit4 指定的位置 1。位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

b 位用于选择指定要被置 1 的位位置的 bit4 值。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BSET.B 0x601, #0x3 ; 将 0x601 中的 bit 3 置 1

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 0600 | F234 | 数据单元 0600 | FA34 |
| SR | 0000 | SR | 0000 |

例 2: BSET 0x444, #0xF ; 将 0x444 中的 bit 15 置 1

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 0444 | 5604 | 数据单元 0444 | D604 |
| SR | 0000 | SR | 0000 |

BSET

将 Ws 中的指定位置 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BSET{.B} Ws, #bit4
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 1 → Ws<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0000 | bbbb | 0B00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将寄存器 Ws 中由 bit4 指定的位置 1。位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被置 1 的位位置的 bit4 值。
B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
p 位用于选择源地址模式。
s 位用于选择源 / 目标寄存器。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 源寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。
- 4: 在 dsPIC33E 和 PIC24E 器件中, 该指令使用 DSRPAG 寄存器来间接生成扩展数据空间中的地址。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BSET.B W3, #0x7 ; 将 W3 中的 bit 7 置 1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W3 | 0026 | W3 | 00A6 |
| SR | 0000 | SR | 0000 |

例 2: BSET [W4++], #0x0 ; 将 [W4] 中的 bit 0 置 1
 ; 指令执行后，递增 W4

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W4 | 6700 | W4 | 6702 |
| 数据单元 6700 | 1734 | 数据单元 6700 | 1735 |
| SR | 0000 | SR | 0000 |

BSW

写 Ws 中的某位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BSW.C Ws, Wb
BSW.Z [Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]

操作: 对于 “.C” 操作:
C → Ws<(Wb)>
对于 “.Z” 操作 (默认):
Z → Ws<(Wb)>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1101 | Zwww | w000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 向寄存器 Ws 中的 (Wb) 位写入状态寄存器中 C 或 Z 标志位的值。位编号从工作寄存器的最低位 (bit 0) 开始, 向最高位 (bit 15) 递增。只有 Wb 中的低 4 位用于确定目标位的编号。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 进行寻址。

Z 位用于选择 C 或 Z 标志作为源操作数。
w 位用于选择位选择寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 该指令只能工作于字模式。如果未提供扩展符, 可认为该指令将执行 “.z” 操作。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BSW.C W2, W3 ; 将 W2 中的 bit W3 设定为 C 位的值

| 指令执行前 | | | 指令执行后 | | |
|-------|------|----------------|-------|------|----------------|
| W2 | F234 | | W2 | 7234 | |
| W3 | 111F | | W3 | 111F | |
| SR | 0002 | (Z = 1, C = 0) | SR | 0002 | (Z = 1, C = 0) |

BTG

将 f 中的指定位翻转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 ;} BTG{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: (f)<bit4> → (f)<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1010 | bbbf | ffff | ffff | fffb |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器 f 中由 bit4 指定的位翻转 (求反)。对于 bit4 操作数, 位编号从字节的最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

b 位用于选择指定要被翻转的位位置的 bit4 值。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BTG.B 0x1001, #0x4 ; 将 0x1001 中的 bit 4 翻转

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 1000 | F234 | 数据单元 1000 | E234 |
| SR | 0000 | SR | 0000 |

例 2: BTG 0x1660, #0x8 ; 将 RAM660 中的 bit 8 翻转

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 1660 | 5606 | 数据单元 1660 | 5706 |
| SR | 0000 | SR | 0000 |

BTG

将 Ws 中的指定位翻转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTG{.B} Ws, #bit4
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: (Ws)<bit4> → Ws<bit4>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0010 | bbbb | 0B00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将寄存器 Ws 中由 bit4 指定的位翻转（求反）。对于 bit4 操作数, 位编号从最低位（bit 0）开始, 向最高位（对于字节操作为 bit 7, 对于字操作为 bit 15）递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被翻转的位位置的 bit4 值。
B 位用于选择字节或字操作（0 选择字操作, 1 选择字节操作）。
s 位用于选择源 / 目标寄存器。
p 位用于选择源地址模式。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 源寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。
- 4: 在 dsPIC33E 和 PIC24E 器件中, 该指令使用 DSRPAG 寄存器来间接生成扩展数据空间中的地址。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: BTG W2, #0x0 ; 将 W2 中的 bit 0 翻转

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W2 | F234 | W2 | F235 |
| SR | 0000 | SR | 0000 |

例 2: BTG [W0++], #0x0 ; 将[W0]中的 bit 0 翻转
; 指令执行后, 递增 W0

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 2300 | W0 | 2302 |
| 数据单元 2300 | 5606 | 数据单元 2300 | 5607 |
| SR | 0000 | SR | 0000 |

BTSC 测试 f 中的指定位，为零则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTSC{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 测试 (f)<bit4>, 如果为零则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1111 | bbbf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 对文件寄存器中由 bit4 指定的位进行测试。如果被测试位为 0，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。如果被测试位为 1，将顺序执行下一条指令。在上述两种情况下，文件寄存器中的内容都不会改变。对于 bit4 操作数，位编号从最低位（bit 0）开始，向最高位（对于字节操作为 bit 7，对于字操作为 bit 15）递增。

- b 位用于选择指定要被测试的位位置的 bit4 值。
f 位用于选择文件寄存器的地址。
- 注 1:** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
2: 当该指令工作于字模式时，文件寄存器地址必须是字对齐的。
3: 当该指令工作于字节模式时，bit4 必须在 0 和 7 之间。

指令字数: 1
指令周期数: 1 (2 或 3) (1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: 002000 HERE: BTSC.B 0x1201, #2 ; 如果 0x1201 中的 bit 2 为 0,
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| | | | | | |
|-----------|---------|--|-----------|---------|--|
| 指令执行前 | | | 指令执行后 | | |
| PC | 00 2000 | | PC | 00 2002 | |
| 数据单元 1200 | 264F | | 数据单元 1200 | 264F | |
| SR | 0000 | | SR | 0000 | |

例 2: 002000 HERE: BTSC 0x804, #14 ; 如果 0x804 中的 bit 14 为 0,
 002002 GOTO BYPASS ; 则跳过 GOTO
 002004 . . .
 002006 . . .
 002008 BYPASS: . . .
 00200A . . .

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 2000 | PC | 00 2006 |
| 数据单元 0804 | 2647 | 数据单元 0804 | 2647 |
| SR | 0000 | SR | 0000 |

BTSC 测试 Ws 中的指定位，为零则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTSC Ws, #bit4
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

操作: 测试 (Ws)<bit4>, 如果为零则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0111 | bbbb | 0000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 对 Ws 中由 bit4 指定的位进行测试。如果被测试位为 0，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。如果被测试位为 1，将顺序执行下一条指令。在上述两种情况下，Ws 中的内容都不会改变。对于 bit4 操作数，位编号从字的最低位（bit 0）开始，向最高位（bit 15）递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被测试的位位置的 bit4 值。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 该指令只能工作于字模式。

指令字数: 1
指令周期数: 1（如果下一条指令被跳过，则为 2 或 3）⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: 002000 HERE: BTSC W0, #0x0 ; 如果 W0 中的 bit 0 为 0，
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 2002 |
| W0 | 264F | W0 | 264F |
| SR | 0000 | SR | 0000 |

例 2:

002000 HERE: BTSC W6, #0xF ; 如果 W6 中的 bit 15 为 0,
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 2000 | PC | 00 2006 |
| W6 | 264F | W6 | 264F |
| SR | 0000 | SR | 0000 |

例 3:

003400 HERE: BTSC [W6++], #0xC ; 如果 [W6] 中的 bit 12 为 0,
003402 GOTO BYPASS ; 则跳过 GOTO
003404 . . . ; 指令执行后, 递增 W6
003406 . . .
003408 BYPASS: . . .
00340A . . .

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 3400 | PC | 00 3402 |
| W6 | 1800 | W6 | 1802 |
| 数据单元 1800 | 1000 | 数据单元 1800 | 1000 |
| SR | 0000 | SR | 0000 |

BTSS

测试 f 中的指定位，为 1 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTSS{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: 测试 (f)<bit4>, 如果为 1 则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1110 | bbbf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 对文件寄存器 f 中由 bit4 指定的位进行测试。如果被测试位为 1, 将放弃执行下一条指令 (在当前指令执行过程中取出), 而在下一个指令周期内执行一条 NOP 指令。如果被测试位为 0, 将顺序执行下一条指令。在上述两种情况下, 文件寄存器中的内容都不会改变。对于 bit4 操作数, 位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

- b 位用于选择指定要被测试的位位置的 bit4 值。
f 位用于选择文件寄存器的地址。
- 注 1:** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。

指令字数: 1

指令周期数: 1 (如果下一条指令被跳过, 则为 2 或 3) (1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: 007100 HERE: BTSS.B 0x1401, #0x1 ; 如果 0x1401 中的 bit 1 为 1,
 007102 CLR WREG ; 则不对 WREG 进行清零
 007104 . . .

| 指令执行前 | | | 指令执行后 | | |
|-----------|---------|--|-----------|---------|--|
| PC | 00 7100 | | PC | 00 7104 | |
| 数据单元 1400 | 0280 | | 数据单元 1400 | 0280 | |
| SR | 0000 | | SR | 0000 | |

例 2: 007100 HERE: BTSS 0x890, #0x9 ; 如果 0x890 中的 bit 9 为 1,
 007102 GOTO BYPASS ; 则跳过 GOTO
 007104 . . .
 007106 BYPASS: . . .

| 指令执行前 | | | 指令执行后 | | |
|-----------|---------|--|-----------|---------|--|
| PC | 00 7100 | | PC | 00 7102 | |
| 数据单元 0890 | 00FE | | 数据单元 0890 | 00FE | |
| SR | 0000 | | SR | 0000 | |

BTSS 测试 Ws 中的指定位，为 1 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTSS Ws, #bit4
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

操作: 测试 (Ws)<bit4>, 如果为 1 则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0110 | bbbb | 0000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 对 Ws 中由 bit4 指定的位进行测试。如果被测试位为 1，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。如果被测试位为 0，将顺序执行下一条指令。在上述两种情况下，Ws 中的内容都不会改变。对于 bit4 操作数，位编号从字的最低位（bit 0）开始，向最高位（bit 15）递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被测试的位位置的 bit4 值。
s 位用于选择源寄存器。
p 位用于选择源地址模式。

注: 该指令只能工作于字模式。

指令字数: 1

指令周期数: 1（如果下一条指令被跳过，则为 2 或 3）(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: 002000 HERE: BTSS W0, #0x0 ; 如果 W0 中的 bit 0 为 1,
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 2000 | PC | 00 2006 |
| W0 | 264F | W0 | 264F |
| SR | 0000 | SR | 0000 |

例 2:

002000 HERE: BTSS W6, #0xF ; 如果 W6 中的 bit 15 为 1,
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 2000 | PC | 00 2002 |
| W6 | 264F | W6 | 264F |
| SR | 0000 | SR | 0000 |

例 3:

003400 HERE: BTSS [W6++], 0xC ; 如果 [W6] 中的 bit 12 为 1,
003402 GOTO BYPASS ; 则跳过 GOTO
003404 . . . ; 指令执行后, 递增 W6
003406 . . .
003408 BYPASS: . . .
00340A . . .

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 3400 | PC | 00 3406 |
| W6 | 1800 | W6 | 1802 |
| 数据单元 1800 | 1000 | 数据单元 1800 | 1000 |
| SR | 0000 | SR | 0000 |

BTST测试 f 中的指定位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTST{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: (f)<bit4> → Z

受影响的状态位: Z

| | | | | | | |
|-------|------|------|------|------|------|-------|
| 指令编码: | 1010 | 1011 | bbbf | ffff | ffff | ffffb |
|-------|------|------|------|------|------|-------|

描述: 对文件寄存器 f 中由 bit4 指定的位进行测试, 并将被测试位的反码值存储到状态寄存器中的 Z 标志位。文件寄存器中的内容将不会改变。对于 bit4 操作数, 位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

b 位用于选择指定要被测试的位位置的 bit4 值。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BTST.B 0x1201, #0x3 ; 设定 Z = 0x1201 中 bit 3 的反码值

| | | | |
|-----------|------|-----------|--------------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 1200 | F7FF | 数据单元 1200 | F7FF |
| SR | 0000 | SR | 0002 (Z = 1) |

例 2: BTST 0x1302, #0x7 ; 设定 Z = 0x1302 中 bit 7 的反码值

| | | | |
|-----------|--------------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 1302 | F7FF | 数据单元 1302 | F7FF |
| SR | 0002 (Z = 1) | SR | 0000 |

BTST

测试 Ws 中的指定位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTST.C Ws, #bit4
BTST.Z [Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

操作: 对于 “.C” 操作:
(Ws)<bit4> → C
对于 “.Z” 操作 (默认):
(Ws)<bit4> → Z

受影响的状态位: Z 或 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0011 | bbbb | z000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 对寄存器 Ws 中由 bit4 指定的位进行测试。如果指定指令中的 “.z” 选项，被测试位的反码值将被存储到状态寄存器中的零标志位。如果指定指令中的 “.c” 选项，被测试位的值将被存储到状态寄存器中的进位标志位。在上述两种情况下，Ws 中的内容都不会改变。

对于 bit4 操作数，位编号从字的最低位 (bit 0) 开始，向最高位 (bit 15) 递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

b 位用于选择指定要被测试的位位置的 bit4 值。

Z 位用于选择 C 或 Z 标志作为目标位。

p 位用于选择源地址模式。

s 位用于选择源寄存器。

注: 该指令只能工作于字模式。如果未提供扩展符，可认为该指令将执行 “.z” 操作。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1:

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|------|
| W0 | 1200 | W0 | 1202 |
| 数据单元 1200 | FFF7 | 数据单元 1200 | FFF7 |
| SR | 0001 (C = 1) | SR | 0000 |

例 2:

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | F234 | W0 | F234 |
| SR | 0000 | SR | 0002 (Z = 1) |

BTST

测试 Ws 中的指定位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTST.C Ws, Wb
BTST.Z [Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]

操作: 对于 “.C” 操作:
(Ws)<(Wb)> → C
对于 “.Z” 操作 (默认):
(Ws)<(Wb)> → Z

受影响的状态位: Z 或 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0101 | Zwww | w000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 对寄存器 Ws 中的 (Wb) 位进行测试。如果指定指令中的 “.c” 选项，被测试位的值将被存储到状态寄存器中的进位标志位。如果指定指令中的 “.z” 选项，被测试位的反码值将被存储到状态寄存器中的零标志位。在上述两种情况下，Ws 中的内容都不会改变。

只有 Wb 中的低 4 位用于确定位的编号。位编号从工作寄存器的最低位 (bit 0) 开始，向最高位 (bit 15) 递增。可使用寄存器直接或间接寻址对 Ws 进行寻址。

Z 位用于选择 C 或 Z 标志作为目标位。
w 位用于选择位选择寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 该指令只能工作于字模式。如果未提供扩展符，可认为该指令将执行 “.z” 操作。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节 “多周期指令” 中的注 3。

BTSTS

测试 / 置 1 f 中的指定位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } BTSTS{.B} f, #bit4

操作数: 对于字节操作, f ∈ [0 ... 8191]
对于字操作, f ∈ [0 ... 8190] (仅限偶数)
对于字节操作, bit4 ∈ [0 ... 7]
对于字操作, bit4 ∈ [0 ... 15]

操作: (f)<bit4> → Z
1 → (f)<bit4>

受影响的状态位: Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 1100 | bbbf | ffff | ffff | fffb |
|-------|------|------|------|------|------|------|

描述: 对文件寄存器 f 中由 bit4 指定的位进行测试, 并将被测试位的反码值存储到状态寄存器中的零标志位。文件寄存器的被测试位随后将被置 1。对于 bit4 操作数, 位编号从最低位 (bit 0) 开始, 向最高位 (对于字节操作为 bit 7, 对于字操作为 bit 15) 递增。

b 位用于选择指定要被测试 / 置 1 的位位置的 bit4 值。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 当该指令工作于字模式时, 文件寄存器地址必须是字对齐的。
- 3: 当该指令工作于字节模式时, bit4 必须在 0 和 7 之间。
- 4: 文件寄存器 f 一定不能是 CPU 状态寄存器 (SR)。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BTSTS.B 0x1201, #0x3 ; 设定 Z = 0x1201 中 bit 3 的反码值,
; 然后设定 0x1201 中的 bit 3 = 1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| 数据单元 1200 | F7FF | 数据单元 1200 | FFFF |
| SR | 0000 | SR | 0002 (Z = 1) |

例 2:

| | | |
|-------|------------|---|
| BTSTS | 0x808, #15 | ; 设定 Z = 0x808 中 bit 15 的反码值, ; 然后设定 0x808 中的 bit 15 = 1 |
|-------|------------|---|

| 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|------|
| RAM300 | 8050 | RAM300 | 8050 |
| SR | 0002 (Z = 1) | SR | 0000 |

BTSTS

测试 / 置 1 Ws 中的指定位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} BTSTS.C Ws, #bit4
BTSTS.Z [Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
bit4 ∈ [0 ... 15]

操作: 对于 “.C” 操作:
(Ws)<bit4> → C
1 → Ws<bit4>
对于 “.Z” 操作 (默认):
(Ws)<bit4> → Z
1 → Ws<bit4>

受影响的状态位: Z 或 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1010 | 0100 | bbbb | z000 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 对寄存器 Ws 中由 bit4 指定的位进行测试。如果指定指令中的 “.z” 选项，被测试位的反码值将被存储到状态寄存器中的零标志位。如果指定指令中的 “.c” 选项，被测试位的值将被存储到状态寄存器中的进位标志位。在上述两种情况下，Ws 中的被测试位都将被置 1。

b 位用于选择指定要被测试 / 置 1 的位位置的 bit4 值。

Z 位用于选择 C 或 Z 标志作为目标位。

p 位用于选择源地址模式。

s 位用于选择源寄存器。

注 1: 该指令只能工作于字模式。如果未提供扩展符，可认为该指令将执行 “.z” 操作。

2: 如果使用 Ws 作为指针，则它一定不能包含 CPU 状态寄存器 (SR) 的地址。

3: 在 dsPIC33E 和 PIC24E 器件中，该指令使用 DSRPAG 寄存器来间接生成扩展数据空间中的地址。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: BTSTS.C [W0++], #0x3 ; 设定 C = [W0] 中的 bit 3
 ; 设定 [W0] 中的 bit 3 = 1
 ; 指令执行后，递增 W0

| 指令执行前 | | | 指令执行后 | | |
|-----------|----|--------------|-----------|----|------|
| 数据单元 1200 | W0 | 1200 | 数据单元 1200 | W0 | 1202 |
| | | | | | |
| | SR | 0001 (C = 1) | | SR | 0000 |

例 2: BTSTS.Z W0, #0x7 ; 设定 Z = W0 中 bit 7 的反码值，
 ; 且设定 W0 中的 bit 7 = 1

| 指令执行前 | | | 指令执行后 | | |
|-------|----|------|-------|----|--------------|
| W0 | | F234 | W0 | | F2BC |
| | SR | 0000 | | SR | 0002 (Z = 1) |

CALL

调用子程序

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CALL Expr

操作数: Expr 可以是标号或表达式（但不能为立即数）。
Expr 由链接器解析为一个 lit23，其中 lit23 ∈ [0 ... 8388606]。

操作: (PC) + 4 → PC
(PC<15:0>) → (TOS)
(W15) + 2 → W15
(PC<23:16>) → (TOS)
(W15) + 2 → W15
lit23 → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0010 | nnnn | nnnn | nnnn | nnn0 |
| 第一个字 | 0000 | 0000 | 0000 | 0000 | 0nnn | nnnn |
| 第二个字 | | | | | | |

描述: 在整个 4 MB 指令程序存储空间内直接调用子程序。在进行调用之前，24 位返回地址 (PC + 4) 被压入堆栈。在返回地址入栈后，23 位值 lit23 被装入 PC。
n 位形成目标地址。

注: 链接器将把指定的表达式解析为要使用的 lit23。

指令字数: 2
指令周期数: 2

例 1: 026000 CALL _FIR ; 调用 _FIR 子程序
026004 MOV W0, W1
.
.
026844 _FIR: MOV #0x400, W2 ; _FIR 子程序起始
026846 .

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 02 6000 | PC | 02 6844 |
| W15 | A268 | W15 | A26C |
| 数据单元 A268 | FFFF | 数据单元 A268 | 6004 |
| 数据单元 A26A | FFFF | 数据单元 A26A | 0002 |
| SR | 0000 | SR | 0000 |

例 2:

072000CALL _G66 ; 调用子程序 _G66
072004MOV W0, W1
.
077A28 _G66: INC W6, [W7++] ; 子程序起始
077A2A
077A2C

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 07 2000 | PC | 07 7A28 |
| W15 | 9004 | W15 | 9008 |
| 数据单元 9004 | FFFF | 数据单元 9004 | 2004 |
| 数据单元 9006 | FFFF | 数据单元 9006 | 0007 |
| SR | 0000 | SR | 0000 |

CALL

调用子程序

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CALL Expr

操作数: Expr 可以是标号或表达式（但不能为立即数）。
Expr 由链接器解析为一个 lit23，其中 lit23 ∈ [0 ... 8388606]。

操作: (PC) + 4 → PC
(PC<15:1>) → TOS<15:1>, SFA 位 → TOS<0>
(W15) + 2 → W15
(PC<23:16>) → TOS
(W15) + 2 → W15
0 → SFA 位
lit23 → PC
NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | |
|-------|------|------|------|------|-----------|
| 指令编码: | | | | | |
| 第一个字 | 0000 | 0010 | nnnn | nnnn | nnn0 |
| 第二个字 | 0000 | 0000 | 0000 | 0000 | 0nnn nnnn |

描述: 在整个 4 MB 指令程序存储空间内直接调用子程序。在进行调用之前，24 位返回地址 (PC + 4) 被压入堆栈。在返回地址入栈后，23 位值 lit23 被装入 PC。

n 位形成目标地址。

注: 链接器将把指定的表达式解析为要使用的 lit23。

指令字数: 2
指令周期数: 4

例 1:

026000CALL _FIR ; 调用 _FIR 子程序

026004MOV W0, W1

...

...

026844_FIR: MOV #0x400, W2 ; _FIR 子程序起始

026846...

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 02 6000 | PC | 02 6844 |
| W15 | A268 | W15 | A26C |
| 数据单元 A268 | FFFF | 数据单元 A268 | 6004 |
| 数据单元 A26A | FFFF | 数据单元 A26A | 0002 |
| SR | 0000 | SR | 0000 |

例 2:

072000CALL _G66 ; 调用子程序 _G66

072004MOV W0, W1

...

...

077A28_G66: INC W6, [W7++] ; 子程序起始

077A2A...

077A2C

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 07 2000 | PC | 07 7A28 |
| W15 | 9004 | W15 | 9008 |
| 数据单元 9004 | FFFF | 数据单元 9004 | 2004 |
| 数据单元 9006 | FFFF | 数据单元 9006 | 0007 |
| SR | 0000 | SR | 0000 |

CALL

间接调用子程序

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CALL Wn

操作数: Wn ∈ [W0 ... W15]

操作:
(PC) + 2 → PC
(PC<15:0>) → TOS
(W15) + 2 → W15
(PC<23:16>) → TOS
(W15) + 2 → W15
0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0000 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 在程序存储空间的前 32K 指令字范围内间接调用子程序。在进行调用之前，24 位返回地址 (PC + 2) 被压入堆栈。在返回地址入栈之后，Wn<15:1> 被装入 PC<15:1> 而 PC<22:16> 被清零。由于 PC<0> 总是为 0，Wn<0> 将被忽略。

s 位用于选择源寄存器。

指令字数: 1

指令周期数: 2

例 1: 001002 CALL W0 ; 使用 W0 间接调用 BOOT 子程序
 001004 ...

 001600 _BOOT: MOV #0x400, W2 ; _BOOT 由此起始
 001602 MOV #0x300, W6

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 1002 | PC | 00 1600 |
| W0 | 1600 | W0 | 1600 |
| W15 | 6F00 | W15 | 6F04 |
| 数据单元 6F00 | FFFF | 数据单元 6F00 | 1004 |
| 数据单元 6F02 | FFFF | 数据单元 6F02 | 0000 |
| SR | 0000 | SR | 0000 |

```
例 2:      004200      CALL  W7          ; 使用 W7 间接调用 TEST 子程序
           004202      ...
           .           ...
           005500 _TEST: INC   W1, W2      ; _TEST 由此起始
           005502      DEC   W1, W3      ;
           .           ...
```

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 00 4200 | PC | 00 5500 |
| W7 | 5500 | W7 | 5500 |
| W15 | 6F00 | W15 | 6F04 |
| 数据单元 6F00 | FFFF | 数据单元 6F00 | 4202 |
| 数据单元 6F02 | FFFF | 数据单元 6F02 | 0000 |
| SR | 0000 | SR | 0000 |

CALL 间接调用子程序

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CALL Wn

- 操作数: Wn ∈ [W0 ... W15]
- 操作:
- (PC) + 2 → PC
 - (PC<15:1>) → TOS, SFA 位 → TOS<0>
 - (W15) + 2 → W15
 - (PC<23:16>) → TOS
 - (W15) + 2 → W15
 - 0 → SFA 位
 - 0 → PC<22:16>
 - (Wn<15:1>) → PC<15:1>
 - NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0000 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 在程序存储空间的前 32K 指令字范围内间接调用子程序。在进行调用之前，24 位返回地址 (PC + 2) 被压入堆栈。在返回地址入栈之后，Wn<15:1> 被装入 PC<15:1> 而 PC<22:16> 被清零。由于 PC<0> 总是为 0，Wn<0> 将被忽略。

s 位用于选择源寄存器。

指令字数: 1

指令周期数: 4

例 1:

001002

001004

.

001600

001602

.

CALL

...

...

_BOOT:

MOV

...

W0

...

...

#0x400, W2

#0x300, W6

...

; 使用 W0 间接调用 BOOT 子程序

; _BOOT 由此起始

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 1002 | PC | 00 1600 |
| W0 | 1600 | W0 | 1600 |
| W15 | 6F00 | W15 | 6F04 |
| 数据单元 6F00 | FFFF | 数据单元 6F00 | 1004 |
| 数据单元 6F02 | FFFF | 数据单元 6F02 | 0000 |
| SR | 0000 | SR | 0000 |

例 2:

004200

004202

.

005500

005502

.

CALL

...

...

_TEST:

INC

...

W7

...

...

W1, W2

W1, W3

...

; 使用 W7 间接调用 TEST 子程序

; _TEST 由此起始

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 4200 | PC | 00 5500 |
| W7 | 5500 | W7 | 5500 |
| W15 | 6F00 | W15 | 6F04 |
| 数据单元 6F00 | FFFF | 数据单元 6F00 | 4202 |
| 数据单元 6F02 | FFFF | 数据单元 6F02 | 0000 |
| SR | 0000 | SR | 0000 |

CALL.L

间接长调用子程序

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CALL.L Wn

操作数: Wn ∈ [W0, W2, W4, W6, W8, W10, W12]
操作: (PC) + 2 → PC,
(PC<15:1>) → TOS<15:1>, SFA 位 → TOS<0>
(W15) + 2 → W15
(PC<23:16>) → TOS,
(W15) + 2 → W15
0 → SFA 位,
PC<23> → PC<23> (见下文); (Wn+1)<6:0> → PC<22:16>;
(Wn) → PC<15:0>
NOP → 指令寄存器

| | |
|----------|--------------------------|
| 受影响的状态位: | SFA |
| 指令编码: | 000000011www00000000ssss |

描述: 对任意用户程序存储器地址进行间接子程序调用。首先, 返回地址 (PC+2) 和堆栈帧激活 (SFA) 位会被压入系统堆栈, 之后 SFA 位会被清零。然后, (Wn+1) 的低 7 位会被装入 PC<22:16>, 16 位值 (Wn) 会被装入 PC<15:0>。该指令不会修改 PC<23>。(Wn+1)<15:7> 的内容会被忽略。Wn<0> 的值也会被忽略, PC<0> 总是设定为 0。s 位用于指定 Wn 源寄存器的地址。w 位用于指定 Wn+1 源寄存器的地址。

指令字数: 1
指令周期数: 4

例 1: 026000 CALL.L W4 ; 调用 _FIR 子程序
026004 MOV W0, W1
.
.
.
026844 _FIR: MOV #0x400, W2 ; _FIR 子程序起始
026846 .

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 02 6000 | PC | 02 6844 |
| W4 | 6844 | W4 | 6844 |
| W5 | 0002 | W5 | 0002 |
| W15 | A268 | W15 | A26C |
| 数据单元 A268 | FFFF | 数据单元 A268 | 6004 |
| 数据单元 A26A | FFFF | 数据单元 A26A | 0002 |
| SR | 0000 | SR | 0000 |

CLR

清零 f 或 WREG

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } CLR{.B} f
WREG

操作数: f ∈ [0 ... 8191]

操作: 0 → 由 D 指定的目标寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1111 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 清零文件寄存器或默认工作寄存器 WREG 中的内容。如果指定了 WREG，WREG 将被清零。否则，指定的文件寄存器 f 将被清零。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1

例 1: CLR.B RAM200 ; 清零 RAM200（字节模式）

| | | | |
|--------|------|--------|------|
| 指令执行前 | | 指令执行后 | |
| RAM200 | 8009 | RAM200 | 8000 |
| SR | 0000 | SR | 0000 |

例 2: CLR WREG ; 清零 WREG（字模式）

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| WREG | 0600 | WREG | 0000 |
| SR | 0000 | SR | 0000 |

CLR

清零 Wd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} CLR{.B} Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wd ∈ [W0 ... W15]

操作: 0 → Wd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1011 | 0Bqq | qddd | d000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 清零寄存器 Wd 中的内容。可使用寄存器直接或间接寻址对 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

q 位用于选择目标地址模式。

d 位用于选择目标寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: CLR.B W2 ; 清零 W2（字节模式）

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W2 | 3333 | W2 | 3300 |
| SR | 0000 | SR | 0000 |

例 2: CLR [W0++] ; 清零 [W0]
; 指令执行后，递增 W0

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 2300 | W0 | 2302 |
| 数据单元 2300 | 5607 | 数据单元 2300 | 0000 |
| SR | 0000 | SR | 0000 |

指令字数: 1
指令周期数: 1

CLRWDT

清零看门狗定时器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } CLRWDT

操作数: 无

操作: 0 → WDT 计数寄存器
0 → WDT 预分频器 A 计数
0 → WDT 预分频器 B 计数

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 0110 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 清零看门狗定时器计数寄存器和预分频器计数寄存器的内容。但不会更改由 FWDT 中配置熔丝所设定的看门狗预分频器 A 和预分频器 B 的设置。

指令字数: 1

指令周期数: 1

例 1: CLRWDT ; 清零看门狗定时器



COM

对 f 的内容取反

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } COM{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $\overline{(f)} \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1110 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 计算文件寄存器内容的反码并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1:** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2:** WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: COM.b RAM200 ; 对 RAM200 求反（字节模式）

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|----------|
| RAM200 | 80FF | RAM200 | 8000 |
| SR | 0000 | SR | 0002 (Z) |

例 2: COM RAM400, WREG ; 对 RAM400 求反并将结果存入 WREG
;（字模式）

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|--------------|
| WREG | 1211 | WREG | F7DC |
| RAM400 | 0823 | RAM400 | 0823 |
| SR | 0000 | SR | 0008 (N = 1) |

COM对 Ws 的内容取反

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} COM{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1010 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 计算源寄存器 Ws 内容的反码并将结果存放在目标寄存器 Wd 中。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: COM.B [W0++], [W1++] ; 对 [w0] 求反并将结果存入 [w1]（字节模式）
; 指令执行后，递增 w0 和 w1

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|--|-----------|------|---------|
| W0 | 2301 | | W0 | 2302 | |
| W1 | 2400 | | W1 | 2401 | |
| 数据单元 2300 | 5607 | | 数据单元 2300 | 5607 | |
| 数据单元 2400 | ABCD | | 数据单元 2400 | ABA9 | |
| SR | 0000 | | SR | 0008 | (N = 1) |

例 2: COM W0, [W1++] ; 对 W0 求反并将结果存入 [W1]（字模式）
; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | D004 | W0 | D004 |
| W1 | 1000 | W1 | 1002 |
| 数据单元 1000 | ABA9 | 数据单元 1000 | 2FFB |
| SR | 0000 | SR | 0000 |

CP 比较 f 和 WREG 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } CP{.B} f

操作数: f ∈ [0 ...8191]

操作: (f) – (WREG)

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0011 | 0B0f | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 计算 (f) – (WREG) 并更新状态寄存器。该指令与 SUBWF 指令具有相同的效果，但并不保存减法的结果。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: CP.B RAM400 ; 将 RAM400 与 WREG 进行比较（字节模式）

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|--------------|
| WREG | 8823 | WREG | 8823 |
| RAM400 | 0823 | RAM400 | 0823 |
| SR | 0000 | SR | 0003 (C = 1) |

例 2: CP 0x1200 ; 将 (0x1200) 与 WREG 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| WREG | 2377 | WREG | 2377 |
| 数据单元 1200 | 2277 | 数据单元 1200 | 2277 |
| SR | 0000 | SR | 0008 (N = 1) |

CP比较 Wb 和 lit5 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} CP{.B} Wb, #lit5

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]

操作: (Wb) – lit5

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 0www | wB00 | 011k | kkkk |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) – lit5 并更新状态寄存器。该指令与 SUB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。

w 位用于选择 Wb 基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: CP.B W4, #0x12 ; 将 W4 与 0x12 进行比较（字节模式）

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W4 | 7711 | W4 | 7711 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: CP W4, #0x12 ; 将 W4 与 0x12 进行比较（字模式）

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W4 | 7713 | W4 | 7713 |
| SR | 0000 | SR | 0001 (C = 1) |

CP比较 Wb 和 lit8 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} CP{.B} Wb, #lit8

操作数: Wb ∈ [W0 ... W15]
lit8 ∈ [0 ... 255]

操作: (Wb) – lit8

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 0www | wBkk | k11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) – lit8 并更新状态寄存器。该指令与 SUB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。

w 位用于选择 Wb 基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: CP.B W4, #0x12 ; 将 W4 与 0x12 进行比较（字节模式）

| | | | |
|----|-------|----|-----------------|
| | 指令执行前 | | 指令执行后 |
| W4 | 7711 | W4 | 7711 |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: CP W4, #0x12 ; 将 W4 与 0x12 进行比较（字模式）

| | | | |
|----|-------|----|--------------|
| | 指令执行前 | | 指令执行后 |
| W4 | 7713 | W4 | 7713 |
| SR | 0000 | SR | 0001 (C = 1) |

CP比较 Wb 和 Ws 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} CP{.B} Wb, Ws
[Ws]
[Ws++]
[Ws--]
[++Ws]
[--Ws]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]

操作: (Wb) – (Ws)

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 0www | wB00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) – (Ws) 并更新状态寄存器。该指令与 SUB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 进行寻址。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
p 位用于选择源地址模式。
s 位用于选择 Ws 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: CP.B W0, [W1++] ; 将 [W1] 与 W0 进行比较（字节模式）
; 指令执行后，递增 W1

| 指令执行前 | | | | 指令执行后 | | | |
|-------|------|-----------|------|------------|------|-----------|------|
| W0 | ABA9 | 数据单元 2000 | D004 | W0 | ABA9 | 数据单元 2000 | D004 |
| W1 | 2000 | SR | 0000 | W1 | 2001 | SR | 0009 |
| | | | | (N, C = 1) | | | |

例 2: CP W5, W6 ; 将 W6 与 W5 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------------------|
| W5 | 2334 | W5 | 2334 |
| W6 | 8001 | W6 | 8001 |
| SR | 0000 | SR | 000C (N, OV = 1) |

CP0

比较 f 和 0x0 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} CP0{.B} f

操作数: f ∈ [0 ... 8191]

操作: (f) − 0x0

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0010 | 0B0f | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 计算 (f) − 0x0 并更新状态寄存器。不保存减法的结果。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
f 位用于选择文件寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: CP0.B RAM100 ; 将 RAM100 与 0x0 进行比较（字节模式）

| | | | |
|--------|------|--------|-----------------|
| 指令执行前 | | 指令执行后 | |
| RAM100 | 44C3 | RAM100 | 44C3 |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: CP0 0x1FFE ; 将 (0x1FFE) 与 0x0 进行比较（字模式）

| | | | |
|-----------|------|-----------|--------------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 1FFE | 0001 | 数据单元 1FFE | 0001 |
| SR | 0000 | SR | 0001 (C = 1) |

CP0

比较 **Ws** 和 **0x0** 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } CP0{.B} Ws
 [Ws]
 [Ws++]
 [Ws--]
 [++Ws]
 [--Ws]

操作数: Ws ∈ [W0 ... W15]

操作: (Ws) – 0x0000

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0000 | 0000 | 0B00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 计算 (Ws) – 0x0000 并更新状态寄存器。不保存减法的结果。可使用寄存器直接或间接寻址对 **Ws** 进行寻址。

 B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

 p 位用于选择源地址模式。

 s 位用于选择 **Ws** 源寄存器的地址。

注： 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的**注 3**。

例 1: CP0.B [W4--] ; 将 [W4] 与 0 进行比较（字节模式）
 ; 指令执行后，递减 W4

| | | | |
|-----------|-------|-----------|--------------|
| | 指令执行前 | | 指令执行后 |
| W4 | 1001 | W4 | 1000 |
| 数据单元 1000 | 0034 | 数据单元 1000 | 0034 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: CP0 [--W5] ; 将 [--W5] 与 0 进行比较（字模式）

| | | | |
|-----------|-------|-----------|-----------------|
| | 指令执行前 | | 指令执行后 |
| W5 | 2400 | W5 | 23FE |
| 数据单元 23FE | 9000 | 数据单元 23FE | 9000 |
| SR | 0000 | SR | 0009 (N, C = 1) |

CPB带借位比较 f 和 WREG 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} CPB{.B} f

操作数: $f \in [0 \dots 8191]$

操作: $(f) - (WREG) - (\overline{C})$

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0011 | 1B0f | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 计算 $(f) - (WREG) - (\overline{C})$ 并更新状态寄存器。该指令与 SUBB 指令具有相同的效果，但并不保存减法的结果。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。
- 3: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: CPB.B RAM400 ; 带借位位 \overline{C} 将 RAM400 与 WREG 进行比较（字节模式）

| 指令执行前 | | | 指令执行后 | | |
|--------|------|--|--------|------|---------|
| WREG | 8823 | | WREG | 8823 | |
| RAM400 | 0823 | | RAM400 | 0823 | |
| SR | 0000 | | SR | 0008 | (N = 1) |

例 2: CPB 0x1200 ; 带借位位 \overline{C} 将 (0x1200) 与 WREG 进行比较（字模式）

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|---------|-----------|------|---------|
| WREG | 2377 | | WREG | 2377 | |
| 数据单元 1200 | 2377 | | 数据单元 1200 | 2377 | |
| SR | 0001 | (C = 1) | SR | 0001 | (C = 1) |

CPB

带借位比较 Wb 和 lit5 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CPB{.B} Wb, #lit5

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]

操作: (Wb) − lit5 − (C̄)

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 1www | wB00 | 011k | kkkk |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) − lit5 − (C̄) 并更新状态寄存器。该指令与 SUBB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于提供一个 5 位整数的立即数操作数。

- 注 1:** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2:** Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1
指令周期数: 1

例 1: CPB.B W4, #0x12 ; 带借位位 C̄ 将 W4 与 0x12 进行比较（字节模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W4 | 7711 | W4 | 7711 |
| SR | 0001 (C = 1) | SR | 0008 (N = 1) |

例 2: CPB.B W4, #0x12 ; 带借位位 C̄ 将 W4 与 0x12 进行比较（字节模式）

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W4 | 7711 | W4 | 7711 |
| SR | 0000 | SR | 0008 (N = 1) |

例 3: CPB W12, #0x1F ; 带借位位 C̄ 将 W12 与 0x1F 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|-----------------|
| W12 | 0020 | W12 | 0020 |
| SR | 0002 (Z = 1) | SR | 0003 (Z, C = 1) |

例 4: CPB W12, #0x1F ; 带借位位 C̄ 将 W12 与 0x1F 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|-----------------|-------|--------------|
| W12 | 0020 | W12 | 0020 |
| SR | 0003 (Z, C = 1) | SR | 0001 (C = 1) |

CPB

带借位比较 Wb 和 lit8 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPB{.B} Wb, #lit8

操作数: Wb ∈ [W0 ... W15]
lit8 ∈ [0 ... 255]

操作: (Wb) − lit8 − (C̄)

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 1www | wBkk | k11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) − lit8 − (C̄) 并更新状态寄存器。该指令与 SUBB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于提供一个 5 位整数的立即数操作数。

- 注 1:** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2:** Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1

指令周期数: 1

例 1: CPB.B W4, #0x12 ; 带借位 C̄ 将 W4 与 0x12 进行比较（字节模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W4 | 7711 | W4 | 7711 |
| SR | 0001 (C = 1) | SR | 0008 (N = 1) |

例 2: CPB.B W4, #0x12 ; 带借位 C̄ 将 W4 与 0x12 进行比较（字节模式）

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W4 | 7711 | W4 | 7711 |
| SR | 0000 | SR | 0008 (N = 1) |

例 3: CPB W12, #0x1F ; 带借位 C̄ 将 W12 与 0x1F 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|-----------------|
| W12 | 0020 | W12 | 0020 |
| SR | 0002 (Z = 1) | SR | 0003 (Z, C = 1) |

例 4: CPB W12, #0x1F ; 带借位 C̄ 将 W12 与 0x1F 进行比较（字模式）

| 指令执行前 | | 指令执行后 | |
|-------|-----------------|-------|--------------|
| W12 | 0020 | W12 | 0020 |
| SR | 0003 (Z, C = 1) | SR | 0001 (C = 1) |

CPB

带借位比较 Ws 和 Wb 并设置状态标志位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} CPB{.B} Wb, Ws
[Ws]
[Ws++]
[Ws--]
[++Ws]
[--Ws]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
操作: (Wb) – (Ws) – (C)
受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0001 | 1www | wB00 | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 计算 (Wb) – (Ws) – (C) 并更新状态寄存器。该指令与 SUBB 指令具有相同的效果，但并不保存减法的结果。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 进行寻址。
w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
p 位用于选择源地址模式。
s 位用于选择 Ws 源寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
2: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: CPB.B W0, [W1++] ; 带借位位 C 将 [W1] 与 W0 进行比较（字节模式）
; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W0 | ABA9 | W0 | ABA9 |
| W1 | 1000 | W1 | 1001 |
| 数据单元 1000 | D0A9 | 数据单元 1000 | D0A9 |
| SR | 0002 (Z = 1) | SR | 0008 (N = 1) |

例 2: CPB.B W0, [W1++] ; 带借位位 \overline{C} 将 [W1] 与 W0 进行比较 (字节模式)
; 指令执行后, 递增 W1

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|--------------|--|-----------|--|--------------|--|
| W0 | | ABA9 | | W0 | | ABA9 | |
| W1 | | 1000 | | W1 | | 1001 | |
| 数据单元 1000 | | D0A9 | | 数据单元 1000 | | D0A9 | |
| SR | | 0001 (C = 1) | | SR | | 0001 (C = 1) | |

例 3: CPB W4, W5 ; 带借位位 \overline{C} 将 W5 与 W4 进行比较 (字模式)

| 指令执行前 | | | | 指令执行后 | | | |
|-------|--|--------------|--|-------|--|--------------|--|
| W4 | | 4000 | | W4 | | 4000 | |
| W5 | | 3000 | | W5 | | 3000 | |
| SR | | 0001 (C = 1) | | SR | | 0001 (C = 1) | |

CPBEQ

比较 Wb 和 Wn，如果相等 (Wb = Wn) 则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} CPBEQ{.B} Wb, Wn, Expr

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) = (Wn)，则 [(PC+2) + 2 * Expr] → PC 且 NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 1www | wBnn | nnnn | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) = (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），并根据 Expr 指定的 6 位有符号偏移量值重新计算 PC，而下一个指令周期则改为执行一条 NOP 指令。如果 (Wb) ≠ (Wn)，将顺序执行下一条指令（不执行转移操作）。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。
n 位用于选择转移目标地址的偏移量。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行转移操作，则为 5）

例 1: 002000 HERE: CPBEQ.B W0, W1, BYPASS ; 如果 W0 = W1（字节模式），
002002 ADD W2, W3, W4 ; 则执行转移到 BYPASS
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|--------------|
| PC | 00 2000 | PC | 00 2008 |
| W0 | 1000 | W0 | 1000 |
| W1 | 1000 | W1 | 1000 |
| SR | 0000 | SR | 0002 (Z = 1) |

CPBGT 有符号比较 Wb 和 Wn，如果大于 (Wb > Wn) 则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} CPBGT{.B} Wb, Wn, Expr

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) > (Wn)，则 [(PC+2) + 2 * Expr] → PC 且 NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 0www | wBnn | nnnn | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) > (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），并根据 Expr 指定的 6 位有符号偏移量值重新计算 PC，而下一个指令周期则改为执行一条 NOP 指令。如果 (Wb) ≤ (Wn)，将顺序执行下一条指令（不执行转移操作）。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字节操作，1 选择字操作）。
s 位用于选择 Wn 源寄存器的地址。
n 位用于选择转移目标地址的偏移量。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行转移操作，则为 5）

例 1: 002000 HERE: CPBGT.B W0, W1, BYPASS ; 如果 W0 > W1（字节模式），
002002 ADD W2, W3, W4 ; 则执行转移到 BYPASS
002004 . . .
002006 . . .
002008 BYPASS . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|-----------------|
| PC | 00 2000 | PC | 00 2008 |
| W0 | 30FF | W0 | 00FF |
| W1 | 26FE | W1 | 26FE |
| SR | 0000 | SR | 0000 (N, C = 0) |

CPBLT 有符号比较 Wb 和 Wn，如果小于 (Wb < Wn) 则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPBLT{.B} Wb, Wn, Expr

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) < (Wn)，则 [(PC+2) + 2 * Expr] → PC 且 NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 1www | wBnn | nnnn | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) < (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），并根据 Expr 指定的 6 位有符号偏移量值重新计算 PC，而下一个指令周期则改为执行一条 NOP 指令。如果 (Wb) ≥ (Wn)，将顺序执行下一条指令（不执行转移操作）。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。
n 位用于选择转移目标地址的偏移量。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行转移操作，则为 5）

例 1: 002000 HERE: CPBLT.B W8, W9, BYPASS ; 如果 W8 < W9（字节模式），
002002 ADD W2, W3, W4 ; 则执行转移到 BYPASS
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|--------------|
| PC | 00 2000 | PC | 00 2008 |
| W8 | 00FF | W8 | 00FF |
| W9 | 26FE | W9 | 26FE |
| SR | 0000 | SR | 0008 (N = 1) |

CPBNE 比较 Wb 和 Wn，如果不等于 (Wb ≠ Wn) 则转移

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} CPBNE{.B} Wb, Wn, Expr

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) ≠ (Wn), 则 [(PC+2) + 2 * Expr] → PC 且 NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 0www | wBnn | nnnn | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn), 将 Wb 和 Wn 的内容进行比较, 但并不保存计算结果。如果 (Wb) ≠ (Wn), 将放弃执行下一条指令 (在当前指令执行过程中取出), 并根据 Expr 指定的 6 位有符号偏移量值重新计算 PC, 而下一个指令周期则改为执行一条 NOP 指令。如果 (Wb) = (Wn), 将顺序执行下一条指令 (不执行转移操作)。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
s 位用于选择 Wn 源寄存器的地址。
n 位用于选择转移目标地址的偏移量。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。

指令字数: 1
指令周期数: 1 (如果执行转移操作, 则为 5)

例 1: 002000 HERE: CPBNE.B W2, W3, BYPASS ; 如果 W2 != W3 (字节模式),
002002 ADD W2, W3, W4 ; 则执行转移到 BYPASS
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| | | | |
|-------|---------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 2000 | PC | 00 200A |
| W2 | 00FF | W2 | 00FF |
| W3 | 26FE | W3 | 26FE |
| SR | 0000 | SR | 0001 (C = 1) |

CPSEQ

比较 Wb 和 Wn，如果相等 (Wb = Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CPSEQ{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) = (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 1www | wB00 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) = (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。如果 (Wb) ≠ (Wn)，将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSEQ.B W0, W1 ; 如果 W0 = W1（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002004 . . .
002006 . . .
002008 BYPASS: . . .
00200A . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 2000 | PC | 00 2002 |
| W0 | 1001 | W0 | 1001 |
| W1 | 1000 | W1 | 1000 |
| SR | 0000 | SR | 0000 |

例 2: 018000 HERE: CPSEQ W4, W8 ; 如果 W4 = W8（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 ...
018008 ...

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 01 8000 | PC | 01 8006 |
| W4 | 3344 | W4 | 3344 |
| W8 | 3344 | W8 | 3344 |
| SR | 0002 (Z = 1) | SR | 0002 (Z = 1) |

CPSEQ 比较 Wb 和 Wn，如果相等 (Wb = Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPSEQ{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) = (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 1www | wB00 | 0001 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) = (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。如果 (Wb) ≠ (Wn)，将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。

B 位用于选择字节或字操作（0 选择字节操作，1 选择字操作）。

s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1:

```
002000 HERE:  CPSEQ.B  W0, W1 ; 如果 W0 = W1（字节模式），
002002          GOTO    BYPASS ; 则跳过 GOTO
002004          . . .
002006          . . .
002008 BYPASS: . . .
00200A          . . .
```

| | | | |
|----|---------|----|---------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 2000 | PC | 00 2002 |
| W0 | 1001 | W0 | 1001 |
| W1 | 1000 | W1 | 1000 |
| SR | 0000 | SR | 0000 |

例 2:

018000 HERE: CPSEQ W4, W8 ; 如果 W4 = W8 (字模式),
018002 CALL _FIR ; 则跳过子程序调用
018006 ...
018008 ...

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 01 8000 | PC | 01 8006 |
| W4 | 3344 | W4 | 3344 |
| W8 | 3344 | W8 | 3344 |
| SR | 0002 (Z = 1) | SR | 0002 (Z = 1) |

CPSGT 有符号比较 Wb 和 Wn，如果大于 (Wb > Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} CPSGT{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) > (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 0www | wB00 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) > (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSGT.B W0, W1 ; 如果 W0 > W1（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002006 . . .
002008 . . .
00200A BYPASS . . .
00200C . . .

| 指令执行前 | | 指令执行后 | |
|-------|-----------------|-------|-----------------|
| PC | 00 2000 | PC | 00 2006 |
| W0 | 00FF | W0 | 00FF |
| W1 | 26FE | W1 | 26FE |
| SR | 0009 (N, C = 1) | SR | 0009 (N, C = 1) |

例 2: 018000 HERE: CPSGT W4, W5 ; 如果 W4 > W5（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 ...
018008 ...

| 指令执行前 | | 指令执行后 | |
|-------|---------------|-------|---------------|
| PC | 01 8000 | PC | 01 8002 |
| W4 | 2600 | W4 | 2600 |
| W5 | 2600 | W5 | 2600 |
| SR | 0004 (OV = 1) | SR | 0004 (OV = 1) |

CPSGT 有符号比较 Wb 和 Wn，如果大于 (Wb > Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPSGT{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) > (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 0www | wB00 | 0001 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) > (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSGT.B W0, W1 ; 如果 W0 > W1（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002006 . . .
002008 . . .
00200A BYPASS . . .
00200C . . .

| 指令执行前 | | 指令执行后 | |
|-------|-----------------|-------|-----------------|
| PC | 00 2000 | PC | 00 2006 |
| W0 | 00FF | W0 | 00FF |
| W1 | 26FE | W1 | 26FE |
| SR | 0009 (N, C = 1) | SR | 0009 (N, C = 1) |

例 2: 018000 HERE: CPSGT W4, W5 ; 如果 W4 > W5（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 ...
018008 ...

| 指令执行前 | | 指令执行后 | |
|-------|---------------|-------|---------------|
| PC | 01 8000 | PC | 01 8002 |
| W4 | 2600 | W4 | 2600 |
| W5 | 2600 | W5 | 2600 |
| SR | 0004 (OV = 1) | SR | 0004 (OV = 1) |

CPSLT 有符号比较 Wb 和 Wn，如果小于 (Wb < Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CPSLT{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) < (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 1www | wB00 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) < (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSLT.B W8, W9 ; 如果 W8 < W9（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002006 . . .
002008 . . .
00200A BYPASS: . . .
00200C . . .

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 00 2000 | PC | 00 2002 |
| W8 | 00FF | W8 | 00FF |
| W9 | 26FE | W9 | 26FE |
| SR | 0008 (N = 1) | SR | 0008 (N = 1) |

例 2: 018000 HERE: CPSLT W3, W6 ; 如果 W3 < W6（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 . . .
018008 . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 01 8000 | PC | 01 8006 |
| W3 | 2600 | W3 | 2600 |
| W6 | 3000 | W6 | 3000 |
| SR | 0000 | SR | 0000 |

CPSLT

有符号比较 Wb 和 Wn，如果小于 (Wb < Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPSLT{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) < (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0110 | 1www | wB00 | 0001 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) < (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSLT.B W8, W9 ; 如果 W8 < W9（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002006 . . .
002008 . . .
00200A BYPASS: . . .
00200C . . .

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 00 2000 | PC | 00 2002 |
| W8 | 00FF | W8 | 00FF |
| W9 | 26FE | W9 | 26FE |
| SR | 0008 (N = 1) | SR | 0008 (N = 1) |

例 2: 018000 HERE: CPSLT W3, W6 ; 如果 W3 < W6（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 . . .
018008 . . .

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 01 8000 | PC | 01 8006 |
| W3 | 2600 | W3 | 2600 |
| W6 | 3000 | W6 | 3000 |
| SR | 0000 | SR | 0000 |

CPSNE 比较 Wb 和 Wn，如果不等于 (Wb ≠ Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } CPSNE{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) ≠ (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 0www | wB00 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) ≠ (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1:

```
002000 HERE:  CPSNE.W W2, W3 ; 如果 W2 != W3（字节模式），
002002          GOTO  BYPASS ; 则跳过 GOTO
002006          . . .
002008          . . .
00200A BYPASS: . . .
00200C          . . .
```

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 00 2000 | PC | 00 2006 |
| W2 | 00FF | W2 | 00FF |
| W3 | 26FE | W3 | 26FE |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

例 2:

```
018000 HERE:  CPSNE.W W0, W8 ; 如果 W0 != W8（字模式），
018002          CALL  _FIR  ; 则跳过子程序调用
018006          ...
018008          ...
```

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 01 8000 | PC | 01 8002 |
| W0 | 3000 | W0 | 3000 |
| W8 | 3000 | W8 | 3000 |
| SR | 0000 | SR | 0000 |

CPSNE 比较 Wb 和 Wn，如果不等于 (Wb ≠ Wn) 则跳过

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } CPSNE{.B} Wb, Wn

操作数: Wb ∈ [W0 ... W15]
Wn ∈ [W0 ... W15]

操作: (Wb) – (Wn)
如果 (Wb) ≠ (Wn)，则跳过

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 0111 | 0www | wB00 | 0001 | ssss |
|-------|------|------|------|------|------|------|

描述: 通过执行减法 (Wb) – (Wn)，将 Wb 和 Wn 的内容进行比较，但并不保存计算结果。如果 (Wb) ≠ (Wn)，将放弃执行下一条指令（在当前指令执行过程中取出），而在下一个指令周期内执行一条 NOP 指令。否则将顺序执行下一条指令。

w 位用于选择 Wb 源寄存器的地址。
B 位用于选择字节或字操作（0 选择字节操作，1 选择字节操作）。
s 位用于选择 Wn 源寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1（如果执行跳过操作，则为 2 或 3）

例 1: 002000 HERE: CPSNE.B W2, W3 ; 如果 W2 != W3（字节模式），
002002 GOTO BYPASS ; 则跳过 GOTO
002006 . . .
002008 . . .
00200A BYPASS: . . .
00200C . . .

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| PC | 00 2000 | PC | 00 2006 |
| W2 | 00FF | W2 | 00FF |
| W3 | 26FE | W3 | 26FE |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

例 2: 018000 HERE: CPSNE W0, W8 ; 如果 W0 != W8（字模式），
018002 CALL _FIR ; 则跳过子程序调用
018006 ...
018008 ...

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 01 8000 | PC | 01 8002 |
| W0 | 3000 | W0 | 3000 |
| W8 | 3000 | W8 | 3000 |
| SR | 0000 | SR | 0000 |

DAW.B 对 Wn 的内容进行十进制调整

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DAW.B Wn

操作数: Wn ∈ [W0 ... W15]

操作: 如果 (Wn<3:0> > 9) 或 (DC = 1)
(Wn<3:0>) + 6 → Wn<3:0>
否则
(Wn<3:0>) → Wn<3:0>

如果 (Wn<7:4> > 9) 或 (C = 1)
(Wn<7:4>) + 6 → Wn<7:4>
否则
(Wn<7:4>) → Wn<7:4>

受影响的状态位: C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1101 | 0100 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 对 Wn 中的低位字节进行调整以产生一个二进制编码的十进制 (BCD) 结果。Wn 中的高位字节将不会改变, 且进位标志将被用来指示十进制溢出。必须使用寄存器直接寻址对 Wn 进行寻址。

s 位用于选择源 / 目标寄存器。

- 注 1: 该指令用来对两个组合 BCD 字节相加之后的数据格式进行调整。
2: 该指令只能工作于字节模式且操作码必须包括 .B 扩展符。

指令字数: 1

指令周期数: 1

例 1: DAW.B W0 ; 十进制调整 w0

| | | | |
|-------|---------------|-------|---------------|
| 指令执行前 | | 指令执行后 | |
| W0 | 771A | W0 | 7720 |
| SR | 0002 (DC = 1) | SR | 0002 (DC = 1) |

例 2: DAW.B W3 ; 十进制调整 w3

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W3 | 77AA | W3 | 7710 |
| SR | 0000 | SR | 0001 (C = 1) |

DEC

f 内容递减 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } DEC{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $(f) - 1 \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1101 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容减 1，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: DEC.B 0x200 ; 将 (0x200) 递减 1（字节模式）

| | | | |
|----------|------|----------|-----------------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 200 | 80FF | 数据单元 200 | 80FE |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: DEC RAM400, WREG ; 将 RAM400 中内容递减 1，并将结果存入 WREG
;（字模式）

| | | | |
|--------|------|--------|------|
| 指令执行前 | | 指令执行后 | |
| WREG | 1211 | WREG | 0822 |
| RAM400 | 0823 | RAM400 | 0823 |
| SR | 0000 | SR | 0000 |

DEC

Ws 内容递减 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DEC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) – 1 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1001 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容减 1，并将结果存放在目标寄存器 Wd 中。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: DEC.B [W7++], [W8++] ; 将 [W7] 中内容递减 1，并将结果存入 [W8]（字节模式）
; 指令执行后，递增 W7 和 W8

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W7 | 2301 | W7 | 2302 |
| W8 | 2400 | W8 | 2401 |
| 数据单元 2300 | 5607 | 数据单元 2300 | 5607 |
| 数据单元 2400 | ABCD | 数据单元 2400 | AB55 |
| SR | 0000 | SR | 0000 |

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|-----------------|
| W5 | D004 | W5 | D004 |
| W6 | 2000 | W6 | 2002 |
| 数据单元 2000 | ABA9 | 数据单元 2000 | D003 |
| SR | 0000 | SR | 0009 (N, C = 1) |

DEC2f 内容递减 2

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DEC2{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (f) − 2 → 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1101 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容减 2，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: DEC2.B 0x200 ; 将 (0x200) 递减 2（字节模式）

| 指令执行前 | | 指令执行后 | |
|----------|------|----------|-----------------|
| 数据单元 200 | 80FF | 数据单元 200 | 80FD |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: DEC2 RAM400, WREG ; 将 RAM400 中内容递减 2，
; 并将结果存入 WREG（字模式）

| 指令执行前 | | 指令执行后 | |
|--------|------|--------|------|
| WREG | 1211 | WREG | 0821 |
| RAM400 | 0823 | RAM400 | 0823 |
| SR | 0000 | SR | 0000 |

DEC2

Ws 内容递减 2

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DEC2{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) − 2 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1001 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容减 2，并将结果存放在目标寄存器 Wd 中。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: DEC2.B [W7--], [W8--] ; 将 [W7] 中内容递减 2，并将结果存入 [W8]（字节模式）
; 指令执行后，递增 W7 和 W8

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| W7 | 2301 | W7 | 2300 |
| W8 | 2400 | W8 | 23FF |
| 数据单元 2300 | 0107 | 数据单元 2300 | 0107 |
| 数据单元 2400 | ABCD | 数据单元 2400 | ABFF |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: DEC2 W5, [W6++] ; 将 W5 中内容递减 2，并将结果存入 [W6]（字模式）
 ; 指令执行后，递增 W6

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|-----------|-------|--|------|------------|
| W5 | | D004 | | W5 | | D004 | |
| W6 | | 1000 | | W6 | | 1002 | |
| 数据单元 1000 | | ABA9 | 数据单元 1000 | | | D002 | |
| SR | | 0000 | SR | | | 0009 | (N, C = 1) |

DISI

暂时禁止中断

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DISI #lit14

操作数: lit14 ∈ [0 ... 16383]

操作: lit14 → DISICNT
1 → DISI
禁止中断，持续时间为 (lit14 + 1) 个指令周期

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1100 | 00kk | kkkk | kkkk | kkkk |
|-------|------|------|------|------|------|------|

描述: 禁止优先级 0 至 6 的中断，禁止时间为 (lit14 + 1) 个指令周期。禁止持续时间从执行 DISI 的指令周期开始，且在以后的 (lit 14) 个指令周期内保持有效。lit14 值将被写入 DISICNT 寄存器，且 DISI 标志 (INTCON2<14>) 将被设定为 1。该指令可用于对执行时间要求苛刻的代码之前以限制中断的影响。

- 注 1: 该指令并不禁止优先级为 7 的中断以及陷阱的处理。有关详细信息，请参见具体器件的系列参考手册。
- 2: 当器件处于休眠模式时，该指令不阻止任何中断。

指令字数: 1

指令周期数: 1

例 1: 002000 HERE: DISI #100 ; 禁止中断，持续时间为 101 个指令周期
002002 ; 下 100 个周期由 DISI 保护
002004 . . .

| 指令执行前 | | 指令执行后 | |
|---------|---------|---------|-----------------|
| PC | 00 2000 | PC | 00 2002 |
| DISICNT | 0000 | DISICNT | 0100 |
| INTCON2 | 0000 | INTCON2 | 4000 (DISI = 1) |
| SR | 0000 | SR | 0000 |

DIV.S

有符号整数除法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DIV.S{W} Wm, Wn
DIV.SD Wm, Wn

操作数: 对于字操作, Wm ∈ [W0 ... W15]
对于双字操作, Wm ∈ [W0, W2, W4 ... W14]
Wn ∈ [W2 ... W15]

操作: 对于字操作 (默认):
Wm → W0
如果 (Wm<15> = 1):
0xFFFF → W1
否则:
0x0 → W1
W1:W0 / Wn → W0
余数 → W1

对于双字操作 (DIV.SD):
Wm + 1:Wm → W1:W0
W1:W0 / Wn → W0
余数 → W1

受影响的状态位: N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 1000 | 0ttt | tvvv | vW00 | ssss |
|-------|------|------|------|------|------|------|

描述: 迭代的有符号整数除法, 其中被除数存储在 Wm (对于 16 位 /16 位除法) 或 Wm + 1:Wm (对于 32 位 /16 位除法) 中, 而除数存储在 Wn 中。在默认的字操作中, Wm 将首先被拷贝至 W0 且通过 W1 进行符号扩展以执行操作。在双字操作中, Wm + 1:Wm 将首先被拷贝至 W1:W0。除法操作的 16 位商将存放在 W0 中, 而 16 位余数则被存放在 W1 中。必须通过 REPEAT 指令 (迭代计数值为 17) 执行该指令 18 次以得到正确的商和余数。如果余数为负, 则 N 标志将被置 1, 否则将被清零。如果除法操作导致溢出, 则 OV 标志将被置 1, 否则将被清零。如果余数为 0, 则 Z 标志将被置 1, 否则将被清零。C 标志用来实现除法算法, 但不应使用其最终值。
t 位用于选择双字操作中被除数的高位字。对于字操作, 这些位将被清零。
v 位用于选择被除数的低位字。
W 位用于选择被除数的长度 (0 选择 16 位, 1 选择 32 位)。
s 位用于选择除数寄存器。

- 注 1: 指令中的扩展符 .D 指明为双字 (32 位) 被除数而非字被除数。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2: 如果商不能表示为 16 位，将出现意想不到的结果。当该情形发生在双字操作 (DIV.SD) 时，OV 状态位将被置 1，而此时的商和余数不应被使用。对于字操作 (DIV.S)，只有一种类型的溢出会发生 (0x8000/0xFFFF = + 32768 或 0x00008000)，这使得 OV 状态位可对结果进行解释。
- 3: 以零作为除数时，将在除法执行过程中的第一个指令周期内触发一个运算出错陷阱。
- 4: 该指令在每一个指令周期边界都是可中断的。

指令字数: 1
指令周期数: 18 (外加执行 REPEAT 的 1 个指令周期)

例 1: REPEAT #17 ; 执行 DIV.S 18 次
DIV.S W3, W4 ; 将 W3 除以 W4
; 将商存入 W0，余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 5555 | W0 | 013B |
| W1 | 1234 | W1 | 0003 |
| W3 | 3000 | W3 | 3000 |
| W4 | 0027 | W4 | 0027 |
| SR | 0000 | SR | 0000 |

例 2: REPEAT #17 ; 执行 DIV.SD 18 次
DIV.SD W0, W12 ; 将 W1:W0 除以 W12
; 将商存入 W0，余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 2500 | W0 | FA6B |
| W1 | FF42 | W1 | EF00 |
| W12 | 2200 | W12 | 2200 |
| SR | 0000 | SR | 0008 (N = 1) |

DIV.U 无符号整数除法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} DIV.U{W} Wm, Wn
DIV.UD Wm, Wn

操作数: 对于字操作, Wm ∈ [W0 ... W15]
对于双字操作, Wm ∈ [W0, W2, W4 ... W14]
Wn ∈ [W2 ... W15]

操作: 对于字操作 (默认):
Wm → W0
0x0 → W1
W1:W0/Wn → W0
余数 → W1

对于双字操作 (DIV.UD):
Wm + 1:Wm → W1:W0
W1:W0/Wns → W0
余数 → W1

受影响的状态位: N、OV、Z 和 C

| | | | | | | |
|-------|------|------|-------|------|------|------|
| 指令编码: | 1101 | 1000 | 1tttt | tvvv | vW00 | ssss |
|-------|------|------|-------|------|------|------|

描述: 迭代的无符号整数除法, 其中被除数存储在 Wm (对于 16 位 /16 位除法) 或 Wm + 1:Wm (对于 32 位 /16 位除法) 中, 而除数存储在 Wn 中。在字操作中, Wm 将首先被拷贝至 W0, 而 W1 将被清零以执行除法操作。在双字操作中, Wm + 1:Wm 将首先被拷贝至 W1:W0。除法操作的 16 位商将存放在 W0 中, 而 16 位余数则被存放在 W1 中。
必须通过 REPEAT 指令 (迭代计数值为 17) 执行该指令 18 次以得到正确的商和余数。N 标志总是被清零。如果除法操作导致溢出, 则 OV 标志将被置 1, 否则将被清零。如果余数为 0, 则 Z 标志将被置 1, 否则将被清零。C 标志用来实现除法算法, 但不应使用其最终值。
t 位用于选择双字操作中被除数的高位字。对于字操作, 这些位将被清零。
v 位用于选择被除数的低位字。
W 位用于选择被除数的长度 (0 选择 16 位, 1 选择 32 位)。
s 位用于选择除数寄存器。

- 注 1: 指令中的扩展符 .D 指明为双字 (32 位) 被除数而非字被除数。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 如果商不能表示为 16 位, 将出现意想不到的结果。仅双字操作 (DIV.UD) 会出现这种情况。当溢出发生时, OV 状态位将被置 1, 而此时的商和余数则不应被使用。
- 3: 以零作为除数时, 将在除法执行过程中的第一个指令周期内触发一个运算出错陷阱。
- 4: 该指令在每一个指令周期边界都是可中断的。

指令字数: 1
指令周期数: 18 (外加执行 REPEAT 的 1 个指令周期)

例 1: REPEAT #17 ; 执行 DIV.U 18 次
 DIV.U W2, W4 ; 将 W2 除以 W4
 ; 将商存入 W0，余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 5555 | W0 | 0040 |
| W1 | 1234 | W1 | 0000 |
| W2 | 8000 | W2 | 8000 |
| W4 | 0200 | W4 | 0200 |
| SR | 0000 | SR | 0002 (Z = 1) |

例 2: REPEAT #17 ; 执行 DIV.UD 18 次
 DIV.UD W10, W12 ; 将 W11:W10 除以 W12
 ; 将商存入 W0，余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 5555 | W0 | 01F2 |
| W1 | 1234 | W1 | 0100 |
| W10 | 2500 | W10 | 2500 |
| W11 | 0042 | W11 | 0042 |
| W12 | 2200 | W12 | 2200 |
| SR | 0000 | SR | 0000 |

DIVF

小数除法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} DIVF Wm, Wn

操作数: Wm ∈ [W0 ... W15]
Wn ∈ [W2 ... W15]

操作: 0x0 → W0
Wm → W1
W1:W0/Wn → W0
余数 → W1

受影响的状态位: N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 1001 | 0ttt | t000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 迭代的有符号 16 位 /16 位小数除法，其中被除数存储在 Wm 中而除数存储在 Wn 中。在操作过程中，W0 将首先被清零而 Wm 将被拷贝至 W1。除法操作的 16 位商将存入 W0，而 16 位余数则被存入 W1。余数的符号将与被除数的符号相同。

必须通过 REPEAT 指令（迭代计数值为 17）执行该指令 18 次以得到正确的商和余数。如果余数为负，则 N 标志将被置 1，否则将被清零。如果除法操作导致溢出，则 OV 标志将被置 1，否则将被清零。如果余数为 0，则 Z 标志将被置 1，否则将被清零。C 标志用来实现除法算法，但不应使用其最终值。

t 位用于选择被除数寄存器。

s 位用于选择除数寄存器。

- 注 1: 为使小数除法有效，Wm 必须小于 Wn。如果 Wm 大于或等于 Wn，将出现意想不到的结果，这是因为小数结果将大于或等于 1.0。当发生这种情况时，OV 状态位将被置 1，而此时的商和余数则不应被使用。
- 2: 以零作为除数时，将在除法执行过程中的第一个指令周期内触发一个运算出错陷阱。
- 3: 该指令在每一个指令周期边界都是可中断的。

指令字数: 1

指令周期数: 18（外加执行 REPEAT 的 1 个指令周期）

例 1:

REPEAT #17
DIVF W8, W9

; 执行 DIVF 18 次
; 将 W8 除以 W9
; 将商存入 W0, 余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 8000 | W0 | 2000 |
| W1 | 1234 | W1 | 0000 |
| W8 | 1000 | W8 | 1000 |
| W9 | 4000 | W9 | 4000 |
| SR | 0000 | SR | 0002 (Z = 1) |

例 2:

REPEAT #17
DIVF W8, W9

; 执行 DIVF 18 次
; 将 W8 除以 W9
; 将商存入 W0, 余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 8000 | W0 | F000 |
| W1 | 1234 | W1 | 0000 |
| W8 | 1000 | W8 | 1000 |
| W9 | 8000 | W9 | 8000 |
| SR | 0000 | SR | 0002 (Z = 1) |

例 3:

REPEAT #17
DIVF W0, W1

; 执行 DIVF 18 次
; 将 W0 除以 W1
; 将商存入 W0, 余数存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 8002 | W0 | 7FFE |
| W1 | 8001 | W1 | 8002 |
| SR | 0000 | SR | 0008 (N = 1) |

DO

初始化硬件循环立即数

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | |

语法: { 标号 :} DO #lit14, Expr

操作数: lit14 ∈ [0 ... 16383]
Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 将与 DO 循环有关的寄存器（DCOUNT、DOEND 和 DOSTART）中内容压入其各自的影子寄存器
(lit14) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
递增 DL<2:0>（CORCON<10:8>）

受影响的状态位: DA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1000 | 00kk | kkkk | kkkk | kkkk |
| | 0000 | 0000 | nnnn | nnnn | nnnn | nnnn |

描述：启动一个无开销的硬件 DO 循环，该循环将执行 (lit14 + 1) 次。DO 循环从紧跟 DO 指令的地址处开始，并且在距离 DO 指令 2 * Slit16 个指令字的地址处结束。14 位计数值 (lit14) 可支持最大循环计数值 16384，而 16 位偏移量值 (Slit16) 可支持向前、向后 32K 个指令字的偏移量。

当执行该指令时，DCOUNT、DOSTART 和 DOEND 中内容首先将被压入其各自的影子寄存器，随后使用指令指定的新的 DO 循环参数对其进行更新。然后，DO 层级计数值 DL<2:0> (CORCON<8:10>) 将递增。在 DO 循环结束执行之后，将恢复压入影子寄存器的 DCOUNT、DOSTART 和 DOEND 寄存器的内容，而 DL<2:0> 将递减。

k 位用于指定循环计数值。

n 位构成一个有符号立即数，用于指定循环中最后一条指令距离 PC 的偏移量 (指令字数)。

- 特征和限制：**
- 以下特征和限制适用于 DO 指令。
- 使用循环计数值 0 将会导致循环仅执行一次。
 - 使用 -2、-1 或 0 作为循环长度是无效的。如果使用上述数值作为偏移量，可能会发生意想不到的结果。
 - DO 循环的**最后两条**指令不能是：
 - 改变程序控制流的指令
 - DO 或 REPEAT 指令如果使用上述任何指令，可能会发生意想不到的结果。
 - 如果在 DO 循环的第二条至最后一条指令或第三条至最后一条指令的过程中发生硬陷阱，则循环将无法正确工作。硬陷阱包括优先级为 13 至 15 的异常 (含优先级 13 和优先级 15)。
- 注 1：** DO 指令是可中断的，且支持 1 层硬件嵌套。用户可在程序中最多实现另外 5 层嵌套。有关详细信息，请参见具体器件的系列参考手册。
- 2：** 链接器将把指定的表达式转换为要使用的偏移量。

指令字数： 2

指令周期数： 2

例 1：

```
002000 LOOP6: DO    #5, END6    ; 启动 DO 循环 (执行循环 6 次)
002004          ADD    W1, W2, W3 ; 循环中第一条指令
002006          . . .
002008          . . .
00200A END6:   SUB    W2, W3, W4 ; 循环中最后一条指令
00200C          . . .
```

| 指令执行前 | | | 指令执行后 | | |
|---------|---------|---------|---------|---------|-------------|
| PC | 00 2000 | | PC | 00 2004 | |
| DCOUNT | 0000 | | DCOUNT | 0005 | |
| DOSTART | FF FFFF | | DOSTART | 00 2004 | |
| DOEND | FF FFFF | | DOEND | 00 200A | |
| CORCON | 0000 | | CORCON | 0100 | (DL = 1) |
| SR | 0001 | (C = 1) | SR | 0201 | (DA, C = 1) |

例 2:

01C000 LOOP12: DO

#0x160, END12 ; 启动 DO 循环（执行循环 353 次）

01C004 DEC

W1, W2 ; 循环中第一条指令

01C006 . . .

01C008 . . .

01C00A . . .

01C00C . . .

01C00E CALL

_FIR88 ; 调用 FIR88 子程序

01C012 NOP

01C014 END12: NOP

; 循环中最后一条指令

; （所需的 NOP 填充）

| 指令执行前 | | 指令执行后 | |
|---------|--------------|---------|------------------|
| PC | 01 C000 | PC | 01 C004 |
| DCOUNT | 0000 | DCOUNT | 0160 |
| DOSTART | FF FFFF | DOSTART | 01 C004 |
| DOEND | FF FFFF | DOEND | 01 C014 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0008 (N = 1) | SR | 0208 (DA, N = 1) |

DO

初始化硬件循环立即数

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 : } DO #lit15, Expr

操作数: lit15 ∈ [0 ... 32767]
Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16, 其中 Slit16 ∈ [-32768 ... +32767]。

操作: 将与 DO 循环有关的寄存器 (DCOUNT、DOEND 和 DOSTART) 中内容压入其各自的影子寄存器
(lit15) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
递增 DL<2:0> (CORCON<10:8>)

受影响的状态位: DA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1000 | 0kkk | kkkk | kkkk | kkkk |
| | 0000 | 0000 | nnnn | nnnn | nnnn | nnnn |

描述: 启动一个无开销的硬件 DO 循环, 该循环将执行 (lit15 + 1) 次。DO 循环从紧跟 DO 指令的地址处开始, 并且在距离 DO 指令 2 * Slit16 个指令字的地址处结束。15 位计数值 (lit15) 可支持最大循环计数值 32768, 而 16 位偏移量值 (Slit16) 可支持向前、向后 32K 个指令字的偏移量。

当执行该指令时, DCOUNT、DOSTART 和 DOEND 中内容首先将被压入其各自的影子寄存器, 随后使用指令指定的新的 DO 循环参数对其进行更新。然后, DO 层级计数值 DL<2:0> 位 (CORCON<8:10>) 将递增。在 DO 循环结束执行之后, 将恢复压入影子寄存器的 DCOUNT、DOSTART 和 DOEND 寄存器的内容, 而 DL<2:0> 将递减。

k 位用于指定循环计数值。
n 位构成一个有符号立即数, 用于指定循环中最后一条指令距离 PC 的偏移量 (指令字数)。

特征和限制:

- 以下特征和限制适用于 DO 指令。
- 使用循环计数值 0 将会导致循环仅执行一次。
 - 使用 -2、-1 或 0 作为循环长度是无效的。如果使用上述数值作为偏移量, 可能会发生意想不到的结果。
 - DO 循环的**最后两条**指令不能是:
 - 改变程序控制流的指令
 - DO 或 REPEAT 指令如果使用上述任何指令, 可能会发生意想不到的结果。
 - 如果在 DO 循环的第二条至最后一条指令或第三条至最后一条指令的过程中发生硬陷阱, 则循环将无法正确工作。硬陷阱包括优先级为 13 至 15 的异常 (含优先级 13 和优先级 15)。
 - DO 循环的第一条指令不能是 PSV 读或表读指令。
- 注 1:** DO 指令是可中断的, 且支持 1 层硬件嵌套。用户可在程序中最多实现另外 5 层嵌套。有关详细信息, 请参见具体器件的系列参考手册。
- 2:** 链接器将把指定的表达式转换为要使用的偏移量。

指令周期数: 2

| 指令执行前 | | 指令执行后 | |
|---------|--------------|---------|------------------|
| PC | 00 2000 | PC | 00 2004 |
| DCOUNT | 0000 | DCOUNT | 0005 |
| DOSTART | FF FFFF | DOSTART | 00 2004 |
| DOEND | FF FFFF | DOEND | 00 200A |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0001 (C = 1) | SR | 0201 (DA, C = 1) |

| 指令执行前 | | 指令执行后 | |
|---------|--------------|---------|------------------|
| PC | 01 C000 | PC | 01 C004 |
| DCOUNT | 0000 | DCOUNT | 0160 |
| DOSTART | FF FFFF | DOSTART | 01 C004 |
| DOEND | FF FFFF | DOEND | 01 C014 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0008 (N = 1) | SR | 0208 (DA, N = 1) |

DO

初始化硬件循环 Wn

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | |

语法: { 标号 : } DO Wn, Expr

操作数: Wn ∈ [W0 ... W15]
Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 将与 DO 循环有关的寄存器（DCOUNT、DOEND 和 DOSTART）中内容压入其各自的影子寄存器
(Wn<13:0>) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
递增 DL<2:0>（CORCON<10:8>）

受影响的状态位: DA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1000 | 1000 | 0000 | 0000 | ssss |
| | 0000 | 0000 | nnnn | nnnn | nnnn | nnnn |

描述: 启动一个无开销的硬件 DO 循环，该循环将执行 (Wn + 1) 次。DO 循环从紧跟 DO 指令的地址处开始，并且在距离 DO 指令 2 * Slit16 个指令字的地址处结束。Wn 的低 14 位可支持最大循环计数值 16384，而 16 位偏移量值 (Slit16) 可支持向前、向后 32K 个指令字的偏移量。

当执行该指令时，DCOUNT、DOSTART 和 DOEND 中内容首先将被压入其各自的影子寄存器，随后使用指令指定的新的 DO 循环参数对其进行更新。然后，DO 层级计数值 DL<2:0>（CORCON<8:10>）将递增。在 DO 循环结束执行之后，将恢复压入影子寄存器的 DCOUNT、DOSTART 和 DOEND 寄存器的内容，而 DL<2:0> 将递减。

s 位用于指定包含循环计数值的寄存器 Wn。
n 位构成一个有符号立即数，用于指定循环中最后一条指令距离 (PC + 4) 的偏移量（指令字数）。

- 特征和限制:**
以下特征和限制适用于 DO 指令。
- 使用循环计数值 0 将会导致循环仅执行一次。
 - 使用 -2、-1 或 0 作为偏移量是无效的。如果使用上述数值作为偏移量，可能会发生意想不到的结果。
 - DO 循环的**最后两条**指令不能是：
 - 改变程序控制流的指令
 - DO 或 REPEAT 指令如果使用上述任何指令，可能会发生意想不到的结果。

注 1: DO 指令是可中断的，且支持 1 层硬件嵌套。用户可在程序中最多实现另外 5 层嵌套。有关详细信息，请参见具体器件的系列参考手册。

2: 链接器将把指定的表达式转换为要使用的偏移量。

指令字数: 2
指令周期数: 2

例 1:

002000 LOOP6: DO W0, END6 ; 启动 DO 循环（循环计数为 W0 中内容）
002004 ADD W1, W2, W3 ; 循环中第一条指令
002006 . . .
002008 . . .
00200A . . .
00200C REPEAT #6
00200E SUB W2, W3, W4
002010 END6: NOP ; 循环中最后一条指令
; （所需的 NOP 填充）

| 指令执行前 | | 指令执行后 | |
|---------|---------|---------|---------------|
| PC | 00 2000 | PC | 00 2004 |
| W0 | 0012 | W0 | 0012 |
| DCOUNT | 0000 | DCOUNT | 0012 |
| DOSTART | FF FFFF | DOSTART | 00 2004 |
| DOEND | FF FFFF | DOEND | 00 2010 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0000 | SR | 0080 (DA = 1) |

例 2:

002000 LOOPA: DO W7, ENDA ; 启动 DO 循环（循环计数为 W7 中内容）
002004 SWAP W0 ; 循环中第一条指令
002006 . . .
002008 . . .
00200A . . .
002010 ENDA: MOV W1, [W2++] ; 循环中最后一条指令

| 指令执行前 | | 指令执行后 | |
|---------|---------|---------|---------------|
| PC | 00 2000 | PC | 00 2004 |
| W7 | E00F | W7 | E00F |
| DCOUNT | 0000 | DCOUNT | 200F |
| DOSTART | FF FFFF | DOSTART | 00 2004 |
| DOEND | FF FFFF | DOEND | 00 2010 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0000 | SR | 0080 (DA = 1) |

DO

初始化硬件循环 Wn

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 : } DO Wn, Expr

操作数: Wn ∈ [W0 ... W15]
Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... +32767]。

操作: 将与 DO 循环有关的寄存器（DCOUNT、DOEND 和 DOSTART）中内容压入其各自的影子寄存器
(Wn) → DCOUNT
(PC) + 4 → PC
(PC) → DOSTART
(PC) + (2 * Slit16) → DOEND
递增 DL<2:0>（CORCON<10:8>）

受影响的状态位: DA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1000 | 1000 | 0000 | 0000 | ssss |
| | 0000 | 0000 | nnnn | nnnn | nnnn | nnnn |

描述: 启动一个无开销的硬件 DO 循环，该循环将执行 (Wn + 1) 次。DO 循环从紧跟 DO 指令的地址处开始，并且在距离 DO 指令 2 * Slit16 个指令字的地址处结束。Wn 的 16 位可支持最大循环计数值 65536，而 16 位偏移量值（Slit16）可支持向前、向后 32K 个指令字的偏移量。

当执行该指令时，DCOUNT、DOSTART 和 DOEND 中内容首先将被压入其各自的影子寄存器，随后使用指令指定的新的 DO 循环参数对其进行更新。然后，DO 层级计数值 DL<2:0>（CORCON<8:10>）将递增。在 DO 循环结束执行之后，将恢复压入影子寄存器的 DCOUNT、DOSTART 和 DOEND 寄存器的内容，而 DL<2:0> 将递减。

s 位用于指定包含循环计数值的寄存器 Wn。
n 位构成一个有符号立即数，用于指定循环中最后一条指令距离 (PC + 4) 的偏移量（指令字数）。

特征和限制:

以下特征和限制适用于 DO 指令。

- 使用循环计数值 0 将会导致循环仅执行一次。
 - 使用 -2、-1 或 0 作为偏移量是无效的。如果使用上述数值作为偏移量，可能会发生意想不到的结果。
 - DO 循环的最后两条指令不能是：
 - 改变程序控制流的指令
 - DO 或 REPEAT 指令如果使用上述任何指令，可能会发生意想不到的结果。
 - DO 循环的第一条指令不能是 PSV 读或表读指令。
- 注 1: DO 指令是可中断的，且支持 1 层硬件嵌套。用户可在程序中最多实现另外 5 层嵌套。有关详细信息，请参见具体器件的系列参考手册。
- 2: 链接器将把指定的表达式转换为要使用的偏移量。

指令字数: 2

DO 初始化硬件循环 Wn

指令周期数: 2

例 1:

```
002000 LOOP6: DO      W0, END6 ; 启动 DO 循环 (循环计数为 W0 中内容)
002004          ADD     W1, W2, W3 ; 循环中第一条指令
002006          . . .
002008          . . .
00200A          . . .
00200C          REPEAT #6
00200E          SUB     W2, W3, W4
002010 END6:    NOP           ; 循环中最后一条指令
                                ; (所需的 NOP 填充)
```

| 指令执行前 | | 指令执行后 | |
|---------|---------|---------|---------------|
| PC | 00 2000 | PC | 00 2004 |
| W0 | 0012 | W0 | 0012 |
| DCOUNT | 0000 | DCOUNT | 0012 |
| DOSTART | FF FFFF | DOSTART | 00 2004 |
| DOEND | FF FFFF | DOEND | 00 2010 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0000 | SR | 0080 (DA = 1) |

例 2:

```
002000 LOOPA: DO      W7, ENDA ; 启动 DO 循环 (循环计数为 W7 中内容)
002004          SWAP    W0      ; 循环中第一条指令
002006          . . .
002008          . . .
00200A          . . .
002010 ENDA:    MOV     W1, [W2++] ; 循环中最后一条指令
```

| 指令执行前 | | 指令执行后 | |
|---------|---------|---------|---------------|
| PC | 00 2000 | PC | 00 2004 |
| W7 | E00F | W7 | E00F |
| DCOUNT | 0000 | DCOUNT | 200F |
| DOSTART | FF FFFF | DOSTART | 00 2004 |
| DOEND | FF FFFF | DOEND | 00 2010 |
| CORCON | 0000 | CORCON | 0100 (DL = 1) |
| SR | 0000 | SR | 0080 (DA = 1) |

求取欧几里德距离（无累加）

DS70157F_CN 第 239 页

ED W5*W5, B, [W9]+=2, [W11+W12], W5 ; 将 W5 平方并将结果存入 ACCB
; [W9]-[W11+W12] 并将结果存入 W5
; 指令执行后, 递增 W9

| | |
|-----------|------|
| 数据单元 1200 | 6A7C |
| 数据单元 2508 | 2B3D |
| SR | 0000 |

EDAC

求取欧几里德距离

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } EDAC Wm * Wm, Acc, [Wx], [Wy], Wxd
[Wx] += kx, [Wy] += ky,
[Wx] -= kx, [Wy] -= ky,
[W9 + [W11 +
W12], W12],

操作数: Acc ∈ [A,B]
Wm * Wm ∈ [W4 * W4, W5 * W5, W6 * W6, W7 * W7]
Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6]
Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6]
Wxd ∈ [W4 ... W7]

操作: (Acc(A 或 B)) + (Wm) * (Wm) → Acc(A 或 B)
([Wx] - [Wy]) → Wxd
(Wx) + kx → Wx
(Wy) + ky → Wy

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 00mm | A1xx | 00ii | ijjj | jj10 |
|-------|------|------|------|------|------|------|

描述: 计算 Wm 的平方以及由 [Wx] 和 [Wy] 指定的预取值的差值。Wm * Wm 的计算结果将被符号扩展为 40 位且被加到指定的累加器中。[Wx] - [Wy] 的计算结果将被存入 Wxd，Wxd 可与 Wm 为同一个寄存器。

如第 4.14.1 节“MAC 预取”中所介绍，操作数 Wx、Wxd 和 Wyd 用于指定预取操作，上述操作数支持间接寻址和寄存器偏移量寻址。

m 位用于选择平方操作的操作数寄存器 Wm。

A 位用于选择存放结果的累加器。

x 位用于选择存放预取差值的 Wxd 目标寄存器。

i 位用于选择 Wx 预取操作。

j 位用于选择 Wy 预取操作。

指令字数: 1

指令周期数: 1

例 1: EDAC W4*W4, A, [W8] += 2, [W10] -= 2, W4 ; 将 W4 平方并
; 将结果加到 ACCA
; [W8] - [W10] 并将结果存入 W4
; 指令执行后，递增 W8
; 指令执行后，递减 W10

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | 009A | W4 | 0057 |
| W8 | 1100 | W8 | 1102 |
| W10 | 2300 | W10 | 22FE |
| ACCA | 00 3D0A 3D0A | ACCA | 00 3D0A 99AE |
| 数据单元 1100 | 007F | 数据单元 1100 | 007F |
| 数据单元 2300 | 0028 | 数据单元 2300 | 0028 |
| SR | 0000 | SR | 0000 |

例 2:

EDAC W5*W5, B, [w9]+=2, [W11+W12], W5

; 将 W5 平方并

; 将结果加到 ACCB

; [W9]-[W11+W12] 并将结果存入 W5

; 指令执行后, 递增 W9

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W5 | 43C2 | W5 | 3F3F |
| W9 | 1200 | W9 | 1202 |
| W11 | 2500 | W11 | 2500 |
| W12 | 0008 | W12 | 0008 |
| ACCB | 00 28E3 F14C | ACCB | 00 3AD3 1050 |
| 数据单元 1200 | 6A7C | 数据单元 1200 | 6A7C |
| 数据单元 2508 | 2B3D | 数据单元 2508 | 2B3D |
| SR | 0000 | SR | 0000 |

EXCH

交换 Wns 和 Wnd 的内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} EXCH Wns, Wnd

操作数: Wns ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作: (Wns) ↔ (Wnd)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1101 | 0000 | 0ddd | d000 | ssss |
|-------|------|------|------|------|------|------|

描述: 交换两个工作寄存器的字内容。必须使用寄存器直接寻址对 Wns 和 Wnd 进行寻址。

d 位用于选择第一个寄存器的地址。
f 位用于选择第二个寄存器的地址。

注: 该指令只能工作于字模式。

指令字数: 1

指令周期数: 1

例 1: EXCH W1, W9 ; 将 W1 和 W9 中的内容交换

| | | | |
|----|-------|----|-------|
| | 指令执行前 | | 指令执行后 |
| W1 | 55FF | W1 | A3A3 |
| W9 | A3A3 | W9 | 55FF |
| SR | 0000 | SR | 0000 |

例 2: EXCH W4, W5 ; 将 W4 和 W5 中的内容交换

| | | | |
|----|-------|----|-------|
| | 指令执行前 | | 指令执行后 |
| W4 | ABCD | W4 | 4321 |
| W5 | 4321 | W5 | ABCD |
| SR | 0000 | SR | 0000 |

FBCL

搜索自左起第一个位变化

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} FBCL Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]
操作: Max_Shift = 15
Sign = (Ws) & 0x8000
Temp = (Ws) << 1
Shift = 0
While ((Shift < Max_Shift) && ((Temp & 0x8000) == Sign))
Temp = Temp << 1
Shift = Shift + 1
-Shift → (Wnd)

受影响的状态位: C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 1111 | 0000 | 0ddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 按照 Ws 中从字操作数符号位后的最高位至最低位的顺序，搜索第一个出现的 1（对于正值）或 0（对于负值）。位编号结果经过符号扩展为 16 位且被存放在 Wnd 中。

紧邻符号位之后的最高位被指定为位编号 0 而最低位被指定为位编号 -14。这种位排序使得 SFTAC 指令可直接使用 Wd 对值进行按比例增大。如果未找到位变化，将返回一个 -15 的结果并将 C 标志置 1。当找到一个位变化时，C 标志将被清零。

d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注： 该指令只能工作于字模式。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: FBCL W1, W9 ; 搜索 W1 中自左起第一个位变化
 ; 并将结果存入 W9

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W1 | 55FF | W1 | 55FF |
| W9 | FFFF | W9 | 0000 |
| SR | 0000 | SR | 0000 |

例 2: FBCL W1, W9 ; 搜索 W1 中自左起第一个位变化
 ; 并将结果存入 W9

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W1 | FFFF | W1 | FFFF |
| W9 | BBBB | W9 | FFF1 |
| SR | 0000 | SR | 0001 (C = 1) |

例 3: FBCL [W1++], W9 ; 搜索 [W1] 中自左起第一个位变化
 ; 并将结果存入 W9
 ; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W1 | 2000 | W1 | 2002 |
| W9 | BBBB | W9 | FFF9 |
| 数据单元 2000 | FF0A | 数据单元 2000 | FF0A |
| SR | 0000 | SR | 0000 |

FF1L

搜索自左起第一个 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} FF1L Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作: Max_Shift = 17
Temp = (Ws)
Shift = 1
While ((Shift < Max_Shift) && !(Temp & 0x8000))
Temp = Temp << 1
Shift = Shift + 1
If (Shift == Max_Shift)
0 → (Wnd)
Else
Shift → (Wnd)

受影响的状态位: C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1111 | 1000 | 0ddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 按照 Ws 中从字操作数最高位至最低位的顺序，搜索第一个出现的 1。位编号结果经过零扩展为 16 位且被存放在 Wnd 中。

位编号顺序从最高位（指定为编号 1）开始至最低位（指定为编号 16）结束。结果为零表明未找到 1，且 C 标志将被置 1。如果找到 1，C 标志将被清零。

d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 该指令只能工作于字模式。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

FF1R 搜索自右起第一个 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} FF1R Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作: Max_Shift = 17
Temp = (Ws)
Shift = 1
While ((Shift < Max_Shift) && !(Temp & 0x1))
Temp = Temp >> 1
Shift = Shift + 1
If (Shift == Max_Shift)
0 → (Wnd)
Else
Shift → (Wnd)

受影响的状态位: C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1111 | 0000 | 0ddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 按照 Ws 中从字操作数最低位至最高位的顺序，搜索第一个出现的 1。位编号结果经过零扩展为 16 位且被存放在 Wnd 中。

位编号顺序从最低位（指定为编号 1）开始至最高位（指定为编号 16）结束。结果为零表明未找到 1，且 C 标志将被置 1。如果找到 1，C 标志将被清零。

d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 该指令只能工作于字模式。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

GOTO

无条件跳转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } GOTO Expr

操作数: Expr 可以是标号或表达式（但不能为立即数）。
Expr 由链接器解析为一个 lit23，其中 lit23 ∈ [0 ... 8388606]。

操作: lit23 → PC
NOP → 指令寄存器

受影响的状态位: 无

指令编码:

| | | | | | | |
|------|------|------|------|------|------|------|
| 第一个字 | 0000 | 0100 | nnnn | nnnn | nnnn | nnn0 |
| 第二个字 | 0000 | 0000 | 0000 | 0000 | 0nnn | nnnn |

描述: 无条件跳转至 4M 指令字程序存储空间中的任何地址。PC 将被装载指令中指定的 23 位立即数。由于 PC 必须总是位于偶数地址边界，lit23<0> 将被忽略。
n 位形成目标地址。

注: 链接器将把指定的表达式解析为要使用的 lit23。

指令字数: 2

指令周期数: 2（PIC24F、PIC24H、dsPIC30F 和 dsPIC33F）
4（PIC24E 和 dsPIC33E）

例 1:

```
026000      GOTO    _THERE          ; 跳转至 _THERE
026004      MOV     W0, W1
.           ...
.           ...
027844 _THERE: MOV     #0x400, W2    ; 代码执行
027846      ...                    ; 在此重新开始
```

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 02 6000 | PC | 02 7844 |
| SR | 0000 | SR | 0000 |

例 2:

```
000100 _code: ...                ; 代码起始
.           ...
026000      GOTO    _code+2        ; 跳转至 _code+2
026004      ...
```

| | | | |
|-------|---------|-------|---------|
| 指令执行前 | | 指令执行后 | |
| PC | 02 6000 | PC | 00 0102 |
| SR | 0000 | SR | 0000 |

GOTO

无条件间接跳转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | | X |

语法: { 标号 : } GOTO Wn

操作数: Wn ∈ [W0 ... W15]

操作: 0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
0 → PC<0>
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0100 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 在前 32K 字的程序存储空间内无条件间接跳转。PC<22:16> 中将装入零而 PC<15:1> 中将装入 (Wn) 中指定的值。由于 PC 必须总是位于偶数地址边界，Wn<0> 将被忽略。
s 位用于选择源寄存器。

指令字数: 1

指令周期数: 2

例 1: 006000 GOTO W4 ; 无条件跳转至
 006002 MOV W0, W1 ; W4 中的 16 位值地址
 .
 .
 .
 007844 _THERE: MOV #0x400, W2 ; 代码执行
 007846 ... ; 在此重新开始

| | | | |
|----|---------|----|---------|
| | 指令执行前 | | 指令执行后 |
| W4 | 7844 | W4 | 7844 |
| PC | 00 6000 | PC | 00 7844 |
| SR | 0000 | SR | 0000 |

GOTO无条件间接跳转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } GOTO Wn

操作数: Wn ∈ [W0 ... W15]

操作: 0 → PC<22:16>
(Wn<15:1>) → PC<15:1>
0 → PC<0>
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0000 | 0100 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 在前 32K 字的程序存储空间内无条件间接跳转。PC<22:16> 中将装入零而 PC<15:1> 中将装入 (Wn) 中指定的值。由于 PC 必须总是位于偶数地址边界，Wn<0> 将被忽略。
s 位用于选择源寄存器。

指令字数: 1

指令周期数: 4

例 1: 006000 GOTO W4 ; 无条件跳转至
006002 MOV W0, W1 ; W4 中的 16 位值地址
...
...
007844 _THERE: MOV #0x400, W2 ; 代码执行
007846 ... ; 在此重新开始

| | | | |
|----|---------|----|---------|
| | 指令执行前 | | 指令执行后 |
| W4 | 7844 | W4 | 7844 |
| PC | 00 6000 | PC | 00 7844 |
| SR | 0000 | SR | 0000 |

GOTO.L

无条件间接长跳转

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } GOTO.L Wn

操作数: Wn ∈ [W0, W2, W4, W6, W8, W10, W12]

操作: PC<23> → PC<23> (见下文) ; (Wn+1)<6:0> → PC<22:16> ; (Wn) → PC<15:0>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 1www | w100 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 无条件间接跳转到任意用户程序存储器地址。

(Wn+1) 的低 7 位会被装入 PC<22:16>, 16 位值 (Wn) 会被装入 PC<15:0>。

该指令不会修改 PC<23>。

(Wn+1)<15:7> 的内容会被忽略。

Wn<0> 的值也会被忽略, PC<0> 总是设定为 0。

GOTO 是一条双周期指令。

s 位用于选择 Wn 源寄存器的地址。

w 位用于指定 Wn+1 源寄存器的地址。

指令字数: 1

指令周期数: 4

例 1:

| | | |
|--------|----------------------|---------------|
| 026000 | GOTO.L W4 | ; 调用 _FIR 子程序 |
| 026004 | MOV W0, W1 | |
| . | ... | |
| . | ... | |
| 026844 | _FIR: MOV #0x400, W2 | ; _FIR 子程序起始 |
| 026846 | ... | |

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 02 6000 | PC | 02 6844 |
| W4 | 6844 | W4 | 6844 |
| W5 | 0002 | W5 | 0002 |
| W15 | A268 | W15 | A26C |
| 数据单元 A268 | FFFF | 数据单元 A268 | 6004 |
| 数据单元 A26A | FFFF | 数据单元 A26A | 0002 |
| SR | 0000 | SR | 0000 |

INCf 内容递增 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} INC{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $(f) + 1 \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1100 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容加 1，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: INC.B 0x1000 ; 将 0x1000 中内容递增 1（字节模式）

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------------------|
| 数据单元 1000 | 8FFF | 数据单元 1000 | 8F00 |
| SR | 0000 | SR | 0101 (DC, C = 1) |

例 2: INC 0x1000, WREG ; 将 0x1000 中内容递增 1，并将结果存入 WREG
;（字模式）

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------------------|
| WREG | ABCD | WREG | 9000 |
| 数据单元 1000 | 8FFF | 数据单元 1000 | 8FFF |
| SR | 0000 | SR | 0108 (DC, N = 1) |

INC

Ws 内容递增 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } INC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) + 1 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1000 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容加 1，并将结果存放在目标寄存器 Wd 中。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: INC.B W1, [++W2] ; 指令执行前，递增 w2
; 递增 w1 并将结果存入 w2
; （字节模式）

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|--|-----------|--|------|-----------------|
| W1 | | FF7F | | W1 | | FF7F | |
| W2 | | 2000 | | W2 | | 2001 | |
| 数据单元 2000 | | ABCD | | 数据单元 2000 | | 80CD | |
| SR | | 0000 | | SR | | 010C | (DC, N, OV = 1) |

例 2:

INC W1, W2

; 递增 w1 并将结果存入 w2
; (字模式)

| 指令执行前 | | | 指令执行后 | | |
|-------|--|------|-------|--|------------------|
| W1 | | FF7F | W1 | | FF7F |
| W2 | | 2000 | W2 | | FF80 |
| SR | | 0000 | SR | | 0108 (DC, N = 1) |

INC2

f 内容递增 2

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } INC2{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $(f) + 2 \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1100 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容加 2，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1：在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: INC2.B 0x1000 ; 将 0x1000 中内容递增 2
; （字节模式）

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| 数据单元 1000 | 8FFF | 数据单元 1000 | 8F01 |
| SR | 0000 | SR | 0101 |

(DC, C = 1)

例 2: INC2 0x1000, WREG ; 将 0x1000 中内容递增 2，并将结果存入 WREG
; （字模式）

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| WREG | ABCD | WREG | 9001 |
| 数据单元 1000 | 8FFF | 数据单元 1000 | 8FFF |
| SR | 0000 | SR | 0108 |

(DC, N = 1)

INC2

Ws 内容递增 2

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} INC2{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) + 2 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1000 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容加 2，并将结果存放在目标寄存器 Wd 中。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: INC2.B W1, [++W2] ; 指令执行前，递增 W2
; 递增 2 并将结果存入 W1
; （字节模式）

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-----------|------|-----------------|--|
| W1 | FF7F | W1 | FF7F | W1 | FF7F | (DC, N, OV = 1) | |
| W2 | 2000 | W2 | 2001 | W2 | 2001 | | |
| 数据单元 2000 | ABCD | 数据单元 2000 | 81CD | 数据单元 2000 | 81CD | | |
| SR | 0000 | SR | 010C | SR | 010C | | |

IOR f 和 WREG 逻辑或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} IOR{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (f).IOR.(WREG) → 由 D 指定的目标寄存器

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0111 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将工作寄存器 WREG 中的内容和文件寄存器中的内容进行逻辑或运算，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: IOR.B 0x1000 ; 将 WREG 中内容和 (0x1000) 相或
; (字节模式)

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| WREG | 1234 | WREG | 1234 |
| 数据单元 1000 | FF00 | 数据单元 1000 | FF34 |
| SR | 0000 | SR | 0000 |

例 2: IOR 0x1000, WREG ; 将 (0x1000) 和 WREG 中内容相或
; (字模式)

| | | | |
|-----------|-------|------------|-------|
| | 指令执行前 | | 指令执行后 |
| WREG | 1234 | WREG | 1FBF |
| 数据单元 1000 | 0FAB | 数据单元 1000 | 0FAB |
| SR | 0008 | (N = 1) SR | 0000 |

IOR

立即数和 Wn 逻辑或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } IOR{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: lit10.IOR.(Wn) → Wn

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0011 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 10 位立即数操作数和工作寄存器 Wn 中的内容进行逻辑或运算, 并将结果存回工作寄存器 Wn。

B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。

k 位用于指定立即数操作数。

d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1

指令周期数: 1

例 1: IOR.B #0xAA, W9 ; 将 0xAA 和 W9 中内容相或
; (字节模式)

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W9 | 1234 | W9 | 12BE |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: IOR #0x2AA, W4 ; 将 0x2AA 和 W4 中内容相或
; (字模式)

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W4 | A34D | W4 | A3EF |
| SR | 0000 | SR | 0008 (N = 1) |

IOR

Wb 和短立即数逻辑或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} IOR{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: (Wb).IOR.lit5 → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0111 | 0www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将基准寄存器 Wb 中的内容和 5 位立即数操作数进行逻辑或运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1

例 1: IOR.B W1, #0x5, [W9++] ; 将 W1 中内容和 0x5 相或（字节模式）
; 结果存入 [W9]
; 指令执行后，递增 W9

| | | | | | | | |
|-----------|------|-----------|------|-----------|------|---------|--|
| 指令执行前 | | | | 指令执行后 | | | |
| W1 | AAAA | W1 | AAAA | W1 | AAAA | (N = 1) | |
| W9 | 2000 | W9 | 2001 | W9 | 2001 | | |
| 数据单元 2000 | 0000 | 数据单元 2000 | 00AF | 数据单元 2000 | 00AF | | |
| SR | 0000 | SR | 0008 | SR | 0008 | | |

例 2: IOR W1, #0x0, W9 ; 将 W1 中内容和 0x0 相或（字模式）
; 结果存入 W9

| | | | | | | | |
|-------|------|----|------|-------|------|---------|--|
| 指令执行前 | | | | 指令执行后 | | | |
| W1 | 0000 | W1 | 0000 | W1 | 0000 | (Z = 1) | |
| W9 | A34D | W9 | 0000 | W9 | 0000 | | |
| SR | 0000 | SR | 0002 | SR | 0002 | | |

IOR

Wb 和 Ws 逻辑或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} IOR{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Wb).IOR.(Ws) → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0111 | 0www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容和基准寄存器 Wb 中的内容进行逻辑或运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: IOR.B W1, [W5++], [W9++] ; 将 W1 和 [W5] 中内容相或（字节模式）
; 结果存入 [W9]
; 指令执行后，递增 W5 和 W9

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W1 | AAAA | W1 | AAAA |
| W5 | 2000 | W5 | 2001 |
| W9 | 2400 | W9 | 2401 |
| 数据单元 2000 | 1155 | 数据单元 2000 | 1155 |
| 数据单元 2400 | 0000 | 数据单元 2400 | 00FF |
| SR | 0000 | SR | 0008 |

(N = 1)

例 2:

IOR W1, W5, W9

;

将 W1 和 W5 中内容相或（字模式）

;

结果存入 W9

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W1 | AAAA | W1 | AAAA |
| W5 | 5555 | W5 | 5555 |
| W9 | A34D | W9 | FFFF |
| SR | 0000 | SR | 0008 (N = 1) |

LAC

装载累加器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } LAC Ws, {#Slit4,} Acc

 [Ws],

 [Ws++],

 [Ws--],

 [--Ws],

 [++Ws],

 [Ws+Wb],

操作数: Ws ∈ [W0 ... W15]
 Wb ∈ [W0 ... W15]
 Slit4 ∈ [-8 ... +7]
 Acc ∈ [A,B]

操作: Shift_{Slit4}(扩展 (Ws)) → Acc(A 或 B)

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1010 | Awww | wrrr | rggg | ssss |
|-------|------|------|------|------|------|------|

描述: 读取源寄存器中的内容，可选操作是对读取结果执行有符号 4 位移位操作，结果将被存入指定累加器。移位位数范围为 -8:7，其中负操作数表明将执行算术左移操作，而正操作数表明将执行算术右移操作。保存在源寄存器中的数据假定为 1.15 小数数据，且在移位操作前将自动进行符号扩展（通过 bit 39）以及零回填（bit [15:0]）。

A 位用于指定目标累加器。
w 位用于指定偏移量寄存器 Wb。
r 位用于对累加器预移位进行编码。
g 位用于选择源地址模式。
s 位用于指定源寄存器 Ws。

注: 如果该指令操作使得移入累加器最高 8 位寄存器（AccxU）的数据超出了符号扩展的范围，或导致饱和，则相应的溢出位和饱和位将置 1。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: LAC [W4++], #-3, B ; 将 [W4] << 3 装入 ACCB
 ; [W4] 中内容将不会改变
 ; 指令执行后, 递增 W4
 ; 假定饱和功能禁止
 ; (SATB = 0)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------------|
| W4 | 2000 | W4 | 2002 |
| ACCB | 00 5125 ABCD | ACCB | FF 9108 0000 |
| 数据单元 2000 | 1221 | 数据单元 2000 | 1221 |
| SR | 0000 | SR | 4800 (OB, OAB = 1) |

例 2: LAC [--W2], #7, A ; 指令执行前, 递减 W2
 ; 将 [W2] >> 7 装入 ACCA
 ; [W2] 中内容将不会改变
 ; 假定饱和功能禁止
 ; (SATA = 0)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W2 | 4002 | W2 | 4000 |
| ACCA | 00 5125 ABCD | ACCA | FF FF22 1000 |
| 数据单元 4000 | 9108 | 数据单元 4000 | 9108 |
| 数据单元 4002 | 1221 | 数据单元 4002 | 1221 |
| SR | 0000 | SR | 0000 |

LNK

分配堆栈帧

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 ;} LNK #lit14

操作数: lit14 ∈ [0 ... 16382]

操作:
(W14) → (TOS)
(W15) + 2 → W15
(W15) → W14
(W15) + lit14 → W15

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1010 | 00kk | kkkk | kkkk | kkk0 |
|-------|------|------|------|------|------|------|

描述: 该指令将为子程序调用序列分配一个大小为 **lit14** 字节的堆栈帧。堆栈帧的分配是通过将帧指针 (**W14**) 的内容压入堆栈, 将更新后的堆栈指针 (**W15**) 存入帧指针, 随后递增堆栈指针, 递增值由 **14** 位无符号立即数确定。该指令可支持最大 **16382** 字节的堆栈帧。

k 位用于指定堆栈帧的大小。

注: 由于堆栈指针只能位于字边界, 因此 **lit14** 必须是偶数。

指令字数: 1

指令周期数: 1

例 1: LNK #0xA0 ; 分配一个 160 字节的堆栈帧

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|--|-----------|--|------|--|
| W14 | | 2000 | | W14 | | 2002 | |
| W15 | | 2000 | | W15 | | 20A2 | |
| 数据单元 2000 | | 0000 | | 数据单元 2000 | | 2000 | |
| SR | | 0000 | | SR | | 0000 | |

LNK分配堆栈帧

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 ;} LNK #lit14

操作数: lit14 ∈ [0 ... 16382]

操作:
(W14) → (TOS)
(W15) + 2 → W15
(W15) → W14
1 → SFA 位
(W15) + lit14 → W15

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1010 | 00kk | kkkk | kkkk | kkk0 |
|-------|------|------|------|------|------|------|

描述: 该指令将为子程序调用序列分配一个大小为 **lit14** 字节的堆栈帧。堆栈帧的分配是通过将帧指针 (**W14**) 的内容压入堆栈, 将更新后的堆栈指针 (**W15**) 存入帧指针, 随后递增堆栈指针, 递增值由 **14** 位无符号立即数确定。该指令可支持最大 **16382** 字节的堆栈帧。

k 位用于指定堆栈帧的大小。

注: 由于堆栈指针只能位于字边界, 因此 **lit14** 必须是偶数。

指令字数: 1

指令周期数: 1

例 1: LNK #0xA0 ; 分配一个 160 字节的堆栈帧

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W14 | 2000 | W14 | 2002 |
| W15 | 2000 | W15 | 20A2 |
| 数据单元 2000 | 0000 | 数据单元 2000 | 2000 |
| SR | 0000 | SR | 0000 |
| CORCON | 0000 | CORCON | 0004 |

LSR


逻辑右移 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} LSR{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
0 → Dest<7>
(f<7:1>) → Dest<6:0>
(f<0>) → C
对于字操作:
0 → Dest<15>
(f<15:1>) → Dest<14:0>
(f<0>) → C

0 → 

受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0101 | 0Bdf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容右移一位，并将结果存放在目标寄存器中。文件寄存器中的最低位将被移入状态寄存器的进位位。目标寄存器的最高位将移入 0。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: LSR.B 0x600 ; 将 (0x600) 逻辑右移 1 位
; (字节模式)

| | | | |
|----------|------|----------|--------------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 600 | 55FF | 数据单元 600 | 557F |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: LSR 0x600, WREG ; 将 (0x600) 逻辑右移 1 位
 ; 结果存入 WREG
 ; (字模式)

| | 指令执行前 | | 指令执行后 |
|----------|-------|----------|--------------|
| 数据单元 600 | 55FF | 数据单元 600 | 55FF |
| WREG | 0000 | WREG | 2AFF |
| SR | 0000 | SR | 0001 (C = 1) |

LSR

逻辑右移 Ws

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} LSR{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
0 → Wd<7>
(Ws<7:1>) → Wd<6:0>
(Ws<0>) → C
对于字操作:
0 → Wd<15>
(Ws<15:1>) → Wd<14:0>
(Ws<0>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0001 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容右移一位，并将结果存放在目标寄存器 Wd 中。Ws 中的最低位将被移入状态寄存器的进位位。Wd 的最高位将移入 0。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

- B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。
- 注：** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1： 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: LSR.B W0, W1 ; 将 W0 中内容逻辑右移 1 位 （字节模式）
 ; 结果存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | FF03 | W0 | FF03 |
| W1 | 2378 | W1 | 2301 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: LSR W0, W1 ; 将 W0 中内容逻辑右移 1 位 （字模式）
 ; 结果存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 8000 | W0 | 8000 |
| W1 | 2378 | W1 | 4000 |
| SR | 0000 | SR | 0000 |

LSR

逻辑右移（移位位数由短立即数确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } LSR Wb, #lit4, Wnd

操作数： Wb ∈ [W0 ... W15]
lit4 ∈ [0 ... 15]
Wnd ∈ [W0 ... W15]

操作： lit4<3:0> → Shift_Val
0 → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

受影响的状态位： N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1101 | 1110 | 0www | wddd | d100 | kkkk |
|-------|------|------|------|------|------|------|

描述： 将源寄存器 Wb 中的内容进行逻辑右移，移位位数由 4 位无符号立即数确定，并将结果存入目标寄存器 Wnd。必须使用直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
k 位用于提供立即数操作数。

注： 该指令只能工作于字模式。

指令字数： 1

指令周期数： 1

例 1: LSR W4, #14, W5 ; 将 W4 中内容逻辑右移 14 位
; 结果存入 W5

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W4 | C800 | W4 | C800 |
| W5 | 1200 | W5 | 0003 |
| SR | 0000 | SR | 0000 |

例 2: LSR W4, #1, W5 ; 将 W4 中内容逻辑右移 1 位
; 结果存入 W5

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W4 | 0505 | W4 | 0505 |
| W5 | F000 | W5 | 0282 |
| SR | 0000 | SR | 0000 |

LSR

逻辑右移（移位位数由 Wns 确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } LSR Wb, Wns, Wnd

操作数： Wb ∈ [W0 ... W15]
Wns ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作： Wns<4:0> → Shift_Val
0 → Wnd<15:15-Shift_Val + 1>
Wb<15:Shift_Val> → Wnd<15-Shift_Val:0>

受影响的状态位： N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1101 | 1110 | 0www | wddd | d000 | ssss |
|-------|------|------|------|------|------|------|

描述： 将源寄存器 Wb 中的内容进行逻辑右移，移位位数由 Wns 的低 5 位确定（最大移位位数为 15 位），并将结果存入目标寄存器 Wnd。必须使用直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
s 位用于选择源寄存器。

注 1: 该指令只能工作于字模式。
2: 如果 Wns 大于 15，Wnd 将被装入 0x0。

指令字数： 1

指令周期数： 1

例 1: LSR W0, W1, W2 ; 将 W0 中的内容逻辑右移，移位位数为 W1 中内容
; 结果存入 W2

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | C00C | W0 | C00C |
| W1 | 0001 | W1 | 0001 |
| W2 | 2390 | W2 | 6006 |
| SR | 0000 | SR | 0000 |

例 2: LSR W5, W4, W3 ; 将 W5 中的内容逻辑右移，移位位数为 W4 中内容
; 结果存入 W3

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W3 | DD43 | W3 | 0000 |
| W4 | 000C | W4 | 000C |
| W5 | 0800 | W5 | 0800 |
| SR | 0000 | SR | 0002 (Z = 1) |

MAC

相乘并累加

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} MAC Wm*Wn, Acc {[Wx], Wxd} {[Wy],Wyd} {,AWB}
[Wx] += kx, Wxd [Wy] += ky, Wyd
[Wx] -= kx, Wxd [Wy] -= ky, Wyd
[W9 + W12],Wxd [W11 + W12],Wyd

操作数: Wm * Wn ∈ [W4 * W5, W4 * W6, W4 * W7, W5 * W6, W5 * W7, W6 * W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6] ; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6] ; Wyd ∈ [W4 ... W7]
AWB ∈ [W13, [W13] + = 2]

操作: (Acc(A 或 B)) + (Wm) * (Wn) → Acc(A 或 B)
([Wx]) → Wxd ; (Wx) + kx → Wx
([Wy]) → Wyd ; (Wy) + ky → Wy
舍入后的 (Acc(B 或 A)) → AWB

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|-------|------|------|------|------|
| 指令编码: | 1100 | 0mmmm | A0xx | yyii | iijj | jjaa |
|-------|------|-------|------|------|------|------|

描述: 将两个工作寄存器的内容相乘，可选操作包括预取操作数以为另一条 MAC 类指令作准备以及保存未指定累加器的结果。有符号乘法操作的 32 位结果将符号扩展为 40 位并加到指定的累加器。

如第 4.14.1 节“MAC 预取”中所介绍，操作数 Wx、Wxd、Wy 和 Wyd 用于指定可选的预取操作，上述操作数支持间接寻址和寄存器偏移量寻址。如第 4.14.4 节“MAC 回写”中所介绍，操作数 AWB 用于指定可选操作：存储“另一个”累加器的内容。

m 位用于选择乘法操作的操作数寄存器 Wm 和 Wn。
A 位用于选择存放结果的累加器。
x 位用于选择预取 Wxd 的目标寄存器。
y 位用于选择预取 Wyd 的目标寄存器。
i 位用于选择 Wx 预取操作。
j 位用于选择 Wy 预取操作。
a 位用于选择累加器回写操作的目标寄存器。

- 注 1: IF 位（CORCON<0>）用于确定乘法操作采用小数形式还是整数形式。
- 2: US<1:0> 位（dsPIC33E 器件中的 CORCON<13:12>，dsPIC30F/dsPIC33F 器件中的 CORCON<12>）用于确定乘法是无符号、有符号还是混合符号的。仅 dsPIC33E 器件支持混合符号乘法。

指令字数: 1
指令周期数: 1

```
MAC    W4*W5, A, [W8]+=6, W4, [W10]+=2, W5
; 执行乘法操作 W4*W5 并将结果加到 ACCA
; 取操作数 [W8] 至 W4, 指令执行后将 W8 递增 6
; 取操作数 [W10] 至 W5, 指令执行后将 W10 递增 2
; CORCON = 0x00C0 (小数乘法, 普通饱和模式)
```

| | | |
|----|------|---------------|
| SR | 8800 | (OA, OAB = 1) |
|----|------|---------------|

```
MAC    W4*W5, A, [W8]=-2, W4, [W10]+=2, W5, W13
; 执行乘法操作 W4*W5 并将结果加到 ACCA
; 取操作数 [W8] 至 W4, 指令执行后将 W8 递减 2
; 取操作数 [W10] 至 W5, 指令执行后将 W10 递增 2
; 回写 ACCB 至 W13
; CORCON = 0x00D0 (小数乘法, 超饱和模式)
```

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | 1000 | W4 | 5BBE |
| W5 | 3000 | W5 | C967 |
| W8 | 0A00 | W8 | 09FE |
| W10 | 1800 | W10 | 1802 |
| W13 | 2000 | W13 | 0001 |
| ACCA | 23 5000 2000 | ACCA | 23 5600 2000 |
| ACCB | 00 0000 8F4C | ACCB | 00 0000 1F4C |
| 数据单元 0A00 | 5BBE | 数据单元 0A00 | 5BBE |
| 数据单元 1800 | C967 | 数据单元 1800 | C967 |
| CORCON | 00D0 | CORCON | 00D0 |
| SR | 0000 | SR | 8800 |

平方并累加

例 1: MAC W4*W4, B, [W9+W12], W4, [W10]-=2, W5
 ; 将 W4 平方并将结果加到 ACCB
 ; 取操作数 [W9+W12] 至 W4
 ; 取操作数 [W10] 至 W5, 指令执行后将 W10 递减 2
 ; CORCON = 0x00C0 (小数乘法, 普通饱和模式)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | A022 | W4 | A230 |
| W5 | B200 | W5 | 650B |
| W9 | 0C00 | W9 | 0C00 |
| W10 | 1900 | W10 | 18FE |
| W12 | 0020 | W12 | 0020 |
| ACCB | 00 2000 0000 | ACCB | 00 67CD 0908 |
| 数据单元 0C20 | A230 | 数据单元 0C20 | A230 |
| 数据单元 1900 | 650B | 数据单元 1900 | 650B |
| CORCON | 00C0 | CORCON | 00C0 |
| SR | 0000 | SR | 0000 |

例 2: MAC W7*W7, A, [W11]-=2, W7
 ; 将 W7 平方并将结果加到 ACCA
 ; 取操作数 [W11] 至 W7, 指令执行后将 W11 递减 2
 ; CORCON = 0x00D0 (小数乘法, 超饱和模式)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W7 | 76AE | W7 | 23FF |
| W11 | 2000 | W11 | 1FFE |
| ACCA | FE 9834 4500 | ACCA | FF 063E 0188 |
| 数据单元 2000 | 23FF | 数据单元 2000 | 23FF |
| CORCON | 00D0 | CORCON | 00D0 |
| SR | 0000 | SR | 8800 |

(OA, OAB = 1)

MOV

将 f 中内容传送到目标寄存器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MOV{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: (f) → 由 D 指定的目标寄存器

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1111 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将指定文件寄存器中的内容传送到目标寄存器。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存回文件寄存器，指令操作仅对状态寄存器进行修改。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

3: 当传送来自文件寄存器存储区的字数据时，“将 f 中内容传送到 Wnd”指令（第 281 页）允许将任何工作寄存器（W0:W15）作为目标寄存器。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MOV.B TMR0, WREG ; 将 (TMR0) 传送到 WREG（字节模式）

| | 指令执行前 | | 指令执行后 |
|-----------|-------|-----------|-------|
| WREG (W0) | 9080 | WREG (W0) | 9055 |
| TMR0 | 2355 | TMR0 | 2355 |
| SR | 0000 | SR | 0000 |

例 2: MOV 0x800 ; 基于 (0x800) 更新 SR（字模式）

| | 指令执行前 | | 指令执行后 |
|-----------|-------|-----------|--------------|
| 数据单元 0800 | B29F | 数据单元 0800 | B29F |
| SR | 0000 | SR | 0008 (N = 1) |

MOV

将 WREG 中内容传送到 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MOV{.B} WREG, f

操作数: f ∈ [0 ... 8191]

操作: (WREG) → f

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0111 | 1B1f | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将默认工作寄存器 WREG 中的内容传送到指定的文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .w 扩展符指明是字传送，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。
- 3: 当将字数据从工作寄存器阵列传送到文件寄存器存储区时，“将 Wns 中内容传送到 f”指令（第 282 页）允许将任何工作寄存器（W0:W15）作为源寄存器。

指令字数: 1

指令周期数: 1

例 1: MOV.B WREG, 0x801 ; 将 WREG 内容传送到 0x801（字节模式）

| | | | | | |
|-----------|------|--|-----------|------|---------|
| 指令执行前 | | | 指令执行后 | | |
| WREG (W0) | 98F3 | | WREG (W0) | 98F3 | |
| 数据单元 0800 | 4509 | | 数据单元 0800 | F309 | |
| SR | 0000 | | SR | 0008 | (N = 1) |

例 2: MOV WREG, DISICNT ; 将 WREG 内容传送到 DISICNT

| | | | | | |
|-----------|------|--|-----------|------|--|
| 指令执行前 | | | 指令执行后 | | |
| WREG (W0) | 00A0 | | WREG (W0) | 00A0 | |
| DISICNT | 0000 | | DISICNT | 00A0 | |
| SR | 0000 | | SR | 0000 | |

MOV

将 f 中内容传送至 Wnd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MOV f, Wnd

操作数: f ∈ [0 ... 65534]
Wnd ∈ [W0 ... W15]

操作: (f) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1000 | 0fff | ffff | ffff | ffff | dddd |
|-------|------|------|------|------|------|------|

描述: 将指定文件寄存器中的字内容传送至 Wnd。文件寄存器可以位于 32K 字数据存储器区中的任何地址，但必须符合字对齐的原则。必须使用寄存器直接寻址对 Wnd 进行寻址。

f 位用于选择文件寄存器的地址。
d 位用于选择目标寄存器。

- 注 1: 该指令只能对字操作数进行操作。
2: 由于文件寄存器地址必须为字对齐的，因此只对文件寄存器地址的高 15 位进行编码 (bit 0 假定为 0)。
3: 可使用“将 f 中内容传送至目标寄存器”指令 (第 279 页) 实现对文件寄存器存储区的数据字节传送。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MOV CORCON, W12 ; 将 CORCON 内容传送至 W12

| | | | |
|--------|------|--------|------|
| 指令执行前 | | 指令执行后 | |
| W12 | 78FA | W12 | 00F0 |
| CORCON | 00F0 | CORCON | 00F0 |
| SR | 0000 | SR | 0000 |

例 2: MOV 0x27FE, W3 ; 将 (0x27FE) 传送至 W3

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| W3 | 0035 | W3 | ABCD |
| 数据单元 27FE | ABCD | 数据单元 27FE | ABCD |
| SR | 0000 | SR | 0000 |

MOV将 Wns 中内容传送至 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MOV Wns, f

操作数: f ∈ [0 ... 65534]
Wns ∈ [W0 ... W15]

操作: (Wns) → f

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1000 | 1fff | ffff | ffff | ffff | ssss |
|-------|------|------|------|------|------|------|

描述: 将工作寄存器 Wns 中的字内容传送至指定的文件寄存器。文件寄存器可以位于 32K 字数据存储区中的任何地址，但必须符合字对齐的原则。必须使用寄存器直接寻址对 Wn 进行寻址。

f 位用于选择文件寄存器的地址。
s 位用于选择源寄存器。

- 注
- 1: 该指令只能对字操作数进行操作。
 - 2: 由于文件寄存器地址必须为字对齐的，因此只对文件寄存器地址的高 15 位进行编码（bit 0 假定为 0）。
 - 3: 可使用“将 WREG 中内容传送至 f”指令（第 280 页）传送一个字节的数 据至文件寄存器存储区。

指令字数: 1

指令周期数: 1

例 1: MOV W4, XMODSRT ; 将 W4 内容传送至 XMODSRT

| 指令执行前 | | | 指令执行后 | | |
|---------|------|--|---------|------|--|
| W4 | 1200 | | W4 | 1200 | |
| XMODSRT | 1340 | | XMODSRT | 1200 | |
| SR | 0000 | | SR | 0000 | |

例 2: MOV W8, 0x1222 ; 将 W8 内容传送至数据单元地址 0x1222

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|--|-----------|------|--|
| W8 | F200 | | W8 | F200 | |
| 数据单元 1222 | FD88 | | 数据单元 1222 | F200 | |
| SR | 0000 | | SR | 0000 | |

MOV.B

将 8 位立即数传送至 Wnd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MOV.B #lit8, Wnd

操作数: lit8 ∈ [0 ... 255]
Wnd ∈ [W0 ... W15]

操作: lit8 → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0011 | 1100 | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 8 位无符号立即数 k 装入 Wnd 中的低位字节。Wnd 中的高位字节将不会改变。必须使用寄存器直接寻址对 Wnd 进行寻址。

k 位用于指定立即数的值。
d 位用于选择工作寄存器的地址。

注: 该指令工作于字节模式且必须提供 .B 扩展符。

指令字数: 1

指令周期数: 1

例 1: MOV.B #0x17, W5 ; 将 #0x17 装入 W5 (字节模式)

| | | | |
|----|-------|----|-------|
| | 指令执行前 | | 指令执行后 |
| W5 | 7899 | W5 | 7817 |
| SR | 0000 | SR | 0000 |

例 2: MOV.B #0xFE, W9 ; 将 #0xFE 装入 W9 (字节模式)

| | | | |
|----|-------|----|-------|
| | 指令执行前 | | 指令执行后 |
| W9 | AB23 | W9 | ABFE |
| SR | 0000 | SR | 0000 |

MOV将 16 位立即数传送至 Wnd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MOV #lit16, Wnd

操作数: lit16 ∈ [-32768 ... 65535]
Wnd ∈ [W0 ... W15]

操作: lit16 → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0010 | kkkk | kkkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 16 位立即数 k 装入 Wnd。必须使用寄存器直接寻址对 Wnd 进行寻址。

k 位用于指定立即数的值。

d 位用于选择工作寄存器的地址。

- 注 1: 该指令只能工作于字模式。
2: 立即数可指定为有符号值 [-32768:32767] 或无符号值 [0:65535]。

指令字数: 1

指令周期数: 1

例 1: MOV #0x4231, W13 ; 将 #0x4231 装入 W13

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W13 | 091B | W13 | 4231 |
| SR | 0000 | SR | 0000 |

例 2: MOV #0x4, W2 ; 将 #0x4 装入 W2

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W2 | B004 | W2 | 0004 |
| SR | 0000 | SR | 0000 |

例 3: MOV #-1000, W8 ; 将 #-1000 装入 W8

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W8 | 23FF | W8 | FC18 |
| SR | 0000 | SR | 0000 |

MOV

将 [Ws + 偏移量] 中内容传送至 Wnd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MOV{.B} [Ws + Slit10], Wnd

操作数: Ws ∈ [W0 ... W15]
对于字节操作, Slit10 ∈ [-512 ... 511]
对于字操作, Slit10 ∈ [-1024 ... 1022] (仅限偶数)
Wnd ∈ [W0 ... W15]

操作: [Ws + Slit10] → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1001 | 0kkk | kBkk | kddd | dkkk | ssss |
|-------|------|------|------|------|------|------|

描述: 将 [Ws + Slit10] 中内容装入 Wnd。在字模式下, Slit10 的范围增加至 [-1024 ... 1022], 而 Slit10 必须是偶数以保持字地址对齐。必须使用寄存器间接寻址对源寄存器进行寻址。必须使用直接寻址对 Wnd 进行寻址。

k 位用于指定立即数的值。
B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
d 位用于选择目标寄存器。
s 位用于选择源寄存器。

- 注 1: 指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .w 扩展符指明是字传送, 但这并不需要。
- 2: 在字节模式下, 由于采用立即数表示距离 Ws 的地址偏移量, Slit10 的范围并不像第 4.6 节 “使用 10 位立即数操作数” 所说明那样被减小。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: MOV.B [W8+0x13], W10 ; 将 [W8+0x13] 装入 W10
; (字节模式)

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W8 | 1008 | W8 | 1008 |
| W10 | 4009 | W10 | 4033 |
| 数据单元 101A | 3312 | 数据单元 101A | 3312 |
| SR | 0000 | SR | 0000 |

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W2 | 9088 | W2 | 5634 |
| W4 | 0800 | W4 | 0800 |
| 数据单元 0BE8 | 5634 | 数据单元 0BE8 | 5634 |
| SR | 0000 | SR | 0000 |

将 Wns 中内容传送至 [Wd + 偏移量]

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 9015 | W0 | 9015 |
| W1 | 1800 | W1 | 1800 |
| 数据单元 1806 | 2345 | 数据单元 1806 | 1545 |
| SR | 0000 | SR | 0000 |

MOV将 Ws 中内容传送至 Wd

指令字数：1
指令周期数：1⁽¹⁾

注 1：在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1：MOV.B [W0--], W4 ; 将 [W0] 传送至 W4（字节模式）
; 指令执行后，递减 W0

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 0A01 | W0 | 0A00 |
| W4 | 2976 | W4 | 2989 |
| 数据单元 0A00 | 8988 | 数据单元 0A00 | 8988 |
| SR | 0000 | SR | 0000 |

例 2：MOV [W6++], [W2+W3] ; 将 [W6] 传送至 [W2+W3]（字模式）
; 指令执行后，递增 W6

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W2 | 0800 | W2 | 0800 |
| W3 | 0040 | W3 | 0040 |
| W6 | 1228 | W6 | 122A |
| 数据单元 0840 | 9870 | 数据单元 0840 | 0690 |
| 数据单元 1228 | 0690 | 数据单元 1228 | 0690 |
| SR | 0000 | SR | 0000 |

MOV.D

双字传送源寄存器内容至 Wnd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MOV.D Wns, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wns ∈ [W0, W2, W4 ... W14]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W14]

操作: 对于源寄存器直接寻址:
Wns → Wnd
Wns + 1 → Wnd + 1
对于源寄存器间接寻址:
参见指令描述

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1110 | 0000 | 0ddd | 0ppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器指定的双字内容传送至目标工作寄存器对 (Wnd:Wnd + 1)。如果使用寄存器直接寻址对源寄存器进行寻址，两个连续工作寄存器 (Wns:Wns + 1) 中的内容将被传送至 Wnd:Wnd + 1。如果使用间接寻址对源寄存器进行寻址，Ws 用于指定双字中低位字的有效地址。执行任何前 / 后递增或前/后递减操作都将以4个字节为单位对Ws进行调整以符合双字规范。

d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择第一个源寄存器的地址。

- 注 1: 该指令只能对双字进行操作。有关在存储区中双字如何对齐的信息，请参见图 4-3。
2: Wnd 必须是偶数编号的工作寄存器。
3: 指令 “POP.D Wnd” 可转换为 MOV.D [--W15], Wnd。

指令字数: 1
指令周期数: 2⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节 “多周期指令” 中的注 3。

例 1: MOV.D W2, W6 ; 将 W2 内容传送至 W6（双字模式）

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W2 | 12FB | W2 | 12FB |
| W3 | 9877 | W3 | 9877 |
| W6 | 9833 | W6 | 12FB |
| W7 | FCC6 | W7 | 9877 |
| SR | 0000 | SR | 0000 |

例 2: MOV.D [W7--], W4 ; 将 [W7] 传送至 W4（双字模式）
 ; 指令执行后，递减 W7

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W4 | B012 | W4 | A319 |
| W5 | FD89 | W5 | 9927 |
| W7 | 0900 | W7 | 08FC |
| 数据单元 0900 | A319 | 数据单元 0900 | A319 |
| 数据单元 0902 | 9927 | 数据单元 0902 | 9927 |
| SR | 0000 | SR | 0000 |

MOVWAG

将立即数传送到页寄存器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } MOVWAG #lit10, DSRWAG
#lit9, DSWWAG
#lit8, TBLWAG

操作数: lit10 ∈ [0 ... 1023], lit9 ∈ [0 ... 511], lit8 ∈ [0 ... 255]

操作: lit10 → DSRWAG 或 lit9 → DSWWAG 或 lit8 → TBLWAG

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 1100 | PPkk | kkkk | kkkk |
|-------|------|------|------|------|------|------|

描述: 无符号立即数“k”中相应的位数会被装入DSRWAG、DSWWAG或TBLWAG寄存器。目标寄存器为 DSWWAG 时，汇编器会将立即数限制为 9 位无符号值；目标寄存器为 TBLWAG 时，它会将立即数限制为 8 位无符号值。

P 位用于选择目标寄存器。

k 位用于指定立即数的值。

注: 该指令只能工作于字模式。

指令字数: 1

指令周期数: 1

例 1: MOVWAG #0x02, DSRWAG

| | | | |
|--------|-------|--------|-------|
| | 指令执行前 | | 指令执行后 |
| DSRWAG | 0000 | DSRWAG | 0002 |

MOVWAG 将 Ws 中内容传送到页寄存器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } MOVWAG Wn, DSRWAG
DSWAG
TBLWAG

操作数: Wn ∈ [W0 ... W15]
操作: Wn<9:0> → DSRWAG 或 Wn<8:0> → DSWAG 或 Wn<7:0> → TBLWAG
受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 1101 | PP00 | 0000 | SSSS |
|-------|------|------|------|------|------|------|

描述: 寄存器 Ws 中相应的位数会被装入 DSRWAG、DSWAG 或 TBLWAG 寄存器。目标寄存器为 DSWAG 时，汇编器会将立即数限制为 9 位无符号值；目标寄存器为 TBLWAG 时，它会将立即数限制为 8 位无符号值。
P 位用于选择目标寄存器。
s 位用于指定源寄存器。

注: 该指令只能工作于字模式。

指令字数: 1
指令周期数: 1

例 1: MOVWAG W2, DSRWAG

| | | | |
|--------|-------|--------|-------|
| | 指令执行前 | | 指令执行后 |
| DSRWAG | 0000 | DSRWAG | 0002 |
| W2 | 0002 | W2 | 0002 |

MPY

Wm 和 Wn 相乘，结果存入累加器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法： { 标号 : } MPY Wm * Wn, Acc {,[Wx],Wxd} {,[Wy],Wyd}
{,[Wx] + = kx, Wxd} {,[Wy] + = ky, Wyd}
{,[Wx] - = kx, Wxd} {,[Wy] - = ky, Wyd}
{,[W9 + W12],Wxd} {,[W11 + W12],Wyd}

操作数： Wm * Wn ∈ [W4 * W5, W4 * W6, W4 * W7, W5 * W6, W5 * W7, W6 * W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6] ; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6] ; Wyd ∈ [W4 ... W7]
AWB ∈ [W13], [W13] + = 2

操作： (Wm) * (Wn) → Acc(A 或 B)
([Wx]) → Wxd ; (Wx) + kx → Wx
([Wy]) → Wyd ; (Wy) + ky → Wy

受影响的状态位： OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|-------|-------|------|------|------|
| 指令编码： | 1100 | 0mmmm | A0xxx | yyii | iijj | jj11 |
|-------|------|-------|-------|------|------|------|

描述： 将两个工作寄存器的内容相乘，可选操作包括预取操作数以为另一条 MAC 类指令作准备。有符号乘法操作的 32 位结果将符号扩展为 40 位并存入指定的累加器。

如第 4.14.1 节“MAC 预取”中所介绍，操作数 Wx、Wxd、Wy 和 Wyd 用于指定可选的预取操作，上述操作数支持间接寻址和寄存器偏移量寻址。

m 位用于选择乘法操作的操作数寄存器 Wm 和 Wn。

A 位用于选择存放结果的累加器。

x 位用于选择预取 Wxd 的目标寄存器。

y 位用于选择预取 Wyd 的目标寄存器。

i 位用于选择 Wx 预取操作。

j 位用于选择 Wy 预取操作。

注 1： IF 位（CORCON<0>）用于确定乘法操作采用小数形式还是整数形式。

2： US<1:0> 位（dsPIC33E 器件中的 CORCON<13:12>，dsPIC30F/dsPIC33F 器件中的 CORCON<12>）用于确定乘法是无符号、有符号还是混合符号的。仅 dsPIC33E 器件支持混合符号乘法。

指令字数： 1

指令周期数： 1

例 1: MPY W4*W5, A, [W8]+=2, W6, [W10]-=2, W7
 ; 执行乘法操作 W4*W5 并将结果存入 ACCA
 ; 取操作数 [W8] 至 W6, 指令执行后将 W8 递增 2
 ; 取操作数 [W10] 至 W7, 指令执行后将 W10 递减 2
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | C000 | W4 | C000 |
| W5 | 9000 | W5 | 9000 |
| W6 | 0800 | W6 | 671F |
| W7 | B200 | W7 | E3DC |
| W8 | 1780 | W8 | 1782 |
| W10 | 2400 | W10 | 23FE |
| ACCA | FF F780 2087 | ACCA | 00 3800 0000 |
| 数据单元 1780 | 671F | 数据单元 1780 | 671F |
| 数据单元 2400 | E3DC | 数据单元 2400 | E3DC |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

例 2: MPY W6*W7, B, [W8]+=2, W4, [W10]-=2, W5
 ; 执行乘法操作 W6*W7 并将结果存入 ACCB
 ; 取操作数 [W8] 至 W4, 指令执行后将 W8 递增 2
 ; 取操作数 [W10] 至 W5, 指令执行后将 W10 递减 2
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | C000 | W4 | 8FDC |
| W5 | 9000 | W5 | 0078 |
| W6 | 671F | W6 | 671F |
| W7 | E3DC | W7 | E3DC |
| W8 | 1782 | W8 | 1784 |
| W10 | 23FE | W10 | 23FC |
| ACCB | 00 9834 4500 | ACCB | FF E954 3748 |
| 数据单元 1782 | 8FDC | 数据单元 1782 | 8FDC |
| 数据单元 23FE | 0078 | 数据单元 23FE | 0078 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

求平方，结果存入累加器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

| | | | |
|-----|--------------------------|--------------------|---------------------|
| 语法: | { 标号 :} MPY Wm * Wm, Acc | {,[Wx], Wxd} | {,[Wy], Wyd} |
| | | {,[Wx] += kx, Wxd} | {,[Wy] += ky, Wyd} |
| | | {,[Wx] -= kx, Wxd} | {,[Wy] -= ky, Wyd} |
| | | {,[W9 + W12], Wxd} | {,[W11 + W12], Wyd} |

操作数: Wm * Wm ∈ [W4 * W4, W5 * W5, W6 * W6, W7 * W7]
 Acc ∈ [A,B]
 Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6] ; Wxd ∈ [W4 ... W7]
 Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6] ; Wyd ∈ [W4 ... W7]

操作: $(Wm) * (Wm) \rightarrow Acc(A \text{ 或 } B)$
 $([Wx]) \rightarrow Wxd ; (Wx) + kx \rightarrow Wx$
 $([Wy]) \rightarrow Wyd ; (Wy) + ky \rightarrow Wy$

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 00mm | A0xx | yyii | ijjj | jj01 |
|-------|------|------|------|------|------|------|

描述: 将工作寄存器中的内容平方, 可选操作包括预取操作数以为另一条 MAC 类指令作准备。有符号乘法操作的 32 位结果将符号扩展为 40 位并存入指定的累加器。

如第 4.14.1 节“MAC 预取”中所介绍，操作数 Wx、Wxd、Wy 和 Wyd 用于指定可选的预取操作，上述操作数支持间接寻址和寄存器偏移量寻址。

m 位用于选择平方操作的操作数寄存器 Wm。

A 位用于选择存放结果的累加器。

x 位用于选择预取 Wxd 的目标寄存器。

y 位用于选择预取 W_yd 的目标寄存器。

i 位用于选择 Wx 预取操作。

i 位用于选择 W_y 预取操作。

注 1: IF 位 (CORCON<0>) 用于确定乘法操作采用小数形式还是整数形式。

2: US<1:0> 位 (dsPIC33E 器件中的 CORCON<13:12>, dsPIC30F/dsPIC33F 器件中的 CORCON<12>) 用于确定乘法是无符号、有符号还是混合符号的。仅 dsPIC33E 器件支持混合符号乘法。

指令字数: 1

指令周期数: 1

例 1: MPY W6*W6, A, [W9]+=2, W6
 ; 将 W6 平方并将结果存入 ACCA
 ; 取操作数 [W9] 至 W6, 指令执行后将 W9 递增 2
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W6 | 6500 | W6 | B865 |
| W9 | 0900 | W9 | 0902 |
| ACCA | 00 7C80 0908 | ACCA | 00 4FB2 0000 |
| 数据单元 0900 | B865 | 数据单元 0900 | B865 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

例 2: MPY W4*W4, B, [W9+W12], W4, [W10]+=2, W5
 ; 将 W4 平方并将结果存入 ACCB
 ; 取操作数 [W9+W12] 至 W4
 ; 取操作数 [W10] 至 W5, 指令执行后将 W10 递增 2
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | E228 | W4 | 8911 |
| W5 | 9000 | W5 | F678 |
| W9 | 1700 | W9 | 1700 |
| W10 | 1B00 | W10 | 1B02 |
| W12 | FF00 | W12 | FF00 |
| ACCB | 00 9834 4500 | ACCB | 00 06F5 4C80 |
| 数据单元 1600 | 8911 | 数据单元 1600 | 8911 |
| 数据单元 1B00 | F678 | 数据单元 1B00 | F678 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

MPY.N

-Wm 和 Wn 相乘，结果存入累加器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法： { 标号 : } MPY.N Wm * Wn, Acc {,[Wx], Wxd} {,[Wy],Wyd}
{,[Wx] += kx, Wxd} {,[Wy] += ky, Wyd}
{,[Wx] -= kx, Wxd} {,[Wy] -= ky, Wyd}
{,[W9 + W12], Wxd} {,[W11 + W12],Wyd}

操作数： Wm * Wn ∈ [W4 * W5; W4 * W6; W4 * W7; W5 * W6; W5 * W7; W6 * W7]
Acc ∈ [A,B]
Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6] ; Wxd ∈ [W4 ... W7]
Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6] ; Wyd ∈ [W4 ... W7]

操作： -(Wm) * (Wn) → Acc(A 或 B)
([Wx]) → Wxd ; (Wx) + kx → Wx
([Wy]) → Wyd ; (Wy) + ky → Wy

受影响的状态位： OA、OB 和 OAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1100 | 0mmm | A1xx | yyii | iijj | jj11 |
|-------|------|------|------|------|------|------|

描述： 将一个工作寄存器的内容与另一个工作寄存器的内容取反的结果相乘，可选操作包括预取操作数以为另一条 MAC 类指令作准备以及保存未指定累加器的结果。有符号乘法操作的 32 位结果将符号扩展为 40 位并存入指定的累加器。

m 位用于选择乘法操作的操作数寄存器 Wm 和 Wn。
A 位用于选择存放结果的累加器。
x 位用于选择预取 Wxd 的目标寄存器。
y 位用于选择预取 Wyd 的目标寄存器。
i 位用于选择 Wx 预取操作。
j 位用于选择 Wy 预取操作。

注 1: IF 位（CORCON<0>）用于确定乘法操作采用小数形式还是整数形式。

2: US<1:0> 位（dsPIC33E 器件中的 CORCON<13:12>，dsPIC30F/dsPIC33F 器件中的 CORCON<12>）用于确定乘法是无符号、有符号还是混合符号的。仅 dsPIC33E 器件支持混合符号乘法。

指令字数： 1
指令周期数： 1

例 1: MPY.N W4*W5, A, [W8]+=2, W4, [W10]+=2, W5
 ; 执行乘法操作 W4*W5, 对结果求反并存入 ACCA
 ; 取操作数 [W8] 至 W4, 指令执行后将 W8 递增 2
 ; 取操作数 [W10] 至 W5, 指令执行后将 W10 递增 2
 ; CORCON = 0x0001 (整数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | 3023 | W4 | 0054 |
| W5 | 1290 | W5 | 660A |
| W8 | 0B00 | W8 | 0B02 |
| W10 | 2000 | W10 | 2002 |
| ACCA | 00 0000 2387 | ACCA | FF FC82 7650 |
| 数据单元 0B00 | 0054 | 数据单元 0B00 | 0054 |
| 数据单元 2000 | 660A | 数据单元 2000 | 660A |
| CORCON | 0001 | CORCON | 0001 |
| SR | 0000 | SR | 0000 |

例 2: MPY.N W4*W5, A, [W8]+=2, W4, [W10]+=2, W5
 ; 执行乘法操作 W4*W5, 对结果求反并存入 ACCA
 ; 取操作数 [W8] 至 W4, 指令执行后将 W8 递增 2
 ; 取操作数 [W10] 至 W5, 指令执行后将 W10 递增 2
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | 3023 | W4 | 0054 |
| W5 | 1290 | W5 | 660A |
| W8 | 0B00 | W8 | 0B02 |
| W10 | 2000 | W10 | 2002 |
| ACCA | 00 0000 2387 | ACCA | FF F904 ECA0 |
| 数据单元 0B00 | 0054 | 数据单元 0B00 | 0054 |
| 数据单元 2000 | 660A | 数据单元 2000 | 660A |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

MSC

相乘并从累加器中减去乘积

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } MSC Wm * Wn, Acc {,[Wx], Wxd} {,[Wy], Wyd} {,AWB}
 {,[Wx] += kx, Wxd} {,[Wy] += ky, Wyd}
 {,[Wx] -= kx, Wxd} {,[Wy] -= ky, Wyd}
 {,[W9 + W12], Wxd} {,[W11 + W12],Wyd}

操作数: Wm * Wn ∈ [W4 * W5, W4 * W6, W4 * W7, W5 * W6, W5 * W7, W6 * W7]
 Acc ∈ [A,B]
 Wx ∈ [W8, W9] ; kx ∈ [-6, -4, -2, 2, 4, 6] ; Wxd ∈ [W4 ... W7]
 Wy ∈ [W10, W11] ; ky ∈ [-6, -4, -2, 2, 4, 6] ; Wyd ∈ [W4 ... W7]
 AWB ∈ [W13, [W13] + = 2]

操作: (Acc(A 或 B)) – (Wm) * (Wn) → Acc(A 或 B)
 ([Wx]) → Wxd ; (Wx) + kx → Wx
 ([Wy]) → Wyd ; (Wy) + ky → Wy
 舍入后的 (Acc(B 或 A)) → AWB

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 0mmm | A1xx | yyii | iijj | jjaa |
|-------|------|------|------|------|------|------|

描述: 将两个工作寄存器的内容相乘，可选操作包括预取操作数以为另一条 MAC 类指令作准备以及保存未指定累加器的结果。有符号乘法操作的 32 位结果将符号扩展为 40 位并从指定的累加器中减去该结果。

如第 4.14.1 节“MAC 预取”中所介绍，操作数 Wx、Wxd、Wy 和 Wyd 用于指定可选的预取操作，上述操作数支持间接寻址和寄存器偏移量寻址。如第 4.14.4 节“MAC 回写”中所介绍，操作数 AWB 用于指定可选操作：存储“另一个”累加器的内容。

m 位用于选择乘法操作的操作数寄存器 Wm 和 Wn。
A 位用于选择存放结果的累加器。
x 位用于选择预取 Wxd 的目标寄存器。
y 位用于选择预取 Wyd 的目标寄存器。
i 位用于选择 Wx 预取操作。
j 位用于选择 Wy 预取操作。
a 位用于选择累加器回写操作的目标寄存器。

注: IF 位（CORCON<0>）用于确定乘法操作采用小数形式还是整数形式。

指令字数: 1
指令周期数: 1

例 1: MSC W6*W7, A, [W8]-=4, W6, [W10]-=4, W7
 ; 执行乘法操作 W6*W7 并从 ACCA 中减去该结果
 ; 取操作数 [W8] 至 W6, 指令执行后将 W8 递减 4
 ; 取操作数 [W10] 至 W7, 指令执行后将 W10 递减 4
 ; CORCON = 0x0001 (整数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W6 | 9051 | W6 | D309 |
| W7 | 7230 | W7 | 100B |
| W8 | 0C00 | W8 | 0BFC |
| W10 | 1C00 | W10 | 1BFC |
| ACCA | 00 0567 8000 | ACCA | 00 3738 5ED0 |
| 数据单元 0C00 | D309 | 数据单元 0C00 | D309 |
| 数据单元 1C00 | 100B | 数据单元 1C00 | 100B |
| CORCON | 0001 | CORCON | 0001 |
| SR | 0000 | SR | 0000 |

例 2: MSC W4*W5, B, [W11+W12], W5, W13
 ; 执行乘法操作 W4*W5 并从 ACCB 中减去该结果
 ; 取操作数 [W11+W12] 至 W5
 ; 回写 ACCA 至 W13
 ; CORCON = 0x0000 (小数乘法, 饱和禁止)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W4 | 0500 | W4 | 0500 |
| W5 | 2000 | W5 | 3579 |
| W11 | 1800 | W11 | 1800 |
| W12 | 0800 | W12 | 0800 |
| W13 | 6233 | W13 | 3738 |
| ACCA | 00 3738 5ED0 | ACCA | 00 3738 5ED0 |
| ACCB | 00 1000 0000 | ACCB | 00 0EC0 0000 |
| 数据单元 2000 | 3579 | 数据单元 2000 | 3579 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

MUL

f 与 WREG 无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } MUL{.B} f

操作数: $f \in [0 \dots 8191]$

操作: 对于字节操作:
 $(WREG) \langle 7:0 \rangle * (f) \langle 7:0 \rangle \rightarrow W2$
对于字操作:
 $(WREG) * (f) \rightarrow W2:W3$

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1100 | 0B0f | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将默认工作寄存器 WREG 的内容与指定文件寄存器的内容相乘，并将结果存放在 W2:W3 寄存器对中。操作数和结果均被解释为无符号整数。如果该指令在字节模式下执行，16 位结果将存入 W2。在字模式下，32 位结果的高位字将存入 W3 而低位字则存入 W2。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。
- 3: IF 位（CORCON<0>）对该操作没有影响。
- 4: 该指令是提供 8 位乘法的唯一指令。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.B 0x800 ; 执行无符号整数乘法操作 (0x800)*WREG (字节模式)

| | 指令执行前 | | 指令执行后 |
|-----------|-------|-----------|-------|
| WREG (W0) | 9823 | WREG (W0) | 9823 |
| W2 | FFFF | W2 | 13B0 |
| W3 | FFFF | W3 | FFFF |
| 数据单元 0800 | 2690 | 数据单元 0800 | 2690 |
| SR | 0000 | SR | 0000 |

例 2: MUL TMR1 ; 执行无符号整数乘法操作 (TMR1)*WREG (字模式)

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| WREG (W0) | F001 | WREG (W0) | F001 |
| W2 | 0000 | W2 | C287 |
| W3 | 0000 | W3 | 2F5E |
| TMR1 | 3287 | TMR1 | 3287 |
| SR | 0000 | SR | 0000 |

MUL.SS

16x16 位有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.SS Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 有符号 (Ws) → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 1www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。源操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.SS W0, W1, W12 ; 执行乘法操作 W0*W1
 ; 结果存入 W12:W13

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 9823 | W0 | 9823 |
| W1 | 67DC | W1 | 67DC |
| W12 | FFFF | W12 | D314 |
| W13 | FFFF | W13 | D5DC |
| SR | 0000 | SR | 0000 |

例 2: MUL.SS W2, [--W4], W0 ; 指令执行前，递减 W4
 ; 执行乘法操作 W2*[W4]
 ; 结果存入 W0:W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | FFFF | W0 | 28F8 |
| W1 | FFFF | W1 | 0000 |
| W2 | 0045 | W2 | 0045 |
| W4 | 27FE | W4 | 27FC |
| 数据单元 27FC | 0098 | 数据单元 27FC | 0098 |
| SR | 0000 | SR | 0000 |

MUL.SS

使用累加器作为目标寄存器的 16x16 位有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 :} MUL.SS Wb, Ws, [Ws], A [Ws++], B [Ws--], [++Ws], [--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
ACC ∈ [A, B]

操作: 有符号 (Wb) * 有符号 (Ws) → ACC(A 或 B)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 1www | w111 | Appp | ssss |
|-------|------|------|------|------|------|------|

描述: 执行 16 位 x 16 位有符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前，32 位结果会进行符号扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。两个源操作数都视为有符号值。

- w 位用于选择基准寄存器的地址。
- d 位用于选择源寄存器的地址。
- p 位用于选择源地址模式。
- A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.SS W0, W1, A

| | 指令执行前 | | 指令执行后 |
|-------|--------------|-------|--------------|
| W0 | 9823 | W0 | 9823 |
| W1 | 67DC | W1 | 67DC |
| Acc A | 00 0000 0000 | Acc A | FF D5DC D314 |
| SR | 0000 | SR | 0000 |

MUL.SU16x16 位有符号 - 无符号短立即数整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.SU Wb, #lit5, Wnd

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 无符号 lit5 → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | wddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 5 位立即数相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。Wb 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。立即数将被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
k 位用于定义 5 位无符号整数立即数。

- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1

例 1: MUL.SU W0, #0x1F, W2 ; W0 内容与立即数 0x1F 相乘
; 结果存入 W2:W3

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | C000 | W0 | C000 |
| W2 | 1234 | W2 | 4000 |
| W3 | C9BA | W3 | FFF8 |
| SR | 0000 | SR | 0000 |

例 2: MUL.SU W2, #0x10, W0 ; W2 内容与立即数 0x10 相乘
; 结果存入 W0:W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | ABCD | W0 | 2400 |
| W1 | 89B3 | W1 | 000F |
| W2 | F240 | W2 | F240 |
| SR | 0000 | SR | 0000 |

MUL.SU

16x16 位有符号 - 无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.SU Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 无符号 (Ws) → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。Wb 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。Ws 操作数将被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.SU W8, [W9], W0 ; 执行乘法操作 W8*[W9]
; 结果存入 W0:W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 68DC | W0 | 0000 |
| W1 | AA40 | W1 | F100 |
| W8 | F000 | W8 | F000 |
| W9 | 178C | W9 | 178C |
| 数据单元 178C | F000 | 数据单元 178C | F000 |
| SR | 0000 | SR | 0000 |

例 2: MUL.SU W2, [++W3], W4 ; 指令执行前, 递增 W3
; 执行乘法操作 W2*[W3]
; 结果存入 W4:W5

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W2 | 0040 | W2 | 0040 |
| W3 | 0280 | W3 | 0282 |
| W4 | 1819 | W4 | 1A00 |
| W5 | 2021 | W5 | 0000 |
| 数据单元 0282 | 0068 | 数据单元 0282 | 0068 |
| SR | 0000 | SR | 0000 |

MUL.SU

使用累加器作为目标寄存器的16x16位有符号-无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法:

{ 标号 :} MUL.SU Wb, Ws, A
 [Ws], B
 [Ws++], B
 [Ws--],
 [++Ws],
 [--Ws],

操作数:

Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
ACC ∈ [A, B]

操作:

有符号 (Wb) * 无符号 (Ws) → ACC(A 或 B)

受影响的状态位:

无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | w111 | Appp | ssss |
|-------|------|------|------|------|------|------|

描述:

执行 16 位 x 16 位有符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前， 32 位结果会进行符号扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。第一个源操作数会被解释为二进制补码有符号值，第二个源操作数会被解释为无符号值。

w 位用于选择基准寄存器的地址。

d 位用于选择源寄存器的地址。

p 位用于选择源地址模式。

A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数:

1

指令周期数:

1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.SU W8, W9, A

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W8 | F000 | W8 | F000 |
| W9 | F000 | W9 | F000 |
| Acc A | 00 0000 0000 | Acc A | FF F100 0000 |
| SR | 0000 | SR | 0000 |

MUL.SU

使用累加器作为目标寄存器的 16x16 位有符号 - 无符号短立即数整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 :} MUL.SU Wb, #lit5, A
B

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
ACC ∈ [A, B]

操作: 有符号 (Wb) * 无符号 (lit5) → ACC(A 或 B)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | w111 | A11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 执行 16 位 x 16 位有符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前，32 位结果会进行符号扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。第一个源操作数会被解释为二进制补码有符号值，第二个源操作数会被解释为无符号值。

w 位用于选择基准寄存器的地址。
k 位用于选择 5 位立即数值。
A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数: 1
指令周期数: 1

例 1: MUL.SU W8, #0x02, A

| 指令执行前 | | | 指令执行后 | | |
|-------|--------------|--|-------|--------------|--|
| W8 | 0042 | | W8 | 0042 | |
| Acc A | 00 0000 0000 | | Acc A | 00 0000 0084 | |
| SR | 0000 | | SR | 0000 | |

MUL.US

16x16 位无符号 - 有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.US Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 无符号 (Wb) * 有符号 (Ws) → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 1www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。Wb 操作数将被解释为无符号整数。Ws 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.US W0, [W1], W2 ; 执行乘法操作 W0*[W1]（无符号 - 有符号）
; 结果存入 W2:W3

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | C000 | W0 | C000 |
| W1 | 2300 | W1 | 2300 |
| W2 | 00DA | W2 | 0000 |
| W3 | CC25 | W3 | F400 |
| 数据单元 2300 | F000 | 数据单元 2300 | F000 |
| SR | 0000 | SR | 0000 |

例 2: MUL.US W6, [W5++], W10 ; 执行乘法操作 W6*[W5]（无符号 - 有符号）
; 结果存入 W10:W11
; 指令执行后，递增 W5

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W5 | 0C00 | W5 | 0C02 |
| W6 | FFFF | W6 | FFFF |
| W10 | 0908 | W10 | 8001 |
| W11 | 6EEB | W11 | 7FFE |
| 数据单元 0C00 | 7FFF | 数据单元 0C00 | 7FFF |
| SR | 0000 | SR | 0000 |

MUL.US

使用累加器作为目标寄存器的16x16位无符号-有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 :} MUL.US Wb, Ws, A
[Ws], B
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
ACC ∈ [A, B]

操作: 无符号 (Wb) * 有符号 (Ws) → ACC(A 或 B)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | w111 | Appp | ssss |
|-------|------|------|------|------|------|------|

描述: 执行 16 位 x 16 位有符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前，32 位结果会进行符号扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。第一个源操作数会被解释为无符号值，第二个源操作数会被解释为二进制补码有符号值。

w 位用于选择基准寄存器的地址。
d 位用于选择源寄存器的地址。
p 位用于选择源地址模式。
A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.US W0, W1, B

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W0 | C000 | W0 | 0000 |
| W1 | F000 | W1 | F000 |
| Acc B | 00 0000 0000 | Acc B | FF F400 0000 |
| SR | 0000 | SR | 0000 |

MUL.UU

16x16 位无符号短立即数整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.UU Wb, #lit5, Wnd

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 无符号 (Wb) * 无符号 lit5 → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | wddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 5 位立即数相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。操作数和结果均被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
k 位用于定义 5 位无符号整数立即数。

- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1

指令周期数: 1

例 1: MUL.UU W0, #0xF, W12 ; W0 内容与立即数 0xF 相乘
; 结果存入 W12:W13

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 2323 | W0 | 2323 |
| W12 | 4512 | W12 | 0F0D |
| W13 | 7821 | W13 | 0002 |
| SR | 0000 | SR | 0000 |

例 2: MUL.UU W7, #0x1F, W0 ; W7 内容与立即数 0x1F 相乘
; 结果存入 W0:W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 780B | W0 | 55C0 |
| W1 | 3805 | W1 | 001D |
| W7 | F240 | W7 | F240 |
| SR | 0000 | SR | 0000 |

MUL.UU16x16 位无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} MUL.UU Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 无符号 (Wb) * 无符号 (Ws) → Wnd:Wnd + 1

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将 32 位结果存入两个连续的工作寄存器。结果的低位字将存入 Wnd（必须为偶数编号的工作寄存器），而结果的高位字将存入 Wnd + 1。源操作数和结果均被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或间接寻址对 Ws 进行寻址。

- w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。
- 注 1: 该指令只能工作于字模式。
2: 由于乘积为 32 位，因此 Wnd 必须是偶数编号的工作寄存器。有关在存储区中双字如何对齐的信息，请参见图 4-3。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.UU W4, W0, W2 ; 执行乘法操作 W4*W0 (无符号 - 无符号)
; 结果存入 W2:W3

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | FFFF | W0 | FFFF |
| W2 | 2300 | W2 | 0001 |
| W3 | 00DA | W3 | FFFE |
| W4 | FFFF | W4 | FFFF |
| SR | 0000 | SR | 0000 |

例 2: MUL.UU W0, [W1++], W4 ; 执行乘法操作 W0*[W1] (无符号 - 无符号)
; 结果存入 W4:W5
; 指令执行后, 递增 W1

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | 1024 | W0 | 1024 |
| W1 | 2300 | W1 | 2302 |
| W4 | 9654 | W4 | 6D34 |
| W5 | BDBC | W5 | 0D80 |
| 数据单元 2300 | D625 | 数据单元 2300 | D625 |
| SR | 0000 | SR | 0000 |

MUL.UU 使用累加器作为目标寄存器的 16x16 位无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 :} MUL.UU Wb, Ws, A
[Ws], B
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
ACC ∈ [A, B]

操作: 无符号 (Wb) * 无符号 (Ws) → ACC(A 或 B)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | w111 | Appp | ssss |
|-------|------|------|------|------|------|------|

描述: 执行 16 位 x 16 位无符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前，32 位结果会进行零扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。两个源操作数都视为无符号值。

w 位用于选择基准寄存器的地址。
d 位用于选择源寄存器的地址。
p 位用于选择源地址模式。
A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MUL.UU W4, W0, B

| 指令执行前 | | | | 指令执行后 | | | |
|-------|--|--------------|--|-------|--|--------------|--|
| W0 | | FFFF | | W0 | | FFFF | |
| W4 | | FFFF | | W4 | | FFFF | |
| Acc B | | 00 0000 0000 | | Acc B | | FF FFFE 0001 | |
| SR | | 0000 | | SR | | 0000 | |

MUL.UU

使用累加器作为目标寄存器的 16x16 位无符号短立即数整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | | | X |

语法: { 标号 :} MUL.UU Wb, #lit5, A
B

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
ACC ∈ [A, B]

操作: 无符号 (Wb) * 无符号 (lit5) → ACC(A 或 B)

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | w111 | A11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 执行 16 位 x 16 位有符号乘法，产生一个 32 位结果，结果存入 DSP 引擎累加器 ACCA 或 ACCB。在装入目标累加器之前，32 位结果会进行零扩展，扩展到 bit 39。

源操作数会根据 DSP 引擎的工作模式（由 CORCON<0> 中的 IF 位定义）而视为整数或小数值。两个源操作数都视为无符号值。

w 位用于选择基准寄存器的地址。
k 位用于选择 5 位立即数。
A 位用于选择存放乘积的目标累加器。

- 注 1: 该指令只能工作于字模式。
- 2: 乘法器模式位（CORCON 中的 US<1:0>）的状态对该指令的操作不会产生任何影响。

指令字数: 1
指令周期数: 1

例 1: MUL.UU W8, #0x02, A

| | 指令执行前 | | 指令执行后 |
|-------|--------------|-------|--------------|
| W8 | 0042 | W8 | 0042 |
| Acc A | 00 0000 0000 | Acc A | 00 0000 0084 |
| SR | 0000 | SR | 0000 |

MULW.SS 产生 16 位结果的 16x16 位有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} MULW.SS Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 有符号 (Ws) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 1www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将结果存入工作寄存器（必须是偶数编号的工作寄存器）。源操作数和 Wnd 中的结果均解释为二进制补码有符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。
w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: Wnd 必须是偶数编号的工作寄存器。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MULW.SS W0, W1, W12 ; 执行乘法操作 W0*W1
; 结果存入 W12

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 9823 | W0 | 9823 |
| W1 | 67DC | W1 | 67DC |
| W12 | FFFF | W12 | D314 |
| SR | 0000 | SR | 0000 |

例 2: MULW.SS W2, [--W4], W0 ; 指令执行前, 递减 W4
; 执行乘法操作 W2*[W4]
; 结果存入 W0

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | FFFF | W0 | 28F8 |
| W2 | 0045 | W2 | 0045 |
| W4 | 27FE | W4 | 27FC |
| 数据单元 27FC | 0098 | 数据单元 27FC | 0098 |
| SR | 0000 | SR | 0000 |

MULW.SU 产生 16 位结果的 16x16 位有符号 - 无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} MUL.SU Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 无符号 (Ws) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将结果存入工作寄存器（必须是偶数编号的工作寄存器）。Wb 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。Ws 操作数将被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: Wnd 必须是偶数编号的工作寄存器。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MULW.SU W8, [W9], W0 ; 执行乘法操作 W8*[W9]
; 结果存入 W0

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-----------|------|-----------|------|
| W0 | 68DC | W0 | 0000 | W0 | 0000 | W0 | 0000 |
| W8 | F000 | W8 | F000 | W8 | F000 | W8 | F000 |
| W9 | 178C | W9 | 178C | W9 | 178C | W9 | 178C |
| 数据单元 178C | F000 | 数据单元 178C | F000 | 数据单元 178C | F000 | 数据单元 178C | F000 |
| SR | 0000 | SR | 0000 | SR | 0000 | SR | 0000 |

例 2:

MULW.SU W2, [++W3], W4 ; 指令执行前, 递增 W3
; 执行乘法操作 W2*[W3]
; 结果存入 W4

指令执行前

| | |
|------|------|
| W2 | 0040 |
| W3 | 0280 |
| W4 | 1819 |
| 0282 | 0068 |
| SR | 0000 |

数据单元 0282

SR

指令执行后

| | |
|------|------|
| W2 | 0040 |
| W3 | 0282 |
| W4 | 1A00 |
| 0282 | 0068 |
| SR | 0000 |

数据单元 0282

SR

MULW.SU 产生 16 位结果的 16x16 位有符号 - 无符号短立即数整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} MULW.SU Wb, #lit5, Wnd

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 有符号 (Wb) * 无符号 (lit5) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1001 | 0www | wddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 5 位立即数值相乘，并将结果存入工作寄存器（必须是偶数编号的工作寄存器）。Wb 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
k 位用于选择 5 位立即数值。

- 注 1: 该指令只能工作于字模式。
2: Wnd 必须是偶数编号的工作寄存器。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1

例 1: MULW.SU W8, #0x04, W0 ; 执行乘法操作 W8 * #0x04
; 结果存入 W0

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 68DC | W0 | 4000 |
| W8 | 1000 | W8 | 1000 |
| SR | 0000 | SR | 0000 |

MULW.US

产生 16 位结果的 16x16 位无符号 - 有符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} MULW.US Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 无符号 (Wb) * 有符号 (Ws) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 1www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将结果存入工作寄存器（必须是偶数编号的工作寄存器）。Wb 操作数将被解释为无符号整数。Ws 操作数和 Wnd 中的结果均被解释为二进制补码有符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或寄存器间接寻址对 Ws 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: Wnd 必须是偶数编号的工作寄存器。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MULW.US W0, [W1], W2 ; 执行乘法操作 W0*[W1]（无符号 - 有符号）
; 结果存入 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W0 | C000 | W0 | C000 |
| W1 | 2300 | W1 | 2300 |
| W2 | 00DA | W2 | 0000 |
| 数据单元 2300 | F000 | 数据单元 2300 | F000 |
| SR | 0000 | SR | 0000 |

例 2: MULW.US W6, [W5++], W10 ; 执行乘法操作 W6*[W5] (无符号 - 有符号)
 ; 结果存入 W10
 ; 指令执行后, 递增 W5

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W5 | 0C00 | W5 | 0C02 |
| W6 | FFFF | W6 | FFFF |
| W10 | 0908 | W10 | 8001 |
| 数据单元 0C00 | 7FFF | 数据单元 0C00 | 7FFF |
| SR | 0000 | SR | 0000 |

MULW.UU 产生 16 位结果的 16x16 位无符号整数乘法

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} MULW.UU Wb, Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wnd ∈ [W0, W2, W4 ... W12]

操作: 无符号 (Wb) * 无符号 (Ws) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1000 | 0www | wddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Wb 内容与 Ws 内容相乘，并将结果存入工作寄存器（必须是偶数编号的工作寄存器）。源操作数和结果均被解释为无符号整数。必须使用寄存器直接寻址对 Wb 和 Wnd 进行寻址。可使用寄存器直接或间接寻址对 Ws 进行寻址。
w 位用于选择基准寄存器的地址。
d 位用于选择低位字目标寄存器的地址。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该指令只能工作于字模式。
2: Wnd 必须是偶数编号的工作寄存器。
3: 由于 W15<0> 固定为零，因此 Wnd 不能为 W14。
4: CORCON 寄存器中的 IF 位和 US<1:0> 位对该操作没有影响。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: MULW.UU W4, W0, W2 ; 执行乘法操作 W4*W0（无符号 - 无符号）
; 结果存入 W2

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | FFFF | W0 | FFFF |
| W2 | 2300 | W2 | 0001 |
| W4 | FFFF | W4 | FFFF |
| SR | 0000 | SR | 0000 |

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W2 | 2300 | W2 | 4000 |
| W4 | 1000 | W4 | 1000 |
| SR | 0000 | SR | 0000 |

NEG

对 f 内容求补

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} NEG{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $\overline{(f)} + 1 \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1110 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 计算文件寄存器内容的二进制补码，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: NEG.B 0x880, WREG ; 对 (0x880) 求补（字节模式）
; 结果存入 WREG

| | | | |
|-----------|-------|-----------|--------------|
| | 指令执行前 | | 指令执行后 |
| WREG (W0) | 9080 | WREG (W0) | 90AB |
| 数据单元 0880 | 2355 | 数据单元 0880 | 2355 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: NEG 0x1200 ; 对 (0x1200) 求补（字模式）

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 1200 | 8923 | 数据单元 1200 | 76DD |
| SR | 0000 | SR | 0000 |

NEG对 Ws 内容求补

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} NEG{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) + 1 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1010 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 计算源寄存器 Ws 内容的二进制补码，并将结果存放在目标寄存器 Wd 中。
可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: NEG.B W3, [W4++] ; 对 W3 内容求补，并将结果存入 [W4]（字节模式）
; 指令执行后，递增 W4

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-------|------|---------|--|
| W3 | 7839 | | | W3 | 7839 | | |
| W4 | 1005 | | | W4 | 1006 | | |
| 数据单元 1004 | 2355 | 数据单元 1004 | C755 | | | | |
| SR | 0000 | SR | 0008 | | | (N = 1) | |

例 2: NEG [W2++], [--W4] ; 指令执行前，递减 W4（字模式）
; 对 [W2] 内容求补，并将结果存入 [W4]
; 指令执行后，递增 W2

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-------|--|--|--|
| W2 | 0900 | W2 | 0902 | | | | |
| W4 | 1002 | W4 | 1000 | | | | |
| 数据单元 0900 | 870F | 数据单元 0900 | 870F | | | | |
| 数据单元 1000 | 5105 | 数据单元 1000 | 78F1 | | | | |
| SR | 0000 | SR | 0000 | | | | |

NEG

对累加器内容求补

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} NEG Acc

操作数: Acc ∈ [A,B]

操作: 如果 (Acc = A):
 -ACCA → ACCA
 否则:
 -ACCB → ACCB

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1011 | A001 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 计算指定累加器内容的二进制补码。无论采用何种饱和模式，该指令可对累加器的所有 40 位进行操作。

 A 位用于指定选择的累加器。

指令字数: 1

指令周期数: 1

例 1: NEG A ; 对 ACCA 内容求补
 ; 结果存入 ACCA
 ; CORCON = 0x0000 (饱和禁止)

| | | | |
|--------|--------------|--------|--------------|
| | 指令执行前 | | 指令执行后 |
| ACCA | 00 3290 59C8 | ACCA | FF CD6F A638 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0000 |

例 2: NEG B ; 对 ACCB 内容求补
 ; 结果存入 ACCB
 ; CORCON = 0x00C0 (普通饱和模式)

| | | | |
|--------|--------------|--------|--------------|
| | 指令执行前 | | 指令执行后 |
| ACCB | FF F230 10DC | ACCB | 00 0DCF EF24 |
| CORCON | 00C0 | CORCON | 00C0 |
| SR | 0000 | SR | 0000 |

NOP

空操作

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } NOP

操作数: 无

操作: 空操作

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0000 | xxxx | xxxx | xxxx | xxxx |
|-------|------|------|------|------|------|------|

描述: 不执行任何操作。

x 位可以是任何值。

指令字数: 1

指令周期数: 1

例 1: NOP ; 不执行任何操作

| | |
|-------|---------|
| 指令执行前 | |
| PC | 00 1092 |
| SR | 0000 |

| | |
|-------|---------|
| 指令执行后 | |
| PC | 00 1094 |
| SR | 0000 |

例 2: NOP ; 不执行任何操作

| | |
|-------|---------|
| 指令执行前 | |
| PC | 00 08AE |
| SR | 0000 |

| | |
|-------|---------|
| 指令执行后 | |
| PC | 00 08B0 |
| SR | 0000 |

NOPR

空操作

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } NOPR

操作数: 无

操作: 空操作

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1111 | xxxx | xxxx | xxxx | xxxx |
|-------|------|------|------|------|------|------|

描述: 不执行任何操作。

x 位可以是任何值。

指令字数: 1

指令周期数: 1

例 1: NOPR ; 不执行任何操作

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 2430 | PC | 00 2432 |
| SR | 0000 | SR | 0000 |

例 2: NOPR ; 不执行任何操作

| 指令执行前 | | 指令执行后 | |
|-------|---------|-------|---------|
| PC | 00 1466 | PC | 00 1468 |
| SR | 0000 | SR | 0000 |

POP 栈顶内容弹出至 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } POP f

操作数: f ∈ [0 ... 65534]
操作: (W15) – 2 → W15
(TOS) → f

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1001 | ffff | ffff | ffff | fff0 |
|-------|------|------|------|------|------|------|

描述: 首先将堆栈指针（W15）递减 2，随后栈顶（TOS）字内容将被写入指定的文件寄存器，该寄存器可位于低 32K 字数据存储区中的任何地址。

f 位用于选择文件寄存器的地址。

- 注 1: 该指令只能工作于字模式。
2: 文件寄存器地址必须符合字对齐原则。

指令字数: 1
指令周期数: 1

例 1: POP 0x1230 ; TOS 弹出至 0x1230

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W15 | 1006 | W15 | 1004 |
| 数据单元 1004 | A401 | 数据单元 1004 | A401 |
| 数据单元 1230 | 2355 | 数据单元 1230 | A401 |
| SR | 0000 | SR | 0000 |

例 2: POP 0x880 ; TOS 弹出至 0x880

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|-----|------|-----------|--|-----|------|
| | | W15 | 2000 | | | W15 | 1FFE |
| 数据单元 0880 | | | E3E1 | 数据单元 0880 | | | A090 |
| 数据单元 1FFE | | | A090 | 数据单元 1FFE | | | A090 |
| SR | | | 0000 | SR | | | 0000 |

POP

栈顶内容弹出至 Wd

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } POP Wd
 [Wd]
 [Wd++]
 [Wd--]
 [--Wd]
 [++Wd]
 [Wd+Wb]

操作数: Wd ∈ [W0 ... W15]
 Wb ∈ [W0 ... W15]

操作: (W15) – 2 → W15
 (TOS) → Wd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0111 | 1www | w0hh | hddd | d100 | 1111 |
|-------|------|------|------|------|------|------|

描述: 首先将堆栈指针 (W15) 递减 2, 随后栈顶 (TOS) 字内容将被写入 Wd。
 可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于定义偏移量寄存器 Wb。
h 位用于选择目标地址模式。
d 位用于选择目标寄存器。

- 注 1: 该指令只能工作于字模式。
 2: 该指令是 “MOV ws, wd” 指令 (MOV [--W15], wd) 的特殊版本。它反汇编为 MOV。

指令字数: 1
指令周期数: 1

例 1: POP W4 ; TOS 弹出至 W4

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|----|------|-----------|--|----|------|
| | | W4 | EDA8 | | | W4 | C45A |
| W15 | | | 1008 | W15 | | | 1006 |
| 数据单元 1006 | | | C45A | 数据单元 1006 | | | C45A |
| SR | | | 0000 | SR | | | 0000 |

例 2: POP [++W10] ; 指令执行前, 递增 W10
; TOS 弹出至 [W10]

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W10 | 0E02 | W10 | 0E04 |
| W15 | 1766 | W15 | 1764 |
| 数据单元 0E04 | E3E1 | 数据单元 0E04 | C7B5 |
| 数据单元 1764 | C7B5 | 数据单元 1764 | C7B5 |
| SR | 0000 | SR | 0000 |

POP.D 将栈顶内容双字弹出至 Wnd:Wnd+1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } POP.D Wnd

操作数: Wnd ∈ [W0, W2, W4, ... W14]

操作: (W15) - 2 → W15
(TOS) → Wnd + 1
(W15) - 2 → W15
(TOS) → Wnd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1110 | 0000 | 0ddd | 0100 | 1111 |
|-------|------|------|------|------|------|------|

描述: 从栈顶 (TOS) 弹出双字并存入 Wnd:Wnd + 1。高位字存入 Wnd + 1，而低位字则存入 Wnd。由于弹出内容为双字，堆栈指针 (W15) 将递减 4。
d 位用于选择目标寄存器对的地址。

- 注 1: 该指令对双字进行操作。有关在存储区中双字如何对齐的信息，请参见图 4-3。
2: Wnd 必须是偶数编号的工作寄存器。
3: 该指令是 “MOV.D Ws, Wnd” 指令 (MOV.D [--W15], Wnd) 的特殊版本。它反汇编为 MOV.D。

指令字数: 1
指令周期数: 2

例 1: POP.D W6 ; 双字弹出 TOS 至 W6

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W6 | 07BB | W6 | 3210 |
| W7 | 89AE | W7 | 7654 |
| W15 | 0850 | W15 | 084C |
| 数据单元 084C | 3210 | 数据单元 084C | 3210 |
| 数据单元 084E | 7654 | 数据单元 084E | 7654 |
| SR | 0000 | SR | 0000 |

例 1: POP.S ; 弹出影子寄存器内容
; (有关影子寄存器的内容, 请参见 PUSH.S 的例 1)

| 指令执行前 | | 指令执行后 | |
|-------|----------------|-------|-----------------------|
| W0 | 07BB | W0 | 0000 |
| W1 | 03FD | W1 | 1000 |
| W2 | 9610 | W2 | 2000 |
| W3 | 7249 | W3 | 3000 |
| SR | 00E0 (IPL = 7) | SR | 00E1 (IPL = 7, C = 1) |

弹出影子寄存器内容

| | | | | | |
|------|------|------|------|------|------|
| 1111 | 1110 | 1000 | 0000 | 0000 | 0000 |
|------|------|------|------|------|------|

注 1: 不能直接访问影子寄存器。只可使用 `PUSH.S` 和 `POP.S` 对其进行访问。

2: 影子寄存器仅为一级深度。

指令周期数: 1

注：在指令执行后，影子寄存器中的内容不改变。

PUSH

将 f 内容压入栈顶

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } PUSH f

操作数： f ∈ [0 ... 65534]

操作： (f) → (TOS)
 (W15) + 2 → W15

受影响的状态位： 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1111 | 1000 | ffff | ffff | ffff | fff0 |
|-------|------|------|------|------|------|------|

描述： 将指定文件寄存器中内容写入栈顶（TOS）地址单元，随后堆栈指针（W15）递增 2。文件寄存器可位于低 32K 字数据存储区中的任何地址。

 f 位用于选择文件寄存器的地址。

- 注 1:
- 该指令只能工作于字模式。
- 2:
- 文件寄存器地址必须符合字对齐原则。

指令字数： 1

指令周期数： 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: PUSH 0x2004 ; (0x2004) 压入 TOS

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-----------|------|--|--|
| W15 | 0B00 | W15 | 0B02 | W15 | 0B02 | | |
| 数据单元 0B00 | 791C | 数据单元 0B00 | D400 | 数据单元 0B00 | D400 | | |
| 数据单元 2004 | D400 | 数据单元 2004 | D400 | 数据单元 2004 | D400 | | |
| SR | 0000 | SR | 0000 | SR | 0000 | | |

例 2: PUSH 0xC0E ; (0xC0E) 压入 TOS

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|------|-----------|------|-----------|------|--|--|
| W15 | 0920 | W15 | 0922 | W15 | 0922 | | |
| 数据单元 0920 | 0000 | 数据单元 0920 | 67AA | 数据单元 0920 | 67AA | | |
| 数据单元 0C0E | 67AA | 数据单元 2004 | 67AA | 数据单元 2004 | 67AA | | |
| SR | 0000 | SR | 0000 | SR | 0000 | | |

PUSH将 Ws 内容压入栈顶

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} PUSH Ws
[Ws]
[Ws++]
[Ws--]
[--Ws]
[++Ws]
[Ws+Wb]

操作数: Ws ∈ [W0 ... W15]
Wb ∈ [W0 ... W15]

操作: (Ws) → (TOS)
(W15) + 2 → W15

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0111 | 1www | w001 | 1111 | 1ggg | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Ws 内容写入栈顶（TOS）地址单元，随后堆栈指针（W15）递增 2。

w 位用于定义偏移量寄存器 Wb。
g 位用于选择源地址模式。
s 位用于选择源寄存器。

注 1: 该指令只能工作于字模式。
2: 该指令是“MOV Ws, Wd”指令（MOV Ws, [W15++]）的特殊版本。它反汇编为 MOV。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: PUSH W2 ; W2 内容压入 TOS

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|-----------|-------|--|------|--|
| W2 | | 6889 | | W2 | | 6889 | |
| W15 | | 1566 | | W15 | | 1568 | |
| 数据单元 1566 | | 0000 | 数据单元 1566 | | | 6889 | |
| SR | | 0000 | SR | | | 0000 | |

例 2: PUSH [W5+W10] ; [W5+W10] 内容压入 TOS

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W5 | 1200 | W5 | 1200 |
| W10 | 0044 | W10 | 0044 |
| W15 | 0806 | W15 | 0808 |
| 数据单元 0806 | 216F | 数据单元 0806 | B20A |
| 数据单元 1244 | B20A | 数据单元 1244 | B20A |
| SR | 0000 | SR | 0000 |

PUSH.D 将 Wns:Wns+1 内容双字压入栈顶

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } PUSH.D Wns

操作数: Wns ∈ [W0, W2, W4 ... W14]

操作: (Wns) → (TOS)
(W15) + 2 → W15
(Wns + 1) → (TOS)
(W15) + 2 → W15

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1110 | 1001 | 1111 | 1000 | sss0 |
|-------|------|------|------|------|------|------|

描述: 将双字 (Wns:Wns + 1) 内容压入栈顶 (TOS)。首先将低位字 (Wns) 压入 TOS, 然后将高位字 (Wns + 1) 压入 TOS。由于执行双字压入操作, 堆栈指针 (W15) 将递增 4。

s 位用于选择源寄存器对的地址。

- 注 1: 该指令对双字进行操作。有关在存储区中双字如何对齐的信息, 请参见图 4-3。
 2: Wns 必须是偶数编号的工作寄存器。
 3: 该指令是“MOV.D Wns, Wd”指令 (MOV.D Wns, [W15++]) 的特殊版本。它反汇编为 MOV.D。

指令字数: 1

指令周期数: 2

例 1: PUSH.D W6 ; W6:W7 内容压入 TOS

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W6 | C451 | W6 | C451 |
| W7 | 3380 | W7 | 3380 |
| W15 | 1240 | W15 | 1244 |
| 数据单元 1240 | B004 | 数据单元 1240 | C451 |
| 数据单元 1242 | 0891 | 数据单元 1242 | 3380 |
| SR | 0000 | SR | 0000 |

例 2: PUSH.D W10 ; W10:W11 内容压入 TOS

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W10 | 80D3 | W10 | 80D3 |
| W11 | 4550 | W11 | 4550 |
| W15 | 0C08 | W15 | 0C0C |
| 数据单元 0C08 | 79B5 | 数据单元 0C08 | 80D3 |
| 数据单元 0C0A | 008E | 数据单元 0C0A | 4550 |
| SR | 0000 | SR | 0000 |

PUSH.S

压入影子寄存器

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } PUSH.S

操作数: 无

操作: 压入影子寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 1010 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 将主寄存器中内容拷贝至各自的影子寄存器。以下寄存器将具有影子寄存器: W0-W3 以及 C、Z、OV、N 和 DC 状态寄存器标志。

- 注 1: 不能直接访问影子寄存器。只可使用 PUSH.S 和 POP.S 对其进行访问。
- 2: 影子寄存器仅为一级深度。

指令字数: 1

指令周期数: 1

例 1: PUSH.S ; 将主寄存器中内容压入影子寄存器

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W0 | 0000 | W0 | 0000 |
| W1 | 1000 | W1 | 1000 |
| W2 | 2000 | W2 | 2000 |
| W3 | 3000 | W3 | 3000 |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

注: 在指令执行后，影子寄存器中的内容将被更新。

PWRSAV 进入低功耗模式

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } PWRSAV #lit1

操作数: lit1 ∈ [0,1]

操作:

0 → WDT 计数寄存器
0 → WDT 预分频器 A 计数
0 → WDT 预分频器 B 计数
0 → WDTO (RCON<4>)
0 → SLEEP (RCON<3>)
0 → IDLE (RCON<2>)
如果 (lit1 = 0):
 则进入休眠模式
否则:
 则进入空闲模式

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 0100 | 0000 | 0000 | 000k |
|-------|------|------|------|------|------|------|

描述:

将处理器置于指定的低功耗模式。如果 lit1 = 0，将进入休眠模式。在休眠模式下，提供给 CPU 及外设的时钟都被关闭。如果使用片内振荡器，它也将被关闭。如果 lit1 = 1，将进入空闲模式。在空闲模式下，CPU 的时钟关闭，但时钟源仍处于工作状态，外设仍将继续工作。

该指令将使看门狗定时器计数寄存器和预分频器计数寄存器复位。此外，复位系统和控制 (RCON) 寄存器的 WDTO、SLEEP 和 IDLE 标志也将被复位。

- 注 1: 中断、处理器复位或看门狗定时器超时将使处理器退出空闲或休眠模式。有关详细信息，请参见具体器件数据手册。
- 2: 如果从空闲模式唤醒，IDLE 位 (RCON<2>) 将被置 1，且时钟源将加到 CPU。
- 3: 如果从休眠模式唤醒，SLEEP 位 (RCON<3>) 将被置 1，且时钟源将起振。
- 4: 如果由于看门狗定时器超时唤醒，WDTO 位 (RCON<4>) 将被置 1。

指令字数: 1

指令周期数: 1

例 1: PWRSAV #0 ; 进入休眠模式

| | |
|-------------------|-------------------|
| 指令执行前 | 指令执行后 |
| SR 0040 (IPL = 2) | SR 0040 (IPL = 2) |

例 2: PWRSAV #1 ; 进入空闲模式

| | |
|-------------------|-------------------|
| 指令执行前 | 指令执行后 |
| SR 0020 (IPL = 1) | SR 0020 (IPL = 1) |

RCALL

相对调用

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } RCALL Expr

操作数: Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... 32767]。

操作: (PC) + 2 → PC
(PC<15:0>) → (TOS)
(W15) + 2 → W15
(PC<22:16>) → (TOS)
(W15) + 2 → W15
(PC) + (2 * Slit16) → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0111 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 自当前 PC 向前或向后的 32K 程序字存储空间范围内进行相对子程序调用。在进行调用之前，返回地址 (PC + 2) 被压入堆栈。在返回地址压入堆栈之后，符号扩展后的 17 位值 (2 * Slit16) 将与 PC 的内容相加且结果存入 PC。

n 位为一个有符号立即数，用于指定相对调用距离 (PC + 2) 的长度（以程序字为单位）。

注: 由于该指令将只占用一个字的程序存储区，因此应尽可能使用该指令而非 CALL 指令。

指令字数: 1
指令周期数: 2

例 1:

| | | | |
|--------|---------|------------|-------------------------|
| 012004 | RCALL | _Task1 | ; 调用 _Task1 |
| 012006 | ADD | W0, W1, W2 | |
| . | ... | | |
| . | ... | | |
| 012458 | _Task1: | SUB | W0, W2, W3 ; _Task1 子程序 |
| 01245A | ... | | |

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 01 2004 | PC | 01 2458 |
| W15 | 0810 | W15 | 0814 |
| 数据单元 0810 | FFFF | 数据单元 0810 | 2006 |
| 数据单元 0812 | FFFF | 数据单元 0812 | 0001 |
| SR | 0000 | SR | 0000 |

例 2:

00620E
006210
.
.
007000
007002

RCALL
MOV
...
...
CLR
...

_Init
W0, [W4++]

W2

; 调用 _Init

; _Init 子程序

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 00 620E | PC | 00 7000 |
| W15 | 0C50 | W15 | 0C54 |
| 数据单元 0C50 | FFFF | 数据单元 0C50 | 6210 |
| 数据单元 0C52 | FFFF | 数据单元 0C52 | 0000 |
| SR | 0000 | SR | 0000 |

DS70157F_CN 第 348 页

© 2011-2016 Microchip Technology Inc.

RCALL

相对调用

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } RCALL Expr

操作数: Expr 可以是绝对地址、标号或表达式。
Expr 由链接器解析为一个 Slit16，其中 Slit16 ∈ [-32768 ... 32767]。

操作: (PC) + 2 → PC
(PC<15:1>) → TOS<15:1>, SFA 位 → TOS<0>
(W15) + 2 → W15
(PC<22:16>) → (TOS)
(W15) + 2 → W15
0 → SFA 位
(PC) + (2 * Slit16) → PC
NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0111 | nnnn | nnnn | nnnn | nnnn |
|-------|------|------|------|------|------|------|

描述: 自当前 PC 向前或向后的 32K 程序字存储空间范围内进行相对子程序调用。
在进行调用之前，返回地址 (PC + 2) 被压入堆栈。在返回地址压入堆栈之后，符号扩展后的 17 位值 (2 * Slit16) 将与 PC 的内容相加且结果存入 PC。
n 位为一个有符号立即数，用于指定相对调用距离 (PC + 2) 的长度（以程序字为单位）。

注: 由于该指令将只占用一个字的程序存储区，因此应尽可能使用该指令而非 CALL 指令。

指令字数: 1

指令周期数: 4

例 1:

| | | | |
|--------|---------|------------|-------------------------|
| 012004 | RCALL | _Task1 | ; 调用 _Task1 |
| 012006 | ADD | W0, W1, W2 | |
| . | ... | | |
| . | ... | | |
| 012458 | _Task1: | SUB | W0, W2, W3 ; _Task1 子程序 |
| 01245A | ... | | |

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 01 2004 | PC | 01 2458 |
| W15 | 0810 | W15 | 0814 |
| 数据单元 0810 | FFFF | 数据单元 0810 | 2006 |
| 数据单元 0812 | FFFF | 数据单元 0812 | 0001 |
| SR | 0000 | SR | 0000 |

例 2:

00620E
006210
.
.
007000
007002

RCALL
MOV
...
...
CLR
...

_Init
W0, [W4++]

W2

; 调用 _Init

; _Init 子程序

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 00 620E | PC | 00 7000 |
| W15 | 0C50 | W15 | 0C54 |
| 数据单元 0C50 | FFFF | 数据单元 0C50 | 6210 |
| 数据单元 0C52 | FFFF | 数据单元 0C52 | 0000 |
| SR | 0000 | SR | 0000 |

RCALL

计算相对调用

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} RCALL Wn

操作数: Wn ∈ [W0 ... W15]
操作: (PC) + 2 → PC
(PC<15:0>) → (TOS)
(W15) + 2 → W15
(PC<22:16>) → (TOS)
(W15) + 2 → W15
(PC) + (2 * (Wn)) → PC
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0010 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 进行由工作寄存器 Wn 指定的计算相对子程序调用。调用范围是自当前 PC 向前或向后的 32K 程序字存储空间。在进行调用之前，返回地址 (PC + 2) 被压入堆栈。在返回地址压入堆栈之后，符号扩展后的 17 位值 (2 * (Wn)) 将与 PC 的内容相加且结果存入 PC。必须使用寄存器直接寻址对 Wn 进行寻址。
s 位用于选择源寄存器。

指令字数: 1
指令周期数: 2

例 1: 00FF8C EX1: INC W2, W3 ; RCALL 的目标
 00FF8E ...

 010008
 01000A RCALL W6 ; 使用 W6 执行 RCALL
 01000C MOVE W4, [W10]

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 01 000A | PC | 00 FF8C |
| W6 | FFC0 | W6 | FFC0 |
| W15 | 1004 | W15 | 1008 |
| 数据单元 1004 | 98FF | 数据单元 1004 | 000C |
| 数据单元 1006 | 2310 | 数据单元 1006 | 0001 |
| SR | 0000 | SR | 0000 |

例 2:

000302

000304

.

.

000450

000452

RCALL

FF1L

...

CLR

...

W2

W0, W1

...

W2

...

; 使用 W2 执行 RCALL

; RCALL 的目标

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 0302 | PC | 00 0450 |
| W2 | 00A6 | W2 | 00A6 |
| W15 | 1004 | W15 | 1008 |
| 数据单元 1004 | 32BB | 数据单元 1004 | 0304 |
| 数据单元 1006 | 901A | 数据单元 1006 | 0000 |
| SR | 0000 | SR | 0000 |

RCALL

计算相对调用

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} RCALL Wn

操作数: Wn ∈ [W0 ... W15]

操作:
(PC) + 2 → PC
(PC<15:1>) → TOS<15:1>, SFA 位 → TOS<0>
(W15) + 2 → W15
(PC<22:16>) → (TOS)
(W15) + 2 → W15
0 → SFA 位
(PC) + (2 * (Wn)) → PC
NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0001 | 0000 | 0010 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 进行由工作寄存器 Wn 指定的计算相对子程序调用。调用范围是自当前 PC 向前或向后的 32K 程序字存储空间。在进行调用之前，返回地址 (PC + 2) 被压入堆栈。在返回地址压入堆栈之后，符号扩展后的 17 位值 (2 * (Wn)) 将与 PC 的内容相加且结果存入 PC。必须使用寄存器直接寻址对 Wn 进行寻址。

s 位用于选择源寄存器。

指令字数: 1

指令周期数: 4

例 1: 00FF8C EX1: INC W2, W3 ; RCALL 的目标
 00FF8E ...

 010008
 01000A RCALL W6 ; 使用 W6 执行 RCALL
 01000C MOVE W4, [W10]

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 01 000A | PC | 00 FF8C |
| W6 | FFC0 | W6 | FFC0 |
| W15 | 1004 | W15 | 1008 |
| 数据单元 1004 | 98FF | 数据单元 1004 | 000C |
| 数据单元 1006 | 2310 | 数据单元 1006 | 0001 |
| SR | 0000 | SR | 0000 |

例 2:

000302

000304

.

.

000450

000452

RCALL

FF1L

...

CLR

...

W2

W0, W1

...

W2

...

; 使用 W2 执行 RCALL

; RCALL 的目标

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 0302 | PC | 00 0450 |
| W2 | 00A6 | W2 | 00A6 |
| W15 | 1004 | W15 | 1008 |
| 数据单元 1004 | 32BB | 数据单元 1004 | 0304 |
| 数据单元 1006 | 901A | 数据单元 1006 | 0000 |
| SR | 0000 | SR | 0000 |

REPEAT

重复执行下一条指令 (lit14 + 1) 次

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} REPEAT #lit14

操作数: lit14 ∈ [0 ... 16383]

操作: (lit14) → RCOUNT
(PC) + 2 → PC
使能代码循环

受影响的状态位: RA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1001 | 00kk | kkkk | kkkk | kkkk |
|-------|------|------|------|------|------|------|

描述: 重复执行紧跟 REPEAT 指令之后的指令 (lit14 + 1) 次。在所有重复执行过程中, 被重复指令 (或称为目标指令) 将被保存在指令寄存器中且该指令只被取一次。
当执行该指令时, RCOUNT 寄存器将装入指令中指定的重复计数值。在每一次执行目标指令后, RCOUNT 将递减。当 RCOUNT 等于零时, 目标指令将再执行一次, 随后继续正常执行目标指令之后的指令。
k 位是用于指定循环计数值的无符号立即数。

特征和限制:

1. 当指定重复次数的立即数为 0 时, REPEAT 指令功能与 NOP 相同且 RA 位将不会被置 1。
2. REPEAT 指令的目标指令不能是:
 - 改变程序流的指令
 - DO、DISI、LNK、MOV.D、PWRSAV、REPEAT 或 ULNK 指令
 - 双字指令

如果用上述这些指令作为目标指令, 可能会发生意想不到的结果。

注: REPEAT 和目标指令都是可中断的。

指令字数: 1

指令周期数: 1

例 1: 000452 REPEAT #9 ; 执行 ADD 10 次
000454 ADD [W0++], W1, [W2++] ; 向量更新

| | | | |
|--------|---------|--------|---------------|
| | 指令执行前 | | 指令执行后 |
| PC | 00 0452 | PC | 00 0454 |
| RCOUNT | 0000 | RCOUNT | 0009 |
| SR | 0000 | SR | 0010 (RA = 1) |

例 2:

00089E

REPEAT

#0x3FF

; 执行 CLR 1024 次

0008A0

CLR

[W6++]

; 清除暂存空间

| 指令执行前 | | 指令执行后 | |
|--------|---------|--------|---------------|
| PC | 00 089E | PC | 00 08A0 |
| RCOUNT | 0000 | RCOUNT | 03FF |
| SR | 0000 | SR | 0010 (RA = 1) |

REPEAT

重复执行下一条指令 (lit15 + 1) 次

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} REPEAT #lit15

操作数: lit15 ∈ [0 ... 32767]

操作: (lit15) → RCOUNT
(PC) + 2 → PC
使能代码循环

受影响的状态位: RA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1001 | 0kkk | kkkk | kkkk | kkkk |
|-------|------|------|------|------|------|------|

描述: 重复执行紧跟 REPEAT 指令之后的指令 (lit15 + 1) 次。在所有重复执行过程中, 被重复指令 (或称为目标指令) 将被保存在指令寄存器中且该指令只被取一次。
当执行该指令时, RCOUNT 寄存器将装入指令中指定的重复计数值。在每一次执行目标指令后, RCOUNT 将递减。当 RCOUNT 等于零时, 目标指令将再执行一次, 随后继续正常执行目标指令之后的指令。
k 位是用于指定循环计数值的无符号立即数。

特征和限制:

- 1. 当指定重复次数的立即数为 0 时, REPEAT 指令功能与 NOP 相同且 RA 位将不会被置 1。
- 2. REPEAT 指令的目标指令不能是:
 - 改变程序流的指令
 - DISI、LNK、MOV.D、PWRSAV、REPEAT 或 ULNK 指令
 - 双字指令

如果用上述这些指令作为目标指令, 可能会发生意想不到的结果。

注: REPEAT 和目标指令都是可中断的。

指令字数: 1

指令周期数: 1

例 1: 000452 REPEAT #9 ; 执行 ADD 10 次
000454 ADD [W0++], W1, [W2++] ; 向量更新

| | | | | | |
|--------|---------|--|--------|---------|----------|
| 指令执行前 | | | 指令执行后 | | |
| PC | 00 0452 | | PC | 00 0454 | |
| RCOUNT | 0000 | | RCOUNT | 0009 | |
| SR | 0000 | | SR | 0010 | (RA = 1) |

例 2:

00089E

REPEAT

#0x3FF

; 执行 CLR 1024 次

0008A0

CLR

[W6++]

; 清除暂存空间

| 指令执行前 | | 指令执行后 | |
|--------|---------|--------|---------------|
| PC | 00 089E | PC | 00 08A0 |
| RCOUNT | 0000 | RCOUNT | 03FF |
| SR | 0000 | SR | 0010 (RA = 1) |

REPEAT

重复执行下一条指令 (Wn + 1) 次

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 :} REPEAT Wn

操作数: Wn ∈ [W0 ... W15]

操作: (Wn<13:0>) → RCOUNT
(PC) + 2 → PC
使能代码循环

受影响的状态位: RA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1001 | 1000 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 重复执行紧跟 REPEAT 指令之后的指令 (Wn<13:0> + 1) 次。在所有重复执行过程中，被重复指令（或称为目标指令）将被保存在指令寄存器中且该指令只被取一次。

当执行该指令时，RCOUNT 寄存器将装入 Wn 的低 14 位值。在每一次执行目标指令后，RCOUNT 将递减。当 RCOUNT 等于零时，目标指令将再执行一次，随后继续正常执行目标指令之后的指令。

s 位用于指定包含重复计数值的 Wn 寄存器。

特征和限制:

- 1. 当(Wn) = 0时，REPEAT指令功能与NOP相同且RA位将不会被置1。
- 2. REPEAT 指令的目标指令不能是:
 - 改变程序流的指令
 - DO、DISI、LNK、MOV.D、PWRSAV、REPEAT 或 ULNK 指令
 - 双字指令

如果用上述这些指令作为目标指令，可能会发生意想不到的结果。

注: REPEAT 和目标指令都是可中断的。

指令字数: 1

指令周期数: 1

例 1: 000A26 REPEAT W4 ; 执行 COM (W4+1) 次
000A28 COM [W0++], [W2++] ; 向量求反

| | | | |
|--------|---------|--------|---------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 0A26 | PC | 00 0A28 |
| W4 | 0023 | W4 | 0023 |
| RCOUNT | 0000 | RCOUNT | 0023 |
| SR | 0000 | SR | 0010 (RA = 1) |

例 2: 00089E REPEAT W10 ; 执行 TBLRD (W10+1) 次
 0008A0 TBLRDL [W2++], [W3++] ; 递减 (0x840)

| 指令执行前 | | 指令执行后 | |
|--------|---------|--------|---------------|
| PC | 00 089E | PC | 00 08A0 |
| W10 | 00FF | W10 | 00FF |
| RCOUNT | 0000 | RCOUNT | 00FF |
| SR | 0000 | SR | 0010 (RA = 1) |

REPEAT

重复执行下一条指令 (Wn + 1) 次

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} REPEAT Wn

操作数: Wn ∈ [W0 ... W15]

操作: (Wn) → RCOUNT
(PC) + 2 → PC
使能代码循环

受影响的状态位: RA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 1001 | 1000 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 重复执行紧跟 REPEAT 指令之后的指令 (Wn + 1) 次。在所有重复执行过程中, 被重复指令 (或称为目标指令) 将被保存在指令寄存器中且该指令只被取一次。

当执行该指令时, RCOUNT 寄存器将装入 Wn 内容。在每一次执行目标指令后, RCOUNT 将递减。当 RCOUNT 等于零时, 目标指令将再执行一次, 随后继续正常执行目标指令之后的指令。

s 位用于指定包含重复计数值的 Wn 寄存器。

特征和限制:

- 1. 当(Wn) = 0时, REPEAT指令功能与NOP相同且RA位将不会被置1。
- 2. REPEAT 指令的目标指令不能是:
 - 改变程序流的指令
 - DO、DISI、LNK、MOV.D、PWRSAV、REPEAT 或 ULNK 指令
 - 双字指令

如果用上述这些指令作为目标指令, 可能会发生意想不到的结果。

注: REPEAT 和目标指令都是可中断的。

指令字数: 1

指令周期数: 1

例 1: 000A26 REPEAT W4 ; 执行 COM (W4+1) 次
000A28 COM [W0++], [W2++] ; 向量求反

| | | | |
|--------|---------|--------|---------------|
| 指令执行前 | | 指令执行后 | |
| PC | 00 0A26 | PC | 00 0A28 |
| W4 | 0023 | W4 | 0023 |
| RCOUNT | 0000 | RCOUNT | 0023 |
| SR | 0000 | SR | 0010 (RA = 1) |

例 2:

00089E REPEAT W10 ; 执行 TBLRD (W10+1) 次

0008A0 TBLRDL [W2++], [W3++] ; 递减 (0x840)

| 指令执行前 | | 指令执行后 | |
|--------|---------|--------|---------------|
| PC | 00 089E | PC | 00 08A0 |
| W10 | 00FF | W10 | 00FF |
| RCOUNT | 0000 | RCOUNT | 00FF |
| SR | 0000 | SR | 0010 (RA = 1) |

RESET

复位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RESET

操作数: 无

操作: 将所有受 $\overline{\text{MCLR}}$ 复位影响的寄存器的内容强制为其复位状态。
1 → SWR (RCON<6>)
0 → PC

受影响的状态位: OA、OB、OAB、SA、SB、SAB、DA、DC、IPL<2:0>、RA、N、OV、Z、C 和 SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1110 | 0000 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 该指令提供一种执行软件复位的方法。所有内核和外设寄存器将恢复至上电复位时的值。PC 将被设定为 0，即 RESET GOTO 指令的地址。SWR 位 (RCON<6>) 将被设定为 1 以表明执行了 RESET 指令。

注: 有关所有寄存器上电复位时的值，请参见具体器件的系列参考手册。

指令字数: 1

指令周期数: 1

例 1: 00202A RESET ; 在 dsPIC33F 上执行软件复位

| 指令执行前 | | 指令执行后 | |
|--------|-------------------|--------|------------------|
| PC | 00 202A | PC | 00 0000 |
| W0 | 8901 | W0 | 0000 |
| W1 | 08BB | W1 | 0000 |
| W2 | B87A | W2 | 0000 |
| W3 | 872F | W3 | 0000 |
| W4 | C98A | W4 | 0000 |
| W5 | AAD4 | W5 | 0000 |
| W6 | 981E | W6 | 0000 |
| W7 | 1809 | W7 | 0000 |
| W8 | C341 | W8 | 0000 |
| W9 | 90F4 | W9 | 0000 |
| W10 | F409 | W10 | 0000 |
| W11 | 1700 | W11 | 0000 |
| W12 | 1008 | W12 | 0000 |
| W13 | 6556 | W13 | 0000 |
| W14 | 231D | W14 | 0000 |
| W15 | 1704 | W15 | 0800 |
| SPLIM | 1800 | SPLIM | 0000 |
| TBLPAG | 007F | TBLPAG | 0000 |
| PSVPAG | 0001 | PSVPAG | 0000 |
| CORCON | 00F0 | CORCON | 0020 (SATDW = 1) |
| RCON | 0000 | RCON | 0040 (SWR = 1) |
| SR | 0021 (IPL, C = 1) | SR | 0000 |

RETFIE

从中断返回

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } RETFIE

操作数: 无

操作: (W15) - 2 → W15
(TOS<15:8>) → (SR<7:0>)
(TOS<7>) → (IPL3, CORCON<3>)
(TOS<6:0>) → (PC<22:16>)
(W15) - 2 → W15
(TOS<15:0>) → (PC<15:0>)
NOP → 指令寄存器

受影响的状态位: IPL<3:0>、RA、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0110 | 0100 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 从中断服务程序返回。第一次堆栈弹出操作将恢复状态寄存器的低位字节、IPL<3>（CORCON<3>）和 PC 的高位字节。第二次堆栈弹出操作将恢复 PC 低 16 位的内容。

- 注 1: 恢复 IPL<3> 及状态寄存器中的低位字节，并将中断优先级恢复至执行中断服务程序前的状态。
- 2: 在执行 RETFIE 之前，必须在程序中将相应的中断标志清零以避免递归中断。

指令字数: 1

指令周期数: 3（如果有异常等待处理，则为 2）

例 1: 000A26 RETFIE ; 从 ISR 返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|-----------------------|
| PC | 00 0A26 | PC | 01 0230 |
| W15 | 0834 | W15 | 0830 |
| 数据单元 0830 | 0230 | 数据单元 0830 | 0230 |
| 数据单元 0832 | 8101 | 数据单元 0832 | 8101 |
| CORCON | 0001 | CORCON | 0001 |
| SR | 0000 | SR | 0081 (IPL = 4, C = 1) |

例 2: 008050 RETFIE ; 从 ISR 返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|-----------------|
| PC | 00 8050 | PC | 00 7008 |
| W15 | 0926 | W15 | 0922 |
| 数据单元 0922 | 7008 | 数据单元 0922 | 7008 |
| 数据单元 0924 | 0300 | 数据单元 0924 | 0300 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0003 (Z, C = 1) |

RET FIE从中断返回

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } RET FIE

操作数: 无
操作: (W15) - 2 → W15
(TOS<15:8>) → (SR<7:0>)
(TOS<7>) → (IPL3, CORCON<3>)
(TOS<6:0>) → (PC<22:16>)
(W15) - 2 → W15
(TOS<15:1>) → (PC<15:1>)
TOS<0> → SFA 位
NOP → 指令寄存器

受影响的状态位: IPL<3:0>、RA、N、OV、Z、C 和 SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0110 | 0100 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 从中断服务程序返回。第一次堆栈弹出操作将恢复状态寄存器的低位字节、IPL<3>（CORCON<3>）和 PC 的高位字节。第二次堆栈弹出操作将恢复 PC 低 16 位的内容。

- 注 1: 恢复 IPL<3> 及状态寄存器中的低位字节，并将中断优先级恢复至执行中断服务程序前的状态。
- 2: 在执行 RET FIE 之前，必须在程序中将相应的中断标志清零以避免递归中断。

指令字数: 1
指令周期数: 6（如果有异常等待处理，则为 5）

例 1: 000A26 RET FIE ; 从 ISR 返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|-----------------------|
| PC | 00 0A26 | PC | 01 0230 |
| W15 | 0834 | W15 | 0830 |
| 数据单元 0830 | 0230 | 数据单元 0830 | 0230 |
| 数据单元 0832 | 8101 | 数据单元 0832 | 8101 |
| CORCON | 0001 | CORCON | 0001 |
| SR | 0000 | SR | 0081 (IPL = 4, C = 1) |

例 2: 008050 RET FIE ; 从 ISR 返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|-----------------|
| PC | 00 8050 | PC | 00 7008 |
| W15 | 0926 | W15 | 0922 |
| 数据单元 0922 | 7008 | 数据单元 0922 | 7008 |
| 数据单元 0924 | 0300 | 数据单元 0924 | 0300 |
| CORCON | 0000 | CORCON | 0000 |
| SR | 0000 | SR | 0003 (Z, C = 1) |

RETLW

返回并将立即数存入 Wn

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } RETLW{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: (W15) – 2 → W15
TOS<15:8> → SR<7:0>
TOS<7:0> → IPL<3> : PC<22:16>
(W15) – 2 → W15
(TOS) → (PC<15:0>)
lit10 → Wn
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0101 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 从子程序返回, 并将指定的 10 位无符号立即数存入 Wn。软件堆栈将弹出两次以恢复 PC 值, 并将有符号立即数存入 Wn。由于执行两次弹出操作, 堆栈指针 (W15) 将递减 4。

B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
k 位用于指定立即数的值。
d 位用于选择目标寄存器。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 3 (如果有异常等待处理, 则为 2)

例 1: 000440 RETLW.B #0xA, W0 ; 返回并将 0xA 存入 W0

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 0440 | PC | 00 7006 |
| W0 | 9846 | W0 | 980A |
| W15 | 1988 | W15 | 1984 |
| 数据单元 1984 | 7006 | 数据单元 1984 | 7006 |
| 数据单元 1986 | 0000 | 数据单元 1986 | 0000 |
| SR | 0000 | SR | 0000 |

例 2: 00050A RETLW #0x230, W2 ; 返回并将 0x230 存入 W2

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 050A | PC | 01 7008 |
| W2 | 0993 | W2 | 0230 |
| W15 | 1200 | W15 | 11FC |
| 数据单元 11FC | 7008 | 数据单元 11FC | 7008 |
| 数据单元 11FE | 0001 | 数据单元 11FE | 0001 |
| SR | 0000 | SR | 0000 |

RETLW

返回并将立即数存入 Wn

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 :} RETLW{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: (W15) – 2 → W15
TOS<15:8> → SR<7:0>
TOS<7:0> → IPL<3> : PC<22:16>
(W15) – 2 → W15
(TOS<15:1>) → (PC<15:1>)
TOS<0> → SFA 位
lit10 → Wn
NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0101 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 从子程序返回, 并将指定的 10 位无符号立即数存入 Wn。软件堆栈将弹出两次以恢复 PC 值, 并将有符号立即数存入 Wn。由于执行两次弹出操作, 堆栈指针 (W15) 将递减 4。

B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。
k 位用于指定立即数的值。
d 位用于选择目标寄存器。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 6 (如果有异常等待处理, 则为 5)

例 1: 000440 RETLW.B #0xA, W0 ; 返回并将 0xA 存入 W0

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 0440 | PC | 00 7006 |
| W0 | 9846 | W0 | 980A |
| W15 | 1988 | W15 | 1984 |
| 数据单元 1984 | 7006 | 数据单元 1984 | 7006 |
| 数据单元 1986 | 0000 | 数据单元 1986 | 0000 |
| SR | 0000 | SR | 0000 |

例 2: 00050A RETLW #0x230, W2 ; 返回并将 0x230 存入 W2

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 050A | PC | 01 7008 |
| W2 | 0993 | W2 | 0230 |
| W15 | 1200 | W15 | 11FC |
| 数据单元 11FC | 7008 | 数据单元 11FC | 7008 |
| 数据单元 11FE | 0001 | 数据单元 11FE | 0001 |
| SR | 0000 | SR | 0000 |

RETURN

返回

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法: { 标号 : } RETURN

操作数: 无

操作: (W15) – 2 → W15
(TOS) → (PC<22:16>)
(W15) – 2 → W15
(TOS) → (PC<15:0>)
NOP → 指令寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0110 | 0000 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 从子程序返回。软件堆栈将弹出两次以恢复 PC。由于执行两次弹出操作，堆栈指针（W15）将递减 4。

指令字数: 1

指令周期数: 3（如果有异常等待处理，则为 2）

例 1: 001A06 RETURN ; 从子程序返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 1A06 | PC | 01 0004 |
| W15 | 1248 | W15 | 1244 |
| 数据单元 1244 | 0004 | 数据单元 1244 | 0004 |
| 数据单元 1246 | 0001 | 数据单元 1246 | 0001 |
| SR | 0000 | SR | 0000 |

例 2: 005404 RETURN ; 从子程序返回

| 指令执行前 | | 指令执行后 | |
|-----------|---------|-----------|---------|
| PC | 00 5404 | PC | 00 0966 |
| W15 | 090A | W15 | 0906 |
| 数据单元 0906 | 0966 | 数据单元 0906 | 0966 |
| 数据单元 0908 | 0000 | 数据单元 0908 | 0000 |
| SR | 0000 | SR | 0000 |

RETURN 返回

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 : } RETURN

操作数: 无
操作: (W15) – 2 → W15
(TOS) → (PC<22:16>)
(W15) – 2 → W15
(TOS<15:1>) → (PC<15:1>)
TOS<0> → SFA 位
NOP → 指令寄存器

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0000 | 0110 | 0000 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 从子程序返回。软件堆栈将弹出两次以恢复 PC。由于执行两次弹出操作，堆栈指针（W15）将递减 4。

指令字数: 1
指令周期数: 6（如果有异常等待处理，则为 5）

例 1: 001A06 RETURN ; 从子程序返回

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 00 1A06 | PC | 01 0004 |
| W15 | 1248 | W15 | 1244 |
| 数据单元 1244 | 0004 | 数据单元 1244 | 0004 |
| 数据单元 1246 | 0001 | 数据单元 1246 | 0001 |
| SR | 0000 | SR | 0000 |

例 2: 005404 RETURN ; 从子程序返回

| | 指令执行前 | | 指令执行后 |
|-----------|---------|-----------|---------|
| PC | 00 5404 | PC | 00 0966 |
| W15 | 090A | W15 | 0906 |
| 数据单元 0906 | 0966 | 数据单元 0906 | 0966 |
| 数据单元 0908 | 0000 | 数据单元 0908 | 0000 |
| SR | 0000 | SR | 0000 |

RLC

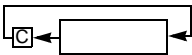
带进位位循环左移 f 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RLC{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
(C) → Dest<0>
(f<6:0>) → Dest<7:1>
(f<7>) → C
对于字操作:
(C) → Dest<0>
(f<14:0>) → Dest<15:1>
(f<15>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0110 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 带进位标志位将文件寄存器 f 中的内容左移一位，并将结果存放在目标寄存器中。状态寄存器中的进位标志位将移入目标寄存器的最低位，随后进位标志位将被文件寄存器 f 内容的最高位覆盖。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择文件寄存器，1 选择 WREG）。

f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1

例 1: RLC.B 0x1233 ; 带进位位将 (0x1233) 左移 1 位（字节模式）

| | | | |
|-----------|-------|-----------|-----------------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 1232 | E807 | 数据单元 1232 | D007 |
| SR | 0000 | SR | 0009 (N, C = 1) |

例 2: RLC 0x820, WREG ; 带进位位将 (0x820) 左移 1 位 (字模式)
 ; 结果存入 WREG

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|---------|-----------|------|---------|
| WREG (W0) | 5601 | | WREG (W0) | 42DD | |
| 数据单元 0820 | 216E | | 数据单元 0820 | 216E | |
| SR | 0001 | (C = 1) | SR | 0000 | (C = 0) |

RLC

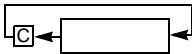
带进位位循环左移 **Ws** 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RLC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
(C) → Wd<0>
(Ws<6:0>) → Wd<7:1>
(Ws<7>) → C
对于字操作:
(C) → Wd<0>
(Ws<14:0>) → Wd<15:1>
(Ws<15>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0010 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 带进位标志位将源寄存器 **Ws** 中的内容左移一位，并将结果存放在目标寄存器 **Wd** 中。状态寄存器中的进位标志位将移入 **Wd** 的最低位，随后进位标志位将被 **Ws** 内容的最高位覆盖。可使用寄存器直接或间接寻址对 **Ws** 和 **Wd** 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 **.B** 指明是字节操作而非字操作。用户可使用 **.w** 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1

例 1: RLC.B W0, W3 ; 带进位位将 (W0) 左移 1 位（字节模式）
; 结果存入 W3

| 指令执行前 | | | 指令执行后 | | |
|-------|------|---------|-------|------|---------|
| W0 | 9976 | (C = 1) | W0 | 9976 | (N = 1) |
| W3 | 5879 | | W3 | 58ED | |
| SR | 0001 | | SR | 0009 | |

例 2: RLC [W2++], [W8] ; 带进位位将 [W2] 左移 1 位 (字模式)
 ; 指令执行后, 递增 W2
 ; 结果存入 [W8]

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|-----------------|
| W2 | 2008 | W2 | 200A |
| W8 | 094E | W8 | 094E |
| 数据单元 094E | 3689 | 数据单元 094E | 8082 |
| 数据单元 2008 | C041 | 数据单元 2008 | C041 |
| SR | 0001 (C = 1) | SR | 0009 (N, C = 1) |

RLNC

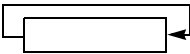
不带进位位循环左移 f 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RLNC{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
(f<6:0>) → Dest<7:1>
(f<7>) → Dest<0>
对于字操作:
(f<14:0>) → Dest<15:1>
(f<15>) → Dest<0>



受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0110 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器 f 中的内容左移一位，并将结果存放在目标寄存器中。f 中的最高位将存入目标寄存器的最低位，而进位标志位将不受影响。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: RLNC.B 0x1233 ; 将 (0x1233) 左移 1 位 (字节模式)

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| 数据单元 1232 | E807 | 数据单元 1233 | D107 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: RLNC 0x820, WREG ; 将 (0x820) 左移 1 位 (字模式)
 ; 结果存入 WREG

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| WREG (W0) | 5601 | WREG (W0) | 42DC |
| 数据单元 0820 | 216E | 数据单元 0820 | 216E |
| SR | 0001 (C = 1) | SR | 0000 (C = 0) |

RLNC

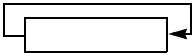
不带进位位循环左移 **Ws** 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RLNC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
(Ws<6:0>) → Wd<7:1>
(Ws<7>) → Wd<0>
对于字操作:
(Ws<14:0>) → Wd<15:1>
(Ws<15>) → Wd<0>



受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0010 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 **Ws** 中的内容左移一位，并将结果存放在目标寄存器 **Wd** 中。
Ws 中的最高位将存入 **Wd** 的最低位，而进位标志位将不受影响。可使用寄存器直接或间接寻址对 **Ws** 和 **Wd** 进行寻址。

B 位用于选择字节或字操作（0 选择字节操作，1 选择字操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 **.B** 指明是字节操作而非字操作。用户可使用 **.w** 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

RRC

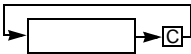
带进位位循环右移 f 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RRC{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
(C) → Dest<7>
(f<7:1>) → Dest<6:0>
(f<0>) → C
对于字操作:
(C) → Dest<15>
(f<15:1>) → Dest<14:0>
(f<0>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0111 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 带进位标志位将文件寄存器 f 中的内容右移一位，并将结果存放在目标寄存器中。状态寄存器中的进位标志位将移入目标寄存器的最高位，随后进位标志位将被文件寄存器 f 内容的最低位覆盖。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字节操作，1 选择字操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: RRC.B 0x1233 ; 带进位位将 (0x1233) 右移 1 位（字节模式）

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 1232 | E807 | 数据单元 1232 | 7407 |
| SR | 0000 | SR | 0000 |

例 2: RRC 0x820, WREG ; 带进位位将 (0x820) 右移 1 位 (字模式)
 ; 结果存入 WREG

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|---------|-----------|------|---------|
| WREG (W0) | 5601 | | WREG (W0) | 90B7 | |
| 数据单元 0820 | 216E | | 数据单元 0820 | 216E | |
| SR | 0001 | (C = 1) | SR | 0008 | (N = 1) |

RRC

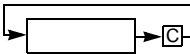
带进位位循环右移 **Ws** 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RRC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
(C) → Wd<7>
(Ws<7:1>) → Wd<6:0>
(Ws<0>) → C
对于字操作:
(C) → Wd<15>
(Ws<15:1>) → Wd<14:0>
(Ws<0>) → C



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0011 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 带进位标志位将源寄存器 **Ws** 中的内容右移一位，并将结果存放在目标寄存器 **Wd** 中。状态寄存器中的进位标志位将移入 **Wd** 的最高位，随后进位标志位将被 **Ws** 内容的最低位覆盖。可使用寄存器直接或间接寻址对 **Ws** 和 **Wd** 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 **.B** 指明是字节操作而非字操作。用户可使用 **.w** 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: RRC.B W0, W3 ; 带进位位将 (W0) 右移 1 位 (字节模式)
 ; 结果存入 W3

| 指令执行前 | | | 指令执行后 | | |
|-------|------|---------|-------|------|---------|
| W0 | 9976 | (C = 1) | W0 | 9976 | (N = 1) |
| W3 | 5879 | | W3 | 58BB | |
| SR | 0001 | | SR | 0008 | |

例 2: RRC [W2++], [W8] ; 带进位位将 [W2] 右移 1 位 (字模式)
 ; 指令执行后, 递增 W2
 ; 结果存入 [W8]

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|---------|-----------|------|------------|
| W2 | 2008 | (C = 1) | W2 | 200A | (N, C = 1) |
| W8 | 094E | | W8 | 094E | |
| 数据单元 094E | 3689 | | 数据单元 094E | E020 | |
| 数据单元 2008 | C041 | | 数据单元 2008 | C041 | |
| SR | 0001 | | SR | 0009 | |

RRNC

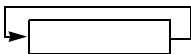
不带进位位循环右移 f 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RRNC{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
(f<7:1>) → Dest<6:0>
(f<0>) → Dest<7>
对于字操作:
(f<15:1>) → Dest<14:0>
(f<0>) → Dest<15>



受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0111 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器 f 中的内容右移一位，并将结果存放在目标寄存器中。f 中的最低位将存入目标寄存器的最高位，而进位标志位将不受影响。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: RRNC.B 0x1233 ; 将 (0x1233) 右移 1 位 (字节模式)

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| 数据单元 1232 | E807 | 数据单元 1232 | 7407 |
| SR | 0000 | SR | 0000 |

例 2: RRNC 0x820, WREG ; 将 (0x820) 右移 1 位 (字模式)
 ; 结果存入 WREG

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| WREG (W0) | 5601 | WREG (W0) | 10B7 |
| 数据单元 0820 | 216E | 数据单元 0820 | 216E |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

RRNC

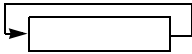
不带进位位循环右移 Ws 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} RRNC{.B} Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
(Ws<7:1>) → Wd<6:0>
(Ws<0>) → Wd<7>
对于字操作:
(Ws<15:1>) → Wd<14:0>
(Ws<0>) → Wd<15>



受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0011 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容右移一位，并将结果存放在目标寄存器 Wd 中。Ws 中的最低位将存入 Wd 的最高位，而进位标志位将不受影响。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: RRNC.B W0, W3 ; 将 (W0) 右移 1 位 (字节模式)
 ; 结果存入 W3

| 指令执行前 | | | 指令执行后 | | |
|-------|------|---------|-------|------|---------|
| W0 | 9976 | (C = 1) | W0 | 9976 | (C = 1) |
| W3 | 5879 | | W3 | 583B | |
| SR | 0001 | | SR | 0001 | |

例 2: RRNC [W2++], [W8] ; 将 [W2] 右移 1 位 (字模式)
 ; 指令执行后, 递增 W2
 ; 结果存入 [W8]

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|-----------|-----------|------|-----------|
| W2 | 2008 | 数据单元 094E | W2 | 200A | 数据单元 094E |
| W8 | 094E | | W8 | 094E | |
| 数据单元 094E | 3689 | | 数据单元 094E | E020 | |
| 数据单元 2008 | C041 | | 数据单元 2008 | C041 | |
| SR | 0000 | | SR | 0008 | (N = 1) |

SAC

保存累加器内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} SAC Acc, {#Slit4,} Wd
[Wd]
[Wd++]
[Wd--]
[--Wd]
[++Wd]
[Wd + Wb]

操作数: Acc ∈ [A,B]
Slit4 ∈ [-8 ... +7]
Wb, Wd ∈ [W0 ... W15]

操作: Shift_{Slit4}(Acc) (可选)
(Acc[31:16]) → Wd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1100 | Awww | wrrr | rrhh | dddd |
|-------|------|------|------|------|------|------|

描述: 对指定的累加器执行可选的有符号 4 位移位操作，随后将移位后的 ACCxH (Acc[31:16]) 内容存入 Wd。移位位数范围为 -8:7，其中负操作数表明将执行算术左移操作，而正操作数表明将执行算术右移操作。可使用寄存器直接或间接寻址对 Wd 进行寻址。

A 位用于指定源累加器。
w 位用于指定偏移量寄存器 Wb。
r 位用于对可选的累加器预移位进行编码。
h 位用于选择目标地址模式。
d 位用于指定目标寄存器 Wd。

- 注 1: 该指令不修改累加器的内容。
2: 该指令对截取的累加器内容进行保存。可使用指令 SAC.R 对舍入后的累加器内容进行保存。
3: 如果使能数据写饱和功能 (SATDW 位 (CORCON<5>) = 1)，在执行可选的移位操作后，存入 Wd 的值服从饱和规律。

指令字数: 1
指令周期数: 1

例 1: SAC A, #4, W5
; ACCA 右移 4 位
; 结果存入 W5
; CORCON = 0x0010 (SATDW = 1)

| 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|--------------|
| W5 | B900 | W5 | 0120 |
| ACCA | 00 120F FF00 | ACCA | 00 120F FF00 |
| CORCON | 0010 | CORCON | 0010 |
| SR | 0000 | SR | 0000 |

例 2:

SAC B, #-4, [W5++]
; ACCB 左移 4 位
; 结果存入 [W5], 指令执行后递增 W5
; CORCON = 0x0010 (SATDW = 1)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W5 | 2000 | W5 | 2002 |
| ACCB | FF C891 8F4C | ACCB | FF C891 1F4C |
| 数据单元 2000 | 5BBE | 数据单元 2000 | 8000 |
| CORCON | 0010 | CORCON | 0010 |
| SR | 0000 | SR | 0000 |

SAC.R

保存舍入后的累加器内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} SAC.R Acc, {#Slit4,} Wd
[Wd]
[Wd++]
[Wd--]
[--Wd]
[++Wd]
[Wd + Wb]

操作数: Acc ∈ [A,B]
Slit4 ∈ [-8 ... +7]
Wb ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: Shift_{Slit4}(Acc) (可选)
舍入 (Acc)
(Acc[31:16]) → Wd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|-------|------|
| 指令编码: | 1100 | 1101 | Awww | wrrr | rh hh | dddd |
|-------|------|------|------|------|-------|------|

描述: 对指定的累加器执行可选的有符号 4 位移位操作，随后将舍入后的 ACCxH (Acc[31:16]) 内容存入 Wd。移位位数范围为 -8:7，其中负操作数表明将执行算术左移操作，而正操作数表明将执行算术右移操作。RND 位 (CORCON<1>) 用于对舍入模式 (普通或收敛) 进行设定。可使用寄存器直接或间接寻址对 Wd 进行寻址。

A 位用于指定源累加器。
w 位用于指定偏移量寄存器 Wb。
r 位用于对可选的累加器预移位进行编码。
h 位用于选择目标地址模式。
d 位用于指定目标寄存器 Wd。

- 注 1: 该指令不修改累加器的内容。
- 2: 该指令对舍入的累加器内容进行保存。可使用指令 SAC 对截取的累加器内容进行保存。
- 3: 如果使能数据写饱和功能 (SATDW 位 (CORCON<5>) = 1)，在执行可选的移位操作后，存入 Wd 的值服从饱和规律。

指令字数: 1

指令周期数: 1

例 1:

SAC.R A, #4, W5
; ACCA 右移 4 位
; 舍入后的结果存入 W5
; CORCON = 0x0010 (SATDW = 1)

| 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|--------------|
| W5 | B900 | W5 | 0121 |
| ACCA | 00 120F FF00 | ACCA | 00 120F FF00 |
| CORCON | 0010 | CORCON | 0010 |
| SR | 0000 | SR | 0000 |

例 2:

SAC.R B, #-4, [W5++]
; ACCB 左移 4 位
; 舍入后的结果存入 [W5], 指令执行后递增 W5
; CORCON = 0x0010 (SATDW = 1)

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|--------------|
| W5 | 2000 | W5 | 2002 |
| ACCB | FF F891 8F4C | ACCB | FF F891 8F4C |
| 数据单元 2000 | 5BBE | 数据单元 2000 | 8919 |
| CORCON | 0010 | CORCON | 0010 |
| SR | 0000 | SR | 0000 |

SE

符号扩展 Ws 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SE Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]
操作: Ws<7:0> → Wnd<7:0>
如果 (Ws<7> = 1):
0xFF → Wnd<15:8>
否则:
0 → Wnd<15:8>

受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1011 | 0000 | 0ddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将 Ws 中的字节进行符号扩展，并将扩展后的 16 位结果存入 Wnd。可使用寄存器直接或间接寻址对 Ws 进行寻址。必须使用寄存器直接寻址对 Wnd 进行寻址。C 标志将设定为 N 标志的反码。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该操作将字节转换为字，且不使用 .B 或 .W 扩展符。
2: 源寄存器 Ws 将作为字节操作数进行寻址，因此所有地址改变都以 1 为单位。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SE W3, W4 ; 对 W3 内容进行符号扩展并将结果存入 W4

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W3 | 7839 | W3 | 7839 |
| W4 | 1005 | W4 | 0039 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: SE [W2++], W12 ; 对 [W2] 内容进行符号扩展并将结果存入 W12
; 指令执行后, 递增 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| W2 | 0900 | W2 | 0901 |
| W12 | 1002 | W12 | FF8F |
| 数据单元 0900 | 008F | 数据单元 0900 | 008F |
| SR | 0000 | SR | 0008 (N = 1) |

SETM

将 f 或 WREG 置全 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SETM{.B} f
WREG

操作数: f ∈ [0 ... 8191]

操作: 对于字节操作:
0xFF → 由 D 指定的目标寄存器
对于字操作:
0xFFFF → 由 D 指定的目标寄存器

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1110 | 1111 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 指定寄存器的所有位都将被置 1。如果指定了 WREG，WREG 的所有位将被置 1。否则，指定文件寄存器的所有位将被置 1。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1

例 1: SETM.B 0x891 ; 将 0x891 内容置全 1（字节模式）

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| 数据单元 0890 | 2739 | 数据单元 0890 | FF39 |
| SR | 0000 | SR | 0000 |

例 2: SETM WREG ; 将 WREG 内容置全 1（字模式）

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| WREG (W0) | 0900 | WREG (W0) | FFFF |
| SR | 0000 | SR | 0000 |

SETM

将 Ws 置全 1

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SETM{.B} Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wd ∈ [W0 ... W15]
操作: 对于字节操作:
对于字节操作, 0xFF → Wd
对于字操作:
对于字操作, 0xFFFF → Wd

| | | | | | | |
|----------|------|------|------|------|------|------|
| 受影响的状态位: | 无 | | | | | |
| 指令编码: | 1110 | 1011 | 1Bqq | qddd | d000 | 0000 |

描述: 指定寄存器的所有位都将被置 1。可使用寄存器直接或间接寻址对 Wd 进行寻址。
B 位用于选择字节或字操作（0 选择字操作, 1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。

指令字数: 1
指令周期数: 1

例 1: SETM.B W13 ; 将 W13 内容置全 1（字节模式）

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W13 | 2739 | W13 | 27FF |
| SR | 0000 | SR | 0000 |

例 2: SETM [--W6] ; 执行指令前, 递减 W6（字模式）
; 将 [W6] 内容置全 1

| | | | |
|-----------|------|-----------|------|
| 指令执行前 | | 指令执行后 | |
| W6 | 1250 | W6 | 124E |
| 数据单元 124E | 3CD9 | 数据单元 124E | FFFF |
| SR | 0000 | SR | 0000 |

SFTAC

算术移位累加器内容 Slit6 位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } SFTAC Acc, #Slit6

操作数: Acc ∈ [A,B]
Slit6 ∈ [-16 ... 16]

操作: Shift_k(Acc) → Acc

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1000 | A000 | 0000 | 01kk | kkkk |
|-------|------|------|------|------|------|------|

描述: 对指定累加器的 40 位内容进行算术移位操作，移位位数由 6 位有符号立即数确定，移位后的结果将存回累加器。移位位数范围为 -16:16，其中负操作数表明将执行左移操作，而正操作数表明将执行右移操作。任何移出累加器的位的信息都将丢失。

A 位用于选择存放结果的累加器。

k 位用于确定移位的位数。

- 注 1: 如果使能目标累加器的饱和功能（SATA 位（CORCON<7>）或 SATB 位（CORCON<6>）），则存入累加器的值服从饱和规律。
- 2: 如果移位位数大于 16 或小于 -16，则累加器内容将不会改变，并将导致算术陷阱发生。

指令字数: 1

指令周期数: 1

例 1: SFTAC A, #12
; 算术右移 ACCA 12 位
; 结果存入 ACCA
; CORCON = 0x0080 (SATA = 1)

| | | | |
|--------|--------------|--------|--------------|
| | 指令执行前 | | 指令执行后 |
| ACCA | 00 120F FF00 | ACCA | 00 0001 20FF |
| CORCON | 0080 | CORCON | 0080 |
| SR | 0000 | SR | 0000 |

例 2: SFTAC B, #-10
; 算术左移 ACCB 10 位
; 结果存入 ACCB
; CORCON = 0x0040 (SATB = 1)

| | | | |
|--------|--------------|--------|--------------|
| | 指令执行前 | | 指令执行后 |
| ACCB | FF FFF1 8F4C | ACCB | FF C63D 3000 |
| CORCON | 0040 | CORCON | 0040 |
| SR | 0000 | SR | 0000 |

SFTAC

算术移位累加器内容 Wb 位

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 :} SFTAC Acc, Wb

操作数: Acc ∈ [A,B]
Wb ∈ [W0 ... W15]

操作: Shift_(Wb)(Acc) → Acc

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1000 | A000 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 对指定累加器的 40 位内容进行算术移位操作，移位后的结果将存回累加器。Wb 中的低 6 位用于指定移位位数。移位位数范围为 -16:16，其中负值表明将执行左移操作，而正值表明将执行右移操作。任何移出累加器的位的信息都将丢失。

A 位用于选择作为源 / 目标的累加器。
s 位用于选择移位计数寄存器的地址。

- 注 1: 如果使能目标累加器的饱和功能（SATA 位（CORCON<7>）或 SATB 位（CORCON<6>）），则存入累加器的值服从饱和规律。
- 2: 如果移位位数大于 16 或小于 -16，则累加器内容将不会改变，并将导致算术陷阱发生。

指令字数: 1
指令周期数: 1

例 1: SFTAC A, W0
; 算术移位 ACCA (W0) 位
; 结果存入 ACCA
; CORCON = 0x0000 (饱和禁止)

| 指令执行前 | | | | 指令执行后 | | | |
|--------|--|--------------|--|--------|--|--------------|---------------|
| W0 | | FFFC | | W0 | | FFFC | |
| ACCA | | 00 320F AB09 | | ACCA | | 03 20FA B090 | |
| CORCON | | 0000 | | CORCON | | 0000 | |
| SR | | 0000 | | SR | | 8800 | (OA, OAB = 1) |

例 2: SFTAC B, W12
; 算术移位 ACCB (W12) 位
; 结果存入 ACCB
; CORCON = 0x0040 (SATB = 1)

| 指令执行前 | | | | 指令执行后 | | | |
|--------|--|--------------|--|--------|--|--------------|--|
| W12 | | 000F | | W12 | | 000F | |
| ACCB | | FF FFF1 8F4C | | ACCB | | FF FFFF FFE3 | |
| CORCON | | 0040 | | CORCON | | 0040 | |
| SR | | 0000 | | SR | | 0000 | |

SL

左移 f 内容

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SL{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: 对于字节操作:
(f<7>) → (C)
(f<6:0>) → Dest<7:1>
0 → Dest<0>
对于字操作:
(f<15>) → (C)
(f<14:0>) → Dest<15:1>
0 → Dest<0>



受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 0100 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将文件寄存器中的内容左移一位，并将结果存放在目标寄存器中。文件寄存器中的最高位将移入状态寄存器的进位位，而目标寄存器的最低位将移入 0。

可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SL.B 0x909 ; 将 (0x909) 左移 1 位（字节模式）

| | | | |
|-----------|-------|-----------|--------------|
| | 指令执行前 | | 指令执行后 |
| 数据单元 0908 | 9439 | 数据单元 0908 | 0839 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: SL 0x1650, WREG ; 将 (0x1650) 左移 1 位 (字模式)
 ; 结果存入 WREG

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| WREG (W0) | 0900 | WREG (W0) | 80CA |
| 数据单元 1650 | 4065 | 数据单元 1650 | 4065 |
| SR | 0000 | SR | 0008 (N = 1) |

实现于：

| PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
|--------|--------|--------|----------|----------|----------|
| X | X | X | X | X | X |

| | | | | |
|-----|---------|--------|---------|--------|
| 语法: | { 标号 :} | SL{.B} | Ws, | Wd |
| | | | [Ws], | [Wd] |
| | | | [Ws++], | [Wd++] |
| | | | [Ws--], | [Wd--] |
| | | | [++Ws], | [++Wd] |
| | | | [--Ws], | [--Wd] |

操作数: Ws ∈ [W0 ... W15]
 Wd ∈ [W0 ... W15]

操作:

对于字节操作:

(Ws<7>) → C

(Ws<6:0>) → Wd<7:1>

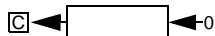
0 → Wd<0>

对于字操作:

(Ws<15>) → C

(Ws<14:0>) → Wd<15:1>

0 → Wd<0>



受影响的状态位: N、Z 和 C

指令编码:

| | | | | | |
|------|------|------|------|------|------|
| 1101 | 0000 | 0Bqq | qddd | dppp | ssss |
|------|------|------|------|------|------|

描述:

将源寄存器 **Ws** 中的内容左移一位，并将结果存放在目标寄存器 **Wd** 中。**Ws** 中的最高位将移入状态寄存器的进位位，而 **Wd** 的最低位将移入 0。可使用寄存器直接或间接寻址对 **Ws** 和 **Wd** 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

q 位用于选择目标地址模式。

d 位用于选择目标寄存器。

p 位用于选择源地址模式。

S 位用于选择源寄存器。

注：指令中的扩展符 `.B` 指明是字节操作而非字操作。用户可使用 `.W` 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中, 列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息, 请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SL.B W3, W4 ; 将 W3 左移 1 位 （字节模式）
 ; 结果存入 W4

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W3 | 78A9 | W3 | 78A9 |
| W4 | 1005 | W4 | 1052 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: SL [W2++], [W12] ; 将 [W2] 左移 1 位 （字模式）
 ; 结果存入 [W12]
 ; 指令执行后，递增 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| W2 | 0900 | W2 | 0902 |
| W12 | 1002 | W12 | 1002 |
| 数据单元 0900 | 800F | 数据单元 0900 | 800F |
| 数据单元 1002 | 6722 | 数据单元 1002 | 001E |
| SR | 0000 | SR | 0001 (C = 1) |

SL

左移（移位位数由短立即数确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } SL Wb, #lit4, Wnd

操作数: Wb ∈ [W0 ... W15]
lit4 ∈ [0 ... 15]
Wnd ∈ [W0 ... W15]

操作: lit4<3:0> → Shift_Val
Wnd<15:Shift_Val> = Wb<15-Shift_Val:0>
Wd<Shift_Val – 1:0> = 0

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1101 | 1101 | 0www | wddd | d100 | kkkk |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Wb 中的内容左移，移位位数由 4 位无符号立即数确定，并将结果存入目标寄存器 Wnd。所有移出源寄存器的位的信息将丢失。必须使用直接寻址对 Wb 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 该指令只能工作于字模式。

指令字数: 1

指令周期数: 1

例 1: SL W2, #4, W2 ; 左移 W2 4 位
; 结果存入 W2

| | | | |
|-------|------|-------|--------------|
| 指令执行前 | | 指令执行后 | |
| W2 | 78A9 | W2 | 8A90 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: SL W3, #12, W8 ; 左移 W3 12 位
; 结果存入 W8

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W3 | 0912 | W3 | 0912 |
| W8 | 1002 | W8 | 2000 |
| SR | 0000 | SR | 0000 |

SL左移（移位位数由 Wns 确定）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法：{ 标号 : } SL Wb, Wns, Wnd

操作数：Wb ∈ [W0 ... W15]
Wns ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]

操作：Wns<4:0> → Shift_Val
Wnd<15:Shift_Val> = Wb<15 – Shift_Val:0>
Wd<Shift_Val – 1:0> = 0

受影响的状态位：N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1101 | 1101 | 0www | wddd | d000 | ssss |
|-------|------|------|------|------|------|------|

描述：将源寄存器 Wb 中的内容左移，移位位数由 Wns 的低 5 位确定（最大移位位数为 15 位），并将结果存入目标寄存器 Wnd。所有移出源寄存器的位的信息将丢失。必须使用寄存器直接寻址对 Wb、Wns 和 Wnd 进行寻址。

w 位用于选择基准寄存器的地址。
d 位用于选择目标寄存器。
s 位用于选择源寄存器。

注 1: 该指令只能工作于字模式。
2: 如果 Wns 内容大于 15，Wnd 将装入 0x0。

指令字数：1
指令周期数：1

例 1: SL W0, W1, W2 ; 左移 W0 内容，移位位数由 W1<0:4> 确定
; 结果存入 W2

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 09A4 | W0 | 09A4 |
| W1 | 8903 | W1 | 8903 |
| W2 | 78A9 | W2 | 4D20 |
| SR | 0000 | SR | 0000 |

例 2: SL W4, W5, W6 ; 左移 W4 内容，移位位数由 W5<0:4> 确定
; 结果存入 W6

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W4 | A409 | W4 | A409 |
| W5 | FF01 | W5 | FF01 |
| W6 | 0883 | W6 | 4812 |
| SR | 0000 | SR | 0000 |

SUB

f 减 WREG

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUB{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $(f) - (WREG) \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0101 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 从指定的文件寄存器内容中减去默认工作寄存器 WREG 的内容，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUB.B 0x1FFF ; 从 (0x1FFF) 中减去 WREG 内容（字节模式）
; 结果存入 0x1FFF

| | 指令执行前 | 指令执行后 |
|-----------|-------|--------------|
| WREG (W0) | 7804 | 7804 |
| 数据单元 1FFE | 9439 | 9039 |
| SR | 0000 | 0001 (C = 1) |

例 2: SUB 0xA04, WREG ; 从 (0xA04) 中减去 WREG 内容（字模式）
; 结果存入 WREG

| | 指令执行前 | 指令执行后 |
|-----------|-------|--------------|
| WREG (W0) | 6234 | E2EF |
| 数据单元 0A04 | 4523 | 4523 |
| SR | 0000 | 0008 (N = 1) |

SUB

Wn 减立即数

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } SUB{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: (Wn) − lit10 → Wn

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0001 | 0Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 从工作寄存器 Wn 内容中减去 10 位无符号立即数操作数, 并将结果存回工作寄存器 Wn。必须使用寄存器直接寻址对 Wn 进行寻址。

B 位用于选择字节或字操作。
k 位用于指定立即数操作数。
d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 1

例 1: SUB.B #0x23, W0 ; 从 W0 内容中减去 0x23 (字节模式)
; 结果存入 W0

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 7804 | W0 | 78E1 |
| SR | 0000 | SR | 0008 (N = 1) |

例 2: SUB #0x108, W4 ; 从 W4 内容中减去 0x108 (字模式)
; 结果存入 W4

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W4 | 6234 | W4 | 612C |
| SR | 0000 | SR | 0001 (C = 1) |

SUB

Wb 减短立即数

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUB{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: (Wb) – lit5 → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0101 | 0www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 从基准寄存器 Wb 内容中减去 5 位无符号立即数操作数，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: SUB.B W4, #0x10, W5 ; 从 W4 内容中减去 0x10（字节模式）
; 结果存入 W5

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------------------|
| W4 | 1782 | W4 | 1782 |
| W5 | 7804 | W5 | 7872 |
| SR | 0000 | SR | 0005 (OV, C = 1) |

例 2: SUB W0, #0x8, [W2++] ; 从 W0 内容中减去 0x8（字模式）
; 结果存入 [W2]
; 指令执行后，递增 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|-----------------|
| W0 | F230 | W0 | F230 |
| W2 | 2004 | W2 | 2006 |
| 数据单元 2004 | A557 | 数据单元 2004 | F228 |
| SR | 0000 | SR | 0009 (N, C = 1) |

SUB

Wb 减 Ws

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUB{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Wb) – (Ws) → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0101 | 0www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 从基准寄存器 Wb 内容中减去源寄存器 Ws 中的内容，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUB.B W0, W1, W0 ; 从 W0 内容中减去 W1 内容（字节模式）
; 结果存入 W0

| | | | |
|-------|------|-------|------------------|
| 指令执行前 | | 指令执行后 | |
| W0 | 1732 | W0 | 17EE |
| W1 | 7844 | W1 | 7844 |
| SR | 0000 | SR | 0108 (DC, N = 1) |

例 2: SUB W7, [W8++], [W9++] ; 从 W7 内容中减去 [W8] 内容（字模式）
 ; 结果存入 [W9]
 ; 指令执行后，递增 W8
 ; 指令执行后，递增 W9

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|----------------------|
| W7 | 2450 | W7 | 2450 |
| W8 | 1808 | W8 | 180A |
| W9 | 2020 | W9 | 2022 |
| 数据单元 1808 | 92E4 | 数据单元 1808 | 92E4 |
| 数据单元 2020 | A557 | 数据单元 2020 | 916C |
| SR | 0000 | SR | 010C (DC, N, OV = 1) |

SUB 累加器相减

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | | X | X | X |

语法: { 标号 : } SUB Acc

操作数: Acc ∈ [A,B]

操作: 如果 (Acc = A):
ACCA – ACCB → ACCA
否则:
ACCB – ACCA → ACCB

受影响的状态位: OA、OB、OAB、SA、SB 和 SAB

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1100 | 1011 | A011 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 从累加器内容中减去未指定累加器中的内容，并将结果存回累加器。该指令执行 40 位减法操作。

A 位用于指定目标累加器。

指令字数: 1

指令周期数: 1

例 1: SUB A ; 从 ACCA 内容中减去 ACCB 内容
; 结果存入 ACCA
; CORCON = 0x0000 (饱和禁止)

| | 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|--------------|--------------|
| ACCA | 76 120F 098A | ACCA | 52 1EFC 4D73 | |
| ACCB | 23 F312 BC17 | ACCB | 23 F312 BC17 | |
| CORCON | 0000 | CORCON | 0000 | |
| SR | 0000 | SR | 1100 | (OA, OB = 1) |

例 2: SUB B ; 从 ACCB 内容中减去 ACCA 内容
; 结果存入 ACCB
; CORCON = 0x0040 (SATB = 1)

| | 指令执行前 | | 指令执行后 | |
|--------|--------------|--------|--------------|---------------|
| ACCA | FF 9022 2EE1 | ACCA | FF 9022 2EE1 | |
| ACCB | 00 2456 8F4C | ACCB | 00 7FFF FFFF | |
| CORCON | 0040 | CORCON | 0040 | |
| SR | 0000 | SR | 1400 | (SB, SAB = 1) |

SUBB

f 减 WREG（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBB{.B} f {,WREG}

操作数: $f \in [0 \dots 8191]$

操作: $(f) - (WREG) - (\overline{C}) \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0101 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 从指定的文件寄存器内容中减去默认工作寄存器 WREG 的内容和借位标志（进位标志的逻辑反 \overline{C} ），并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

3: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1

指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUBB.B 0x1FFF ; 从 (0x1FFF) 中减去 WREG 内容和 \overline{C} （字节模式）
; 结果存入 0x1FFF

| | 指令执行前 | 指令执行后 |
|-----------|-------|------------------|
| WREG (W0) | 7804 | 7804 |
| 数据单元 1FFE | 9439 | 8F39 |
| SR | 0000 | 0011 (DC, C = 1) |

例 2: SUBB 0xA04, WREG ; 从 (0xA04) 中减去 WREG 内容和 \overline{C} （字模式）
; 结果存入 WREG

| | 指令执行前 | 指令执行后 |
|-----------|-------|--------------|
| WREG (W0) | 6234 | 0000 |
| 数据单元 0A04 | 6235 | 6235 |
| SR | 0000 | 0001 (C = 1) |

SUBB

立即数减 Wn（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } SUBB{.B} #lit10, Wn

操作数： 对于字节操作， lit10 ∈ [0 ... 255]
对于字操作， lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作： (Wn) − lit10 − \overline{C} → Wn

受影响的状态位： DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1011 | 0001 | 1Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述： 从工作寄存器 Wn 内容中减去 10 位无符号立即数操作数和借位标志（进位标志的逻辑反 \overline{C} ），并将结果存回工作寄存器 Wn。必须使用寄存器直接寻址对 Wn 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
k 位用于指定立即数操作数。
d 位用于选择工作寄存器的地址。

- 注 1：** 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2：** 对于字节操作，立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息，请参见第 4.6 节“使用 10 位立即数操作数”。
- 3：** Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数： 1

指令周期数： 1

例 1： SUBB.B #0x23, W0 ; 从 W0 内容中减去 0x23 和 \overline{C} （字节模式）
; 结果存入 W0

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------------------|
| W0 | 7804 | W0 | 78E0 |
| SR | 0000 | SR | 0108 (DC, N = 1) |

例 2： SUBB #0x108, W4 ; 从 W4 内容中减去 0x108 和 \overline{C} （字模式）
; 结果存入 W4

| 指令执行前 | | 指令执行后 | |
|-------|--------------|-------|--------------|
| W4 | 6234 | W4 | 612C |
| SR | 0001 (C = 1) | SR | 0001 (C = 1) |

SUBB

Wb 减短立即数（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } SUBB{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数： Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]
操作： (Wb) − lit5 − (C̄) → Wd
受影响的状态位： DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 0101 | 1www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述：从基准寄存器 Wb 内容中减去 5 位无符号立即数操作数和借位标志（进位标志的逻辑反 C̄），并将结果存入目标寄存器 Wd。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

- 注 1：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2：Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数： 1
指令周期数： 1

例 1： SUBB.B W4, #0x10, W5 ; 从 w4 内容中减去 0x10 和 C̄（字节模式）
; 结果存入 w5

| | | | |
|-------|------|-------|------------------|
| 指令执行前 | | 指令执行后 | |
| W4 | 1782 | W4 | 1782 |
| W5 | 7804 | W5 | 7871 |
| SR | 0000 | SR | 0005 (OV, C = 1) |

例 2: SUBB W0, #0x8, [W2++] ; 从 W0 内容中减去 0x8 和 \bar{C} （字模式）
 ; 结果存入 [W2]
 ; 指令执行后，递增 W2

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|---------|-----------|--|------|----------------|
| W0 | | 0009 | | W0 | | 0009 | |
| W2 | | 2004 | | W2 | | 2006 | |
| 数据单元 2004 | | A557 | (Z = 1) | 数据单元 2004 | | 0000 | (DC, Z, C = 1) |
| SR | | 0002 | | SR | | 0103 | |

SUBB

Wb 减 Ws（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } SUBB{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数： Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]
操作： (Wb) – (Ws) – (\overline{C}) → Wd
受影响的状态位： DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 0101 | 1www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述：从基准寄存器 Wb 内容中减去源寄存器 Ws 的内容和借位标志（进位标志的逻辑反 \overline{C} ），并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。
w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注 1：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
2：Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数： 1
指令周期数： 1⁽¹⁾

注 1：在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1： SUBB.B W0, W1, W0 ; 从 W0 内容中减去 W1 内容和 \overline{C} （字节模式）
; 结果存入 W0

| | | | |
|-------|------|-------|------------------|
| 指令执行前 | | 指令执行后 | |
| W0 | 1732 | W0 | 17ED |
| W1 | 7844 | W1 | 7844 |
| SR | 0000 | SR | 0108 (DC, N = 1) |

例 2: SUBB W7,[W8++],[W9++] ; 从 W7 内容中减去 [W8] 内容和 \overline{C} (字模式)
 ; 结果存入 [W9]
 ; 指令执行后, 递增 W8
 ; 指令执行后, 递增 W9

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|----------------------|
| W7 | 2450 | W7 | 2450 |
| W8 | 1808 | W8 | 180A |
| W9 | 2022 | W9 | 2024 |
| 数据单元 1808 | 92E4 | 数据单元 1808 | 92E4 |
| 数据单元 2022 | A557 | 数据单元 2022 | 916B |
| SR | 0000 | SR | 010C (DC, N, OV = 1) |

SUBBR

WREG 减 f（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法： { 标号 : } SUBBR{.B} f {,WREG}

操作数： $f \in [0 \dots 8191]$

操作： $(WREG) - (f) - (\overline{C}) \rightarrow$ 由 D 指定的目标寄存器

受影响的状态位： DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1011 | 1101 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述：从 WREG 内容中减去指定文件寄存器 f 的内容和借位标志（进位标志的逻辑反 \overline{C} ），并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1：指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

2：WREG 设定为工作寄存器 W0。

3：Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数： 1

指令周期数： 1⁽¹⁾

注 1：在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1： SUBBR.B 0x803 ; 从 WREG 内容中减去 (0x803) 和 \overline{C} （字节模式）
; 结果存入 0x803

| | 指令执行前 | | 指令执行后 |
|-----------|--------------|-----------|-------|
| WREG (W0) | 7804 | WREG (W0) | 7804 |
| 数据单元 0802 | 9439 | 数据单元 0802 | 6F39 |
| SR | 0002 (Z = 1) | SR | 0000 |

例 2： SUBBR 0xA04, WREG ; 从 WREG 内容中减去 (0xA04) 和 \overline{C} （字模式）
; 结果存入 WREG

| | 指令执行前 | | 指令执行后 |
|-----------|-------|-----------|--------------|
| WREG (W0) | 6234 | WREG (W0) | FFFE |
| 数据单元 0A04 | 6235 | 数据单元 0A04 | 6235 |
| SR | 0000 | SR | 0008 (N = 1) |

SUBBR

短立即数减 Wb（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBBR{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: lit5 − (Wb) − (C̄) → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0001 | 1www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 从 5 位无符号立即数中减去基准寄存器 Wb 的内容和借位标志（进位标志的逻辑反 C̄），并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
- 2: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1
指令周期数: 1

例 1: SUBBR.B W0, #0x10, W1 ; 从 0x10 中减去 W0 内容和 C̄（字节模式）
; 结果存入 W1

| 指令执行前 | | | 指令执行后 | | |
|-------|------|------------|-------|------|----------------|
| W0 | F310 | | W0 | F310 | |
| W1 | 786A | | W1 | 7800 | |
| SR | 0003 | (Z, C = 1) | SR | 0103 | (DC, Z, C = 1) |

| 指令执行前 | | 指令执行后 | |
|-----------|--------------|-----------|------------------|
| W0 | 0009 | W0 | 0009 |
| W2 | 2004 | W2 | 2006 |
| 数据单元 2004 | A557 | 数据单元 2004 | FFFE |
| SR | 0020 (Z = 1) | SR | 0108 (DC, N = 1) |

SUBBR

Ws 减 Wb（带借位）

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBBR{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) − (Wb) − (C̄) → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0001 | 1www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 从源寄存器 Ws 内容中减去基准寄存器 Wb 的内容和借位标志（进位标志的逻辑反 C̄），并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。
2: Z 标志对于 ADDC、CPB、SUBB 和 SUBBR 指令呈现“粘住”特性，这些指令只能将其清零。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUBBR.B W0, W1, W0 ; 从 W1 内容中减去 W0 内容和 C̄（字节模式）
; 结果存入 W0

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|--------------|
| W0 | 1732 | W0 | 1711 |
| W1 | 7844 | W1 | 7844 |
| SR | 0000 | SR | 0001 (C = 1) |

例 2: SUBBR W7,[W8++],[W9++] ; 从 [W8] 内容中减去 W7 内容和 \overline{C} （字模式）
 ; 结果存入 [W9]
 ; 指令执行后，递增 W8
 ; 指令执行后，递增 W9

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------------------|
| W7 | 2450 | W7 | 2450 |
| W8 | 1808 | W8 | 180A |
| W9 | 2022 | W9 | 2024 |
| 数据单元 1808 | 92E4 | 数据单元 1808 | 92E4 |
| 数据单元 2022 | A557 | 数据单元 2022 | 6E93 |
| SR | 0000 | SR | 0005 (OV, C = 1) |

SUBR

WREG 减 f

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBR{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (WREG) – (f) → 由 D 指定的目标寄存器

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1101 | 0BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 从默认工作寄存器 WREG 内容中减去指定文件寄存器的内容，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。

f 位用于选择文件寄存器的地址。

注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

2: WREG 设定为工作寄存器 W0。

指令字数: 1

指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUBR.B 0x1FFF ; 从 WREG 内容中减去 (0x1FFF)（字节模式）
; 结果存入 0x1FFF

| | | | |
|-----------|-------|-----------|-------|
| | 指令执行前 | | 指令执行后 |
| WREG (W0) | 7804 | WREG (W0) | 7804 |
| 数据单元 1FFE | 9439 | 数据单元 1FFE | 7039 |
| SR | 0000 | SR | 0000 |

例 2: SUBR 0xA04, WREG ; 从 WREG 内容中减去 (0xA04)（字模式）
; 结果存入 WREG

| | | | |
|-----------|-------|-----------|--------------|
| | 指令执行前 | | 指令执行后 |
| WREG (W0) | 6234 | WREG (W0) | FFFF |
| 数据单元 0A04 | 6235 | 数据单元 0A04 | 6235 |
| SR | 0000 | SR | 0008 (N = 1) |

SUBR

短立即数减 Wb

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBR{.B} Wb, #lit5 Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: lit5 − (Wb) → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0001 | 0www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 从 5 位无符号立即数操作数中减去基准寄存器 Wb 的内容，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: SUBR.B W0, #0x10, W1 ; 从 0x10 中减去 W0 内容（字节模式）
; 结果存入 W1

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|---------------------|
| W0 | F310 | W0 | F310 |
| W1 | 786A | W1 | 7800 |
| SR | 0000 | SR | 0103 (DC, Z, C = 1) |

例 2: SUBR W0, #0x8, [W2++] ; 从 0x8 中减去 W0 内容（字模式）
; 结果存入 [W2]
; 指令执行后，递增 W2

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------------------|
| W0 | 0009 | W0 | 0009 |
| W2 | 2004 | W2 | 2006 |
| 数据单元 2004 | A557 | 数据单元 2004 | FFFF |
| SR | 0000 | SR | 0108 (DC, N = 1) |

SUBR

Ws 减 Wb

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SUBR{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Ws) – (Wb) → Wd

受影响的状态位: DC、N、OV、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0001 | 0www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 从源寄存器 Ws 内容中减去基准寄存器 Wb 的内容，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1(1)

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: SUBR.B W0, W1, W0 ; 从 W1 内容中减去 W0 内容（字节模式）
; 结果存入 W0

| | | | | | |
|-------|------|--|-------|------|---------|
| 指令执行前 | | | 指令执行后 | | |
| W0 | 1732 | | W0 | 1712 | (C = 1) |
| W1 | 7844 | | W1 | 7844 | |
| SR | 0000 | | SR | 0001 | |

例 2: SUBR W7, [W8++], [W9++] ; 从 [W8] 内容中减去 W7（字模式）
 ; 结果存入 [W9]
 ; 指令执行后，递增 W8
 ; 指令执行后，递增 W9

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W7 | 2450 | W7 | 2450 |
| W8 | 1808 | W8 | 180A |
| W9 | 2022 | W9 | 2024 |
| 数据单元 1808 | 92E4 | 数据单元 1808 | 92E4 |
| 数据单元 2022 | A557 | 数据单元 2022 | 6E94 |
| SR | 0000 | SR | 0005 |

(OV, C = 1)

SWAP

Wn 内容字节或半字节交换

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} SWAP{.B} Wn

操作数: Wn ∈ [W0 ... W15]

操作: 对于字节操作:
(Wn)<7:4> ↔ (Wn)<3:0>
对于字操作:
(Wn)<15:8> ↔ (Wn)<7:0>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1101 | 1B00 | 0000 | 0000 | ssss |
|-------|------|------|------|------|------|------|

描述: 交换工作寄存器 Wn 中的内容。在字模式下，交换 Wn 中的两个字节。在字节模式下，交换 Wn 中低位字节的两个半字节，而 Wn 中的高位字节将保持不变。必须使用寄存器直接寻址对 Wn 进行寻址。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。

s 位用于选择工作寄存器的地址。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。

指令字数: 1

指令周期数: 1

例 1: SWAP.B W0 ; (W0) 半字节交换

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | AB87 | W0 | AB78 |
| SR | 0000 | SR | 0000 |

例 2: SWAP W0 ; (W0) 字节交换

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W0 | 8095 | W0 | 9580 |
| SR | 0000 | SR | 0000 |

TBLRDH表读高位字

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} TBLRDH{.B} [Ws], Wd
[Ws++], [Wd]
[Ws--], [Wd++]
[++Ws], [Wd--]
[--Ws], [++Wd]
[--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
如果 (LSB(Ws) = 1)
0 → Wd
否则
程序存储单元 [(TBLPAG),(Ws)] <23:16> → Wd
对于字操作:
程序存储单元 [(TBLPAG),(Ws)] <23:16> → Wd <7:0>
0 → Wd <15:8>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1010 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 读程序存储单元高位字的内容，并将其存入目标寄存器 Wd。程序存储单元的目标字地址由 8 位表指针寄存器 TBLPAG<7:0> 与 Ws 指定的有效地址组合形成。必须使用间接寻址对 Ws 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

在字模式下，将零存入目标寄存器的高字节（由于不存在的程序存储单元），且指定程序存储器地址的第三个程序存储字节（PM<23:16>）将存入目标寄存器的低字节。

在字节模式下，源地址取决于 Ws 中的内容。如果 Ws 是非字对齐的，将在目标寄存器中存入零（由于不存在的程序存储单元）。如果 Ws 是字对齐的，指定程序存储器地址的第三个程序存储字节（PM<23:16>）将存入目标寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .W 扩展符指明是字传送，但这并不需要。

指令字数: 1
指令周期数: 2（PIC24F、PIC24H、dsPIC30F 和 dsPIC33F）
5（PIC24E 和 dsPIC33E）

例 1: TBLRDH.B [W0], [W1++] ; 读程序存储单元 (TBLPAG:[W0]) 高位字 (字节模式)
 ; 结果存入 [W1]
 ; 指令执行后, 递增 W1

| | 指令执行前 | | 指令执行后 |
|----------------|---------|----------------|---------|
| W0 | 0812 | W0 | 0812 |
| W1 | 0F71 | W1 | 0F72 |
| 数据单元 0F70 | 0944 | 数据单元 0F70 | EF44 |
| 程序存储单元 01 0812 | EF 2042 | 程序存储单元 01 0812 | EF 2042 |
| TBLPAG | 0001 | TBLPAG | 0001 |
| SR | 0000 | SR | 0000 |

例 2: TBLRDH [W6++], W8 ; 读程序存储单元 (TBLPAG:[W6]) 高位字 (字模式)
 ; 结果存入 W8
 ; 指令执行后, 递增 W6

| | 指令执行前 | | 指令执行后 |
|----------------|---------|----------------|---------|
| W6 | 3406 | W6 | 3408 |
| W8 | 65B1 | W8 | 0029 |
| 程序存储单元 00 3406 | 29 2E40 | 程序存储单元 00 3406 | 29 2E40 |
| TBLPAG | 0000 | TBLPAG | 0000 |
| SR | 0000 | SR | 0000 |

TBLRD

表读低位字

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 ;} TBLRD{.B} [Ws], Wd
[Ws++], [Wd]
[Ws--], [Wd++]
[++Ws], [Wd--]
[--Ws], [++Wd]
[--Wd]

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
如果 (LSB(Ws) = 1)
程序存储单元 [(TBLPAG), (Ws)] <15:8> → Wd
否则
程序存储单元 [(TBLPAG), (Ws)] <7:0> → Wd
对于字操作:
程序存储单元 [(TBLPAG), (Ws)] <15:0> → Wd

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1010 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 读程序存储单元低位字的内容，并将其存入目标寄存器 Wd。程序存储单元的目标字地址由 8 位表指针寄存器 TBLPAG<7:0> 与 Ws 指定的有效地址组合形成。必须使用间接寻址对 Ws 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

在字模式下，程序存储单元的两个低位字节将存入目标寄存器。在字节模式下，源地址取决于 Ws 中的内容。如果 Ws 是非字对齐的，程序存储字的第二个字节 (PM<15:7>) 将存入目标寄存器。如果 Ws 是字对齐的，程序存储字的第一个字节 (PM<7:0>) 将存入目标寄存器。

B 位用于选择字节或字操作 (0 选择字操作，1 选择字节操作)。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .w 扩展符指明是字传送，但这并不需要。

指令字数: 1
指令周期数: 2 (PIC24F、PIC24H、dsPIC30F 和 dsPIC33F)
5 (PIC24E 和 dsPIC33E)

例 1: TBLRDL.B [W0++], W1 ; 读程序存储单元(TBLPAG:[W0])低位字（字节模式）
 ; 结果存入 W1
 ; 指令执行后，递增 W0

| 指令执行前 | | 指令执行后 | |
|-------------------|---------|-------------------|---------|
| W0 | 0813 | W0 | 0814 |
| W1 | 0F71 | W1 | 0F20 |
| 数据单元 0F70 | 0944 | 数据单元 0F70 | EF44 |
| 程序存储单元 01 0812 | EF 2042 | 程序存储单元 01 0812 | EF 2042 |
| TBLPAG | 0001 | TBLPAG | 0001 |
| SR | 0000 | SR | 0000 |

| | |
|-------------|--|
| 例 2: | TBLRDL [W6], [W8++] ; 读程序存储单元 (TBLPAG:[W6]) 低位字 (字模式) ; 结果存入 W8 ; 指令执行后, 递增 W8 |
|-------------|--|

| 指令执行前 | | 指令执行后 | |
|-------------------|---------|-------------------|---------|
| W6 | 3406 | W6 | 3406 |
| W8 | 1202 | W8 | 1204 |
| 数据单元 1202 | 658B | 数据单元 1202 | 2E40 |
| 程序存储单元 00 3406 | 29 2E40 | 程序存储单元 00 3406 | 29 2E40 |
| TBLPAG | 0000 | TBLPAG | 0000 |
| SR | 0000 | SR | 0000 |

TBLWTH表写高位字

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} TBLWTH{.B} Ws, [Wd]
[Ws], [Wd++]
[Ws++], [Wd--]
[Ws--], [++Wd]
[++Ws], [--Wd]
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
如果 (LSB(Wd) = 1)
NOP
否则
(Ws) → 程序存储单元 [(TBLPAG),(Wd)]<23:16>
对于字操作:
(Ws)<7:0> → 程序存储单元 [(TBLPAG),(Wd)] <23:16>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1011 | 1Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源工作寄存器 Ws 的内容存入程序存储单元的高位字。程序存储单元的目标字地址由 8 位表指针寄存器 TBLPAG<7:0> 与 Wd 指定的有效地址组合形成。可使用直接或间接寻址对 Ws 进行寻址。必须使用间接寻址对 Wd 进行寻址。

由于程序存储器为 24 位宽，因此该指令只能写入程序存储单元的高位字节（PM<23:16>）。上述操作可在字节模式或字模式下 Wd 为字对齐时进行。如果工作于字节模式且 Wd 为非字对齐，将不执行任何操作。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注：指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .w 扩展符指明是字传送，但这并不需要。

指令字数: 1

指令周期数: 2⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: TBLWTH.B [W0++], [W1] ; 写 [W0] (字节模式)
 ; 到程序锁存单元 (TBLPAG:[W1]) 高位字
 ; 指令执行后, 递增 W0

| 指令执行前 | | 指令执行后 | |
|----------------|---------|----------------|---------|
| W0 | 0812 | W0 | 0814 |
| W1 | 0F70 | W1 | 0F70 |
| 数据单元 0812 | 0944 | 数据单元 0812 | EF44 |
| 程序存储单元 01 0F70 | EF 2042 | 程序存储单元 01 0F70 | 44 2042 |
| TBLPAG | 0001 | TBLPAG | 0001 |
| SR | 0000 | SR | 0000 |

注: 仅写入程序锁存器。在按照具体器件的系列参考手册中所描述的过程对闪存程序存储器进行编程前, 程序存储器中的内容不会被更新。

例 2: TBLWTH W6, [W8++] ; 写 W6 (字模式)
 ; 到程序锁存单元 (TBLPAG:[W8]) 高位字
 ; 指令执行后, 递增 W8

| 指令执行前 | | 指令执行后 | |
|----------------|---------|----------------|---------|
| W6 | 0026 | W6 | 0026 |
| W8 | 0870 | W8 | 0872 |
| 程序存储单元 00 0870 | 22 3551 | 程序存储单元 00 0870 | 26 3551 |
| TBLPAG | 0000 | TBLPAG | 0000 |
| SR | 0000 | SR | 0000 |

注: 仅写入程序锁存器。在按照具体器件的系列参考手册中所描述的过程对闪存程序存储器进行编程前, 程序存储器中的内容不会被更新。

TBLWTL 表写低位字

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} TBLWTL{.B} Ws, [Wd]
[Ws], [Wd++]
[Ws++], [Wd--]
[Ws--], [++Wd]
[++Ws], [--Wd]
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: 对于字节操作:
如果 (LSB(Wd) = 1)
(Ws) → 程序存储单元 [(TBLPAG),(Wd)] <15:8>
否则
(Ws) → 程序存储单元 [(TBLPAG),(Wd)] <7:0>
对于字操作:
(Ws) → 程序存储单元 [(TBLPAG),(Wd)] <15:0>

受影响的状态位: 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 1011 | 0Bqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源工作寄存器 Ws 的内容存入程序存储单元的低位字。程序存储单元的目标字地址由 8 位表指针寄存器 TBLPAG<7:0> 与 Wd 指定的有效地址组合形成。可使用直接或间接寻址对 Ws 进行寻址。必须使用间接寻址对 Wd 进行寻址。

在字模式下，Ws 中内容将存入程序存储单元的两个低位字节。在字节模式下，Wd 中的最低位用于确定目标字节。如果 Wd 是非字对齐的，Ws 中内容将被存入程序存储单元的第二个字节 (PM<15:8>)。如果 Wd 是字对齐的，Ws 中内容将被存入程序存储单元的第一个字节 (PM<7:0>)。

B 位用于选择字节或字操作 (0 选择字操作，1 选择字节操作)。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节传送而非字传送。用户可使用 .W 扩展符指明是字传送，但这并不需要。

指令字数: 1

指令周期数: 2⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: TBLWTL.B W0, [W1++] ; 写 W0（字节模式）
 ; 到程序锁存单元 (TBLPAG:[W1]) 低位字
 ; 指令执行后，递增 W1

| 指令执行前 | | 指令执行后 | |
|-------------------|---------|-------------------|---------|
| W0 | 6628 | W0 | 6628 |
| W1 | 1225 | W1 | 1226 |
| 程序存储单元 00 1224 | 78 0080 | 程序存储单元 01 1224 | 78 2880 |
| TBLPAG | 0000 | TBLPAG | 0000 |
| SR | 0000 | SR | 0000 |

注： 仅写入程序锁存器。在按照具体器件的系列参考手册中所描述的过程对闪存程序存储器进行编程前，程序存储器中的内容不会被更新。

例 2: TBLWTL [W6], [W8] ; 写 [W6]（字模式）
 ; 到程序锁存单元 (TBLPAG:[W8]) 低位字
 ; 指令执行后，递增 W8

| 指令执行前 | | 指令执行后 | |
|-------------------|---------|-------------------|---------|
| W6 | 1600 | W6 | 1600 |
| W8 | 7208 | W8 | 7208 |
| 数据单元 1600 | 0130 | 数据单元 1600 | 0130 |
| 程序存储单元 01 7208 | 09 0002 | 程序存储单元 01 7208 | 09 0130 |
| TBLPAG | 0001 | TBLPAG | 0001 |
| SR | 0000 | SR | 0000 |

注： 仅写入程序锁存器。在按照具体器件的系列参考手册中所描述的过程对闪存程序存储器进行编程前，程序存储器中的内容不会被更新。

ULNK

释放堆栈帧

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于： | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | | X | X | |

语法： { 标号 ; } ULNK

操作数： 无

操作： W14 → W15
(W15) – 2 → W15
(TOS) → W14

受影响的状态位： 无

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码： | 1111 | 1010 | 1000 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述： 该指令释放为子程序调用序列分配的堆栈帧。堆栈帧释放是通过将堆栈指针（W15）设置为等于帧指针（W14），随后弹出堆栈以使帧指针（W14）复位来完成的。

指令字数： 1

指令周期数： 1

例 1： ULNK ; 释放堆栈帧

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W14 | 2002 | W14 | 2000 |
| W15 | 20A2 | W15 | 2000 |
| 数据单元 2000 | 2000 | 数据单元 2000 | 2000 |
| SR | 0000 | SR | 0000 |

例 2： ULNK ; 释放堆栈帧

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|------|
| W14 | 0802 | W14 | 0800 |
| W15 | 0812 | W15 | 0800 |
| 数据单元 0800 | 0800 | 数据单元 0800 | 0800 |
| SR | 0000 | SR | 0000 |

ULNK

释放堆栈帧

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | | | X | | | X |

语法: { 标号 ;} ULNK

操作数: 无
操作: W14 → W15
(W15) – 2 → W15
(TOS) → W14
0 → SFA 位

受影响的状态位: SFA

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1010 | 1000 | 0000 | 0000 | 0000 |
|-------|------|------|------|------|------|------|

描述: 该指令释放为子程序调用序列分配的堆栈帧。堆栈帧释放是通过将堆栈指针 (W15) 设置为等于帧指针 (W14)，随后弹出堆栈以使帧指针 (W14) 复位来完成的。

指令字数: 1
指令周期数: 1

例 1: ULNK ; 释放堆栈帧

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|-----------|-------|------|--|
| W14 | 2002 | | W14 | 2000 | |
| W15 | 20A2 | | W15 | 2000 | |
| 数据单元 2000 | 2000 | 数据单元 2000 | 2000 | | |
| SR | 0000 | SR | 0000 | | |

例 2: ULNK ; 释放堆栈帧

| 指令执行前 | | | 指令执行后 | | |
|-----------|------|-----------|-------|------|--|
| W14 | 0802 | | W14 | 0800 | |
| W15 | 0812 | | W15 | 0800 | |
| 数据单元 0800 | 0800 | 数据单元 0800 | 0800 | | |
| SR | 0000 | SR | 0000 | | |

XOR

f 和 WREG 逻辑异或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} XOR{.B} f {,WREG}

操作数: f ∈ [0 ... 8191]

操作: (f).XOR.(WREG) → 由 D 指定的目标寄存器

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0110 | 1BDf | ffff | ffff | ffff |
|-------|------|------|------|------|------|------|

描述: 将默认工作寄存器 WREG 中的内容和指定文件寄存器中的内容进行逻辑异或运算，并将结果存放在目标寄存器中。可选的 WREG 操作数用于确定目标寄存器。如果指定了 WREG，则结果将存入 WREG。如果未指定 WREG，则结果将存入文件寄存器。

B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
D 位用于选择目标寄存器（0 选择 WREG，1 选择文件寄存器）。
f 位用于选择文件寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .W 扩展符指明是字操作，但这并不需要。
2: WREG 设定为工作寄存器 W0。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: XOR.B 0x1FFF ; (0x1FFF) 和 WREG 异或（字节模式）
; 结果存入 0x1FFF

| | 指令执行前 | 指令执行后 |
|-----------|-------|--------------|
| WREG (W0) | 7804 | 7804 |
| 数据单元 1FFE | 9439 | 9039 |
| SR | 0000 | 0008 (N = 1) |

例 2: XOR 0xA04, WREG ; (0xA04) 和 WREG 异或（字模式）
; 结果存入 WREG

| | 指令执行前 | 指令执行后 |
|-----------|-------|--------------|
| WREG (W0) | 6234 | C267 |
| 数据单元 0A04 | A053 | A053 |
| SR | 0000 | 0008 (N = 1) |

XOR

立即数和 Wn 逻辑异或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } XOR{.B} #lit10, Wn

操作数: 对于字节操作, lit10 ∈ [0 ... 255]
对于字操作, lit10 ∈ [0 ... 1023]
Wn ∈ [W0 ... W15]

操作: lit10.XOR.(Wn) → Wn

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1011 | 0010 | 1Bkk | kkkk | kkkk | dddd |
|-------|------|------|------|------|------|------|

描述: 将 10 位无符号立即数操作数和工作寄存器 Wn 中的内容进行逻辑异或运算, 并将结果存回工作寄存器 Wn。必须使用寄存器直接寻址对 Wn 进行寻址。

B 位用于选择字节或字操作 (0 选择字操作, 1 选择字节操作)。

k 位用于指定立即数操作数。

d 位用于选择工作寄存器的地址。

- 注 1: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作, 但这并不需要。
- 2: 对于字节操作, 立即数必须指定为 [0:255] 中的无符号值。有关在字节模式下使用 10 位立即数操作数的信息, 请参见第 4.6 节“使用 10 位立即数操作数”。

指令字数: 1
指令周期数: 1

例 1: XOR.B #0x23, W0 ; 0x23 和 W0 异或 (字节模式)
; 结果存入 W0

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W0 | 7804 | W0 | 7827 |
| SR | 0000 | SR | 0000 |

例 2: XOR #0x108, W4 ; 0x108 和 W4 异或 (字模式)
; 结果存入 W4

| | | | |
|-------|------|-------|------|
| 指令执行前 | | 指令执行后 | |
| W4 | 6134 | W4 | 603C |
| SR | 0000 | SR | 0000 |

XOR

Wb 和短立即数逻辑异或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } XOR{.B} Wb, #lit5, Wd
[Wd]
[Wd++]
[Wd--]
[++Wd]
[--Wd]

操作数: Wb ∈ [W0 ... W15]
lit5 ∈ [0 ... 31]
Wd ∈ [W0 ... W15]

操作: (Wb).XOR.lit5 → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0110 | 1www | wBqq | qddd | d11k | kkkk |
|-------|------|------|------|------|------|------|

描述: 将基准寄存器Wb中的内容和5位无符号立即数操作数进行逻辑异或运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
k 位用于提供一个 5 位整数的立即数操作数。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1

例 1: XOR.B W4, #0x14, W5 ; W4 和 0x14 异或（字节模式）
; 结果存入 W5

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W4 | C822 | W4 | C822 |
| W5 | 1200 | W5 | 1234 |
| SR | 0000 | SR | 0000 |

例 2: XOR W2, #0x1F, [W8++] ; W2 和 0x1F 异或（字模式）
; 结果存入 [W8]
; 指令执行后，递增 W8

| 指令执行前 | | 指令执行后 | |
|-----------|------|-----------|--------------|
| W2 | 8505 | W2 | 8505 |
| W8 | 1004 | W8 | 1006 |
| 数据单元 1004 | 6628 | 数据单元 1004 | 851A |
| SR | 0000 | SR | 0008 (N = 1) |

XOR

Wb 和 Ws 逻辑异或

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 : } XOR{.B} Wb, Ws, Wd
[Ws], [Wd]
[Ws++], [Wd++]
[Ws--], [Wd--]
[++Ws], [++Wd]
[--Ws], [--Wd]

操作数: Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

操作: (Wb).XOR.(Ws) → Wd

受影响的状态位: N 和 Z

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 0110 | 1www | wBqq | qddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源寄存器 Ws 中的内容和基准寄存器 Wb 中的内容进行逻辑异或运算，并将结果存放在目标寄存器 Wd 中。必须使用寄存器直接寻址对 Wb 进行寻址。可使用寄存器直接或间接寻址对 Ws 和 Wd 进行寻址。

w 位用于选择基准寄存器的地址。
B 位用于选择字节或字操作（0 选择字操作，1 选择字节操作）。
q 位用于选择目标地址模式。
d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

注: 指令中的扩展符 .B 指明是字节操作而非字操作。用户可使用 .w 扩展符指明是字操作，但这并不需要。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

ZE

零扩展 Ws

| | | | | | | |
|------|--------|--------|--------|----------|----------|----------|
| 实现于: | PIC24F | PIC24H | PIC24E | dsPIC30F | dsPIC33F | dsPIC33E |
| | X | X | X | X | X | X |

语法: { 标号 :} ZE Ws, Wnd
[Ws],
[Ws++],
[Ws--],
[++Ws],
[--Ws],

操作数: Ws ∈ [W0 ... W15]
Wnd ∈ [W0 ... W15]
操作: Ws<7:0> → Wnd<7:0>
0 → Wnd<15:8>

受影响的状态位: N、Z 和 C

| | | | | | | |
|-------|------|------|------|------|------|------|
| 指令编码: | 1111 | 1011 | 1000 | 0ddd | dppp | ssss |
|-------|------|------|------|------|------|------|

描述: 将源工作寄存器 Ws 中的低位字节零扩展为 16 位值，并将结果存入目标工作寄存器 Wnd。可使用寄存器直接或间接寻址对 Ws 进行寻址。必须使用寄存器直接寻址对 Wnd 进行寻址。由于零扩展字总是为正，因此 N 标志将被清零而 C 标志将置 1。

d 位用于选择目标寄存器。
p 位用于选择源地址模式。
s 位用于选择源寄存器。

- 注 1: 该操作将字节转换为字，且不使用 .B 或 .W 扩展符。
2: 源寄存器 Ws 将作为字节操作数进行寻址，因此所有地址改变都以 1 为单位。

指令字数: 1
指令周期数: 1⁽¹⁾

注 1: 在 dsPIC33E 和 PIC24E 器件中，列出的周期数不适用于对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作。更多详细信息，请参见第 3.2.1 节“多周期指令”中的注 3。

例 1: ZE W3, W4 ; 零扩展 W3
; 结果存入 W4

| 指令执行前 | | 指令执行后 | |
|-------|------|-------|------|
| W3 | 7839 | W3 | 7839 |
| W4 | 1005 | W4 | 0039 |
| SR | 0000 | SR | 0001 |

(C = 1)

例 2: ZE [W2++], W12 ; 零扩展 [W2]
 ; 结果存入 W12
 ; 指令执行后, 递增 W2

| 指令执行前 | | | | 指令执行后 | | | |
|-----------|--|------|--|-----------|--|------|---------|
| W2 | | 0900 | | W2 | | 0901 | |
| W12 | | 1002 | | W12 | | 008F | |
| 数据单元 0900 | | 268F | | 数据单元 0900 | | 268F | |
| SR | | 0000 | | SR | | 0001 | (C = 1) |

注:

第 6 章 内建函数

目录

本章包括以下主要内容：

| | | |
|-----|--------------|-----|
| 6.1 | 简介 | 446 |
| 6.2 | 内建函数列表 | 447 |

6.1 简介

本章介绍特定于 MPLAB XC16 C 编译器的内建函数。

内建函数使 C 语言程序员可以访问目前只能通过行内汇编访问但却十分有用、适用于广泛应用程序的汇编操作符或机器指令。内建函数在 C 源文件中编写，语法类似于函数调用，但它们被编译为直接实现函数的汇编代码，不涉及函数调用或库程序。

有一些原因使得提供内建函数优于要求程序员使用行内汇编。这些原因包括：

1. 提供用于特定目的的内建函数可以简化编码。
2. 使用行内汇编时，某些优化会被禁止。内建函数则不会如此。
3. 对于使用专用寄存器的机器指令，在编写行内汇编代码时，需要特别细心处理，以避免寄存器分配错误。内建函数让这个变得简单，因为您不需要关心每条机器指令的具体寄存器要求。

下面列出了这些内建函数，后面各小节将对各个内建函数进行详细说明。

- | | |
|---------------------------------------|---|
| • <code>__builtin_addab</code> | • <code>__builtin_mpy</code> |
| • <code>__builtin_add</code> | • <code>__builtin_msc</code> |
| • <code>__builtin_btg</code> | • <code>__builtin_mulss</code> |
| • <code>__builtin_clr</code> | • <code>__builtin_mulsu</code> |
| • <code>__builtin_clr_prefetch</code> | • <code>__builtin_mulus</code> |
| • <code>__builtin_divf</code> | • <code>__builtin_muluu</code> |
| • <code>__builtin_divmodsd</code> | • <code>__builtin_nop</code> |
| • <code>__builtin_divmodud</code> | • <code>__builtin_psvpage</code> |
| • <code>__builtin_divsd</code> | • <code>__builtin_psvoffset</code> |
| • <code>__builtin_divud</code> | • <code>__builtin_readsfr</code> |
| • <code>__builtin_dmaoffset</code> | • <code>__builtin_return_address</code> |
| • <code>__builtin_ed</code> | • <code>__builtin_sac</code> |
| • <code>__builtin_edac</code> | • <code>__builtin_sacr</code> |
| • <code>__builtin_edsoffset</code> | • <code>__builtin_sftac</code> |
| • <code>__builtin_edspace</code> | • <code>__builtin_subab</code> |
| • <code>__builtin_fbcl</code> | • <code>__builtin_tbladdress</code> |
| • <code>__builtin_lac</code> | • <code>__builtin_tblpage</code> |
| • <code>__builtin_mac</code> | • <code>__builtin_tbloffset</code> |
| • <code>__builtin_modsd</code> | • <code>__builtin_tblrdrh</code> |
| • <code>__builtin_modud</code> | • <code>__builtin_tblrldl</code> |
| • <code>__builtin_movsac</code> | • <code>__builtin_tblwth</code> |
| • <code>__builtin_mpy</code> | • <code>__builtin_tblwtl</code> |

本章仅介绍与 CPU 操作相关的内建函数。编译器提供用于写闪存程序存储器和更改振荡器设置等操作的其他内建函数。请参见《MPLAB[®] XC16 C 编译器用户指南》(DS50002071E_CN)了解所有编译器内建函数。

6.2 内建函数列表

本节介绍编译器内建函数的程序员接口。由于这些函数是“内建”的，所以不存在与之关联的头文件。类似地，不存在与内建函数关联的命令行开关——它们总是可用。所选择的内建函数名称使它们属于编译器的命名空间（均具有前缀 `__builtin_`），所以它们不会与程序员命名空间中的函数或变量名称发生冲突。

`__builtin_addab`

说明：

将累加器 A 和 B 相加，并将结果回写到指定的累加器。例如：

```
register int result asm("A");
register int B asm("A");
```

```
result = __builtin_addab(result,B);
```

将生成：

```
add A
```

原型：

```
int __builtin_addab(int Accum_a, int Accum_b);
```

参数：

Accum_a 要相加的第一个累加器。

Accum_b 要相加的第二个累加器。

返回值：

将加法结果返回到累加器。

汇编操作符 / 机器指令：

```
add
```

错误消息：

如果结果不是累加器寄存器，则将显示一条错误消息。

__builtin_add

说明:

将 `value` 与 `result` 指定的累加器相加，并按立即数指定的移位量进行移位。例如：

```
register int result asm("A");
int value;
result = __builtin_add(result,value,0);
```

如果 `value` 存放在 `w0` 中，将生成以下代码：

```
add w0, #0, A
```

原型:

```
int __builtin_add(int Accum,int value,
const int shift);
```

参数:

| | |
|--------------------|---------------|
| <code>Accum</code> | 要相加的累加器。 |
| <code>value</code> | 要与累加器值相加的整型数。 |
| <code>shift</code> | 对得到的累加器值的移位量。 |

返回值:

将移位后的加法结果返回到累加器。

汇编操作符 / 机器指令:

`add`

错误消息:

如果出现以下情况，将显示一条错误消息：

- 结果不是累加器寄存器
- 参数 0 不是累加器
- 移位值不是处于范围内的立即数

__builtin_btg

说明:

该函数将生成一条 btg 机器指令。一些示例包括:

```
int i;    /* near by default */
int l __attribute__((far));
```

```
struct foo {
    int bit1:1;
} barbits;
```

```
int bar;
```

```
void some_bittoggles() {
    register int j asm("w9");
    int k;
```

```
    k = i;
```

```
    __builtin_btg(&i,1);
    __builtin_btg(&j,3);
    __builtin_btg(&k,4);
    __builtin_btg(&l,11);
```

```
    return j+k;
}
```

请注意, 获取位于寄存器中的变量的地址时, 编译器会产生警告, 并导致寄存器被保存到堆栈中 (从而可以获取它的地址); 建议不要使用这种形式。该警告仅适用于由编程人员显式存放在寄存器中的变量。

原型:

```
void __builtin_btg(unsigned int *, unsigned int 0xn);
```

参数:

* 指向要翻转其中某个位的数据项的指针。
0xn 介于 0 到 15 范围内的立即数。

返回值:

返回一条 btg 机器指令。

汇编操作符 / 机器指令:

btg

错误消息:

如果参数值不处于范围内, 将显示一条错误消息。

__builtin_clr

说明:

清零指定的累加器。例如:

```
register int result asm("A");  
result = __builtin_clr();
```

将生成:

```
clr A
```

原型:

```
int __builtin_clr(void);
```

参数:

无

返回值:

将清零值结果返回到累加器。

汇编操作符 / 机器指令:

```
clr
```

错误消息:

如果结果不是累加器寄存器, 则将显示一条错误消息。

__builtin_clr_prefetch**说明:**

清零累加器并为之后的 MAC 操作预取数据。

xptr 可以为空，表示不需要执行 X 预取，这种情况下 *xincr* 和 *xval* 的值会被忽略，但它们是必需的。

yptr 可以为空，表示不需要执行 Y 预取，这种情况下 *yincr* 和 *yval* 的值会被忽略，但它们是必需的。

xval 和 *yval* 指定 C 变量的地址，预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

如果 *AWB* 非空，则另一个累加器会被回写到所引用的变量中。

例如:

```
register int result asm("A");
register int B asm("B");
int x_memory_buffer[256]
__attribute__((space(xmemory)));
int y_memory_buffer[256]
__attribute__((space(ymemory)));
int *xmemory;
int *ymemory;
int awb;
int xVal, yVal;

xmemory = x_memory_buffer;
ymemory = y_memory_buffer;
result = __builtin_clr(&xmemory, &xVal, 2,
                      &ymemory, &yVal, 2, &awb, B);
```

可能生成:

```
clr A, [w8]+=2, w4, [w10]+=2, w5, w13
```

编译器可能需要溢出（spill）w13，以确保可以对它进行回写。建议用户将该寄存器声明为用于该目的。

在该指令之后:

- 结果将被清零
- xVal 将包含 x_memory_buffer[0]
- yVal 将包含 y_memory_buffer[0]
- xmemory 和 ymemory 将递增 2，准备好进行下一个 mac 操作

原型:

```
int __builtin_clr_prefetch(
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr, int *AWB,
int AWB_accum);
```

`_builtin_clr_prefetch` (续)

参数:

`xptr` 指向 `x` 预取的整型指针。
`xval` `x` 预取的整型值。
`xincr` `x` 预取的整型递增值。
`yptr` 指向 `y` 预取的整型指针。
`yval` `y` 预取的整型值。
`yincr` `y` 预取的整型递增值。
`AWB` 累加器回写目标。
`AWB_accum` 要被回写的累加器。

| |
|---|
| 注: 参数 <code>xptr</code> 和 <code>yptr</code> 必须分别指向位于 <code>x</code> 数据存储区和 <code>y</code> 数据存储区中的数组。 |
|---|

返回值:

将清零值结果返回到累加器。

汇编操作符 / 机器指令:

`clr`

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- `xval` 是一个空值，但 `xptr` 不为空
- `yval` 是一个空值，但 `yptr` 不为空
- `AWB_accum` 不是累加器，并且 `AWB` 不为空

__builtin_divf

说明:

计算商 num/den 。如果 den 为零，会发生数学错误异常。函数的参数和结果都是无符号的。

原型:

```
unsigned int __builtin_divf(unsigned int num,  
unsigned int den);
```

参数:

num 分子
den 分母

返回值:

返回商 num/den 的无符号整型值。

汇编操作符 / 机器指令:

`div.f`

__builtin_divmodsd

说明:

发出 16 位架构的本机有符号除法支持。值得注意的是，如果商无法装入 16 位结果中，结果（包括余数）将是非预期的。这种形式的内建函数会同时捕捉商和余数。

原型:

```
signed int __builtin_divmodsd(  
signed long dividend, signed int divisor,  
signed int *remainder);
```

参数:

dividend 被除数
divisor 除数
remainder 指向余数的指针

返回值:

商和余数。

汇编操作符 / 机器指令:

`divmodsd`

错误消息:

无。

`__builtin_divmodud`

说明:

发出 16 位架构的本机无符号除法支持。值得注意的是, 如果商无法装入 16 位结果中, 结果 (包括余数) 将是非预期的。这种形式的内建函数会同时捕捉商和余数。

原型:

```
unsigned int __builtin_divmodud(  
    unsigned long dividend, unsigned int divisor,  
    unsigned int *remainder);
```

参数:

dividend 被除数
divisor 除数
remainder 指向余数的指针

返回值:

商和余数。

汇编操作符 / 机器指令:

`divmodud`

错误消息:

无。

`__builtin_divsd`

说明:

计算商 num/den 。如果 den 为零, 会发生数学错误异常。函数的参数和结果都是有符号的。可以使用命令行选项 `-wconversions` 来检测意外的符号转换。

原型:

```
int __builtin_divsd(const long num, const int den);
```

参数:

num 分子
den 分母

返回值:

返回商 num/den 的有符号整型值。

汇编操作符 / 机器指令:

`div.sd`

__builtin_divud

说明:

计算商 num/den 。如果 den 为零，会发生数学错误异常。函数的参数和结果都是无符号的。可以使用命令行选项 `-Wconversions` 来检测意外的符号转换。

原型:

```
unsigned int __builtin_divud(const unsigned
long num, const unsigned int den);
```

参数:

num 分子
den 分母

返回值:

返回商 num/den 的无符号整型值。

汇编操作符 / 机器指令:

`div.ud`

__builtin_dmaoffset

说明:

获取 DMA 存储区中的某个符号的偏移量。

例如:

```
unsigned int result;
char buffer[256] __attribute__((space(dma)));
```

```
result = __builtin_dmaoffset(&buffer);
```

可能生成:

```
mov #dmaoffset(buffer), w0
```

原型:

```
unsigned int __builtin_dmaoffset(const void *p);
```

参数:

**p* 指向 DMA 地址值的指针

返回值:

返回位于 DMA 存储区中的某个变量的偏移量。

汇编操作符 / 机器指令:

`dmaoffset`

错误消息:

如果参数不是全局符号的地址，将显示一条错误消息。

__builtin_ed

说明:

求平方值 *sqr*，并将它作为结果返回。此外，为之后的平方运算预取数据，方法是计算 ***xptr - **yptr*，并将结果存储在 **distance* 中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

例如:

```
register int result asm("A");
int *xmemory, *ymemory;
int distance;

result = __builtin_ed(distance,
                      &xmemory, 2,
                      &ymemory, 2,
                      &distance);
```

可能生成:

```
ed w4*w4, A, [w8]+=2, [W10]+=2, w4
```

原型:

```
int __builtin_ed(int sqr, int **xptr, int xincr,
int **yptr, int yincr, int *distance);
```

参数:

sqr 整型平方值。
xptr 指向 *x* 预取的指针的整型指针。
xincr *x* 预取的整型递增值。
yptr 指向 *y* 预取的指针的整型指针。
yincr *y* 预取的整型递增值。
distance 指向 *distance* 的整型指针。

| |
|---|
| 注: 参数 <i>xptr</i> 和 <i>yptr</i> 必须分别指向位于 <i>x</i> 数据存储区和 <i>y</i> 数据存储区中的数组。 |
|---|

返回值:

将平方结果返回到累加器。

汇编操作符 / 机器指令:

ed

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- *xptr* 为空
- *yptr* 为空
- *distance* 为空

__builtin_edac**说明:**

求平方值 *sqr*，并与指定的累加器寄存器相加，将它作为结果返回。此外，为之后的平方运算预取数据，方法是计算 ***xptr - **yptr*，并将结果存储在 **distance* 中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

例如：

```
register int result asm("A");
int *xmemory, *ymemory;
int distance;

result = __builtin_edac(result, distance,
                        &xmemory, 2,
                        &ymemory, 2,
                        &distance);
```

可能生成：

```
edac w4*w4, A, [w8]+=2, [W10]+=2, w4
```

原型:

```
int __builtin_edac(int Accum, int sqr,
int **xptr, int xincr, int **yptr, int yincr,
int *distance);
```

参数:

| | |
|-----------------|---------------------------|
| <i>Accum</i> | 要相加的累加器。 |
| <i>sqr</i> | 整型平方值。 |
| <i>xptr</i> | 指向 <i>x</i> 预取的指针的整型指针。 |
| <i>xincr</i> | <i>x</i> 预取的整型递增值。 |
| <i>yptr</i> | 指向 <i>y</i> 预取的指针的整型指针。 |
| <i>yincr</i> | <i>y</i> 预取的整型递增值。 |
| <i>distance</i> | 指向 <i>distance</i> 的整型指针。 |

注： 参数 *xptr* 和 *yptr* 必须分别指向位于 *x* 数据存储区和 *y* 数据存储区中的数组。

返回值:

将平方结果返回到指定累加器。

汇编操作符 / 机器指令:

edac

错误消息:

如果出现以下情况，将显示一条错误消息：

- 结果不是累加器寄存器
- *Accum* 不是累加器寄存器
- *xptr* 为空
- *yptr* 为空
- *distance* 为空

`__builtin_edsoffset`

说明:

返回地址作为参数给出的对象的 **eds** 页偏移量。参数 *p* 必须是位于扩展数据空间中的对象的地址；否则将产生错误消息，并且编译会失败。请参见《MPLAB[®] XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 **space** 属性。

原型:

```
unsigned int __builtin_edsoffset(int *p);
```

参数:

p 对象的地址

返回值:

返回地址作为参数给出的对象的 **eds** 页偏移量。

汇编操作符 / 机器指令:

`edsoffset`

`__builtin_edspage`

说明:

返回地址作为参数给出的对象的 **eds** 页号。参数 *p* 必须是位于扩展数据空间中的对象的地址；否则将产生错误消息，并且编译会失败。请参见《MPLAB[®] XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 **space** 属性。

原型:

```
unsigned int __builtin_edspage(int *p);
```

参数:

p 对象的地址

返回值:

返回地址作为参数给出的对象的 **eds** 页号。

汇编操作符 / 机器指令:

`edspage`

__builtin_fbcl

说明:

在 *value* 中从左边开始查找第一个位变化。它对于定点数据的动态比例换算很有用。

例如:

```
int result, value;  
result = __builtin_fbcl(value);
```

可能生成:

```
fbcl w4, w5
```

原型:

```
int __builtin_fbcl(int value);
```

参数:

value 第一个有位变化的整型数。

返回值:

返回一个进行符号扩展的立即数，表示要左移的位数。

汇编操作符 / 机器指令:

```
fbcl
```

错误消息:

如果结果不是累加器寄存器，将显示一条错误消息。

__builtin_lac

说明:

将值移位 *shift*（介于 -8 和 7 之间的立即数）位，并返回要存储到累加器寄存器中的值。

例如:

```
register int result asm("A");  
int value;  
result = __builtin_lac(value,3);
```

可能生成:

```
lac w4, #3, A
```

原型:

```
int __builtin_lac(int value, int shift);
```

参数:

value 要进行移位的整型数。

shift 表示移位量的立即数。

返回值:

将移位后的结果返回到累加器。

汇编操作符 / 机器指令:

```
lac
```

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- 移位值不是处于范围内的立即数

__builtin_mac

说明:

计算 $a \times b$ ，并与累加器相加；此外，为之后的 MAC 操作预取数据。

xptr 可以为空，表示不需要执行 X 预取，这种情况下 *xincr* 和 *xval* 的值会被忽略，但它们是必需的。

yptr 可以为空，表示不需要执行 Y 预取，这种情况下 *yincr* 和 *yval* 的值会被忽略，但它们是必需的。

xval 和 *yval* 指定 C 变量的地址，预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

如果 *AWB* 非空，则另一个累加器会被回写到所引用的变量中。

例如:

```
register int result asm("A");
register int B asm("B");
int *xmemory;
int *ymemory;
int xVal, yVal;

result = __builtin_mac(result, xVal, yVal,
                        &xmemory, &xVal, 2,
                        &ymemory, &yVal, 2, 0, B);
```

可能生成:

```
mac w4*w5, A, [w8]+=2, w4, [w10]+=2, w5
```

原型:

```
int __builtin_mac(int Accum, int a, int b,
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr, int *AWB,
int AWB_accum);
```

参数:

| | |
|-----------|------------------|
| Accum | 要相加的累加器。 |
| a | 整型被乘数。 |
| b | 整型乘数。 |
| xptr | 指向 x 预取的指针的整型指针。 |
| xval | 指向 x 预取的值的整型指针。 |
| xincr | x 预取的整型递增值。 |
| yptr | 指向 y 预取的指针的整型指针。 |
| yval | 指向 y 预取的值的整型指针。 |
| yincr | y 预取的整型递增值。 |
| AWB | 累加器回写目标。 |
| AWB_accum | 要被回写的累加器。 |

注: 参数 *xptr* 和 *yptr* 必须分别指向位于 x 数据存储区和 y 数据存储区中的数组。

返回值:

返回累加器加上 $a \times b$ 的结果得到的值。

汇编操作符 / 机器指令:

mac

`__builtin_mac` (续)

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- *Accum* 不是累加器寄存器
- *xval* 是一个空值，但 *xptr* 不为空
- *yval* 是一个空值，但 *yptr* 不为空
- *AWB_accum* 不是累加器寄存器，并且 *AWB* 不为空

`__builtin_modsd`

说明:

发出 16 位架构的本机有符号除法支持。值得注意的是，如果商无法装入 16 位结果中，结果（包括余数）将是非预期的。这种形式的内建函数只会捕捉余数。

原型:

```
signed int __builtin_modsd(signed long dividend,  
signed int divisor);
```

参数:

dividend 被除数
divisor 除数

返回值:

余数。

汇编操作符 / 机器指令:

`modsd`

错误消息:

无。

`__builtin_modud`

说明:

发出 16 位架构的本机无符号除法支持。值得注意的是，如果商无法装入 16 位结果中，结果（包括余数）将是非预期的。这种形式的内建函数只会捕捉余数。

原型:

```
unsigned int __builtin_modud(unsigned long dividend,  
unsigned int divisor);
```

参数:

dividend 被除数
divisor 除数

返回值:

余数。

汇编操作符 / 机器指令:

`modud`

错误消息:

无。

__builtin_movsac**说明:**

不进行任何计算，但为之后的 MAC 操作预取数据。

xptr 可以为空，表示不需要执行 X 预取，这种情况下 *xincr* 和 *xval* 的值会被忽略，但它们是必需的。

yptr 可以为空，表示不需要执行 Y 预取，这种情况下 *yincr* 和 *yval* 的值会被忽略，但它们是必需的。

xval 和 *yval* 指定 C 变量的地址，预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

如果 *AWB* 非空，则另一个累加器会被回写到所引用的变量中。

例如：

```
register int result asm("A");
int *xmemory;
int *ymemory;
int xVal, yVal;

result = __builtin_movsac(&xmemory, &xVal, 2,
                        &ymemory, &yVal, 2, 0, 0);
```

可能生成：

```
movsac A, [w8]+=2, w4, [w10]+=2, w5
```

原型:

```
int __builtin_movsac(
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr, int *AWB
int AWB_accum);
```

参数:

xptr 指向 x 预取的指针的整型指针。
xval 指向 x 预取的值的整型指针。
xincr x 预取的整型递增值。
yptr 指向 y 预取的指针的整型指针。
yval 指向 y 预取的值的整型指针。
yincr y 预取的整型递增值。
AWB 累加器回写目标。
AWB_accum 要被回写的累加器。

注： 参数 *xptr* 和 *yptr* 必须分别指向位于 x 数据存储区和 y 数据存储区中的数组。

返回值:

返回预取数据。

汇编操作符 / 机器指令:

```
movsac
```

错误消息:

如果出现以下情况，将显示一条错误消息：

- 结果不是累加器寄存器
- *xval* 是一个空值，但 *xptr* 不为空
- *yval* 是一个空值，但 *yptr* 不为空
- *AWB_accum* 不是累加器寄存器，并且 *AWB* 不为空

__builtin_mpy

说明:

计算 $a \times b$; 此外, 为之后的 MAC 操作预取数据。

xptr 可以为空, 表示不需要执行 X 预取, 这种情况下 *xincr* 和 *xval* 的值会被忽略, 但它们是必需的。

yptr 可以为空, 表示不需要执行 Y 预取, 这种情况下 *yincr* 和 *yval* 的值会被忽略, 但它们是必需的。

xval 和 *yval* 指定 C 变量的地址, 预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数: -6、-4、-2、0、2、4、6 或整型值。

例如:

```
register int result asm("A");
int *xmemory;
int *ymemory;
int xVal, yVal;

result = __builtin_mpy(xVal, yVal,
                       &xmemory, &xVal, 2,
                       &ymemory, &yVal, 2);
```

可能生成:

```
mac w4*w5, A, [w8]+=2, w4, [w10]+=2, w5
```

原型:

```
int __builtin_mpy(int a, int b,
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr);
```

参数:

| | |
|--------------|------------------|
| <i>a</i> | 整型被乘数。 |
| <i>b</i> | 整型乘数。 |
| <i>xptr</i> | 指向 x 预取的指针的整型指针。 |
| <i>xval</i> | 指向 x 预取的值的整型指针。 |
| <i>xincr</i> | x 预取的整型递增值。 |
| <i>yptr</i> | 指向 y 预取的指针的整型指针。 |
| <i>yval</i> | 指向 y 预取的值的整型指针。 |
| <i>yincr</i> | y 预取的整型递增值。 |
| <i>AWB</i> | 指向所选累加器的整型指针。 |

注: 参数 *xptr* 和 *yptr* 必须分别指向位于 x 数据存储区和 y 数据存储区中的数组。

返回值:

返回 $a \times b$ 的值。

汇编操作符 / 机器指令:

mpy

错误消息:

如果出现以下情况, 将显示一条错误消息:

- 结果不是累加器寄存器
- *xval* 是一个空值, 但 *xptr* 不为空
- *yval* 是一个空值, 但 *yptr* 不为空

__builtin_mpy**说明:**

计算 $-a \times b$; 此外, 为之后的 MAC 操作预取数据。

xptr 可以为空, 表示不需要执行 X 预取, 这种情况下 *xincr* 和 *xval* 的值会被忽略, 但它们是必需的。

yptr 可以为空, 表示不需要执行 Y 预取, 这种情况下 *yincr* 和 *yval* 的值会被忽略, 但它们是必需的。

xval 和 *yval* 指定 C 变量的地址, 预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数: -6、-4、-2、0、2、4、6 或整型值。

例如:

```
register int result asm("A");
int *xmemory;
int *ymemory;
int xVal, yVal;

result = __builtin_mpy(xVal, yVal,
                      &xmemory, &xVal, 2,
                      &ymemory, &yVal, 2);
```

可能生成:

```
mac w4*w5, A, [w8]+=2, w4, [w10]+=2, w5
```

原型:

```
int __builtin_mpy(int a, int b,
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr);
```

参数:

| | |
|--------------|------------------|
| <i>a</i> | 整型被乘数。 |
| <i>b</i> | 整型乘数。 |
| <i>xptr</i> | 指向 x 预取的指针的整型指针。 |
| <i>xval</i> | 指向 x 预取的值的整型指针。 |
| <i>xincr</i> | x 预取的整型递增值。 |
| <i>yptr</i> | 指向 y 预取的指针的整型指针。 |
| <i>yval</i> | 指向 y 预取的值的整型指针。 |
| <i>yincr</i> | y 预取的整型递增值。 |
| <i>AWB</i> | 指向所选累加器的整型指针。 |

注: 参数 *xptr* 和 *yptr* 必须分别指向位于 x 数据存储区和 y 数据存储区中的数组。

返回值:

返回 $-a \times b$ 的值。

汇编操作符 / 机器指令:

mpyn

错误消息:

如果出现以下情况, 将显示一条错误消息:

- 结果不是累加器寄存器
- *xval* 是一个空值, 但 *xptr* 不为空
- *yval* 是一个空值, 但 *yptr* 不为空

__builtin_msc

说明:

计算 $a \times b$ ，并从累加器中减去；此外，为之后的 MAC 操作预取数据。

xptr 可以为空，表示不需要执行 X 预取，这种情况下 *xincr* 和 *xval* 的值会被忽略，但它们是必需的。

yptr 可以为空，表示不需要执行 Y 预取，这种情况下 *yincr* 和 *yval* 的值会被忽略，但它们是必需的。

xval 和 *yval* 指定 C 变量的地址，预取值将存储在上述变量中。

xincr 和 *yincr* 可以为立即数：-6、-4、-2、0、2、4、6 或整型值。

如果 *AWB* 非空，则另一个累加器会被回写到所引用的变量中。

例如：

```
register int result asm("A");
int *xmemory;
int *ymemory;
int xVal, yVal;

result = __builtin_msc(result, xVal, yVal,
                        &xmemory, &xVal, 2,
                        &ymemory, &yVal, 2, 0, 0);
```

可能生成：

```
msc w4*w5, A, [w8]+=2, w4, [w10]+=2, w5
```

原型:

```
int __builtin_msc(int Accum, int a, int b,
int **xptr, int *xval, int xincr,
int **yptr, int *yval, int yincr, int *AWB,
int AWB_accum);
```

参数:

| | |
|-----------|------------------|
| Accum | 要相减的累加器。 |
| a | 整型被乘数。 |
| b | 整型乘数。 |
| xptr | 指向 x 预取的指针的整型指针。 |
| xval | 指向 x 预取的值的整型指针。 |
| xincr | x 预取的整型递增值。 |
| yptr | 指向 y 预取的指针的整型指针。 |
| yval | 指向 y 预取的值的整型指针。 |
| yincr | y 预取的整型递增值。 |
| AWB | 累加器回写目标。 |
| AWB_accum | 要被回写的累加器。 |

| |
|---|
| 注： 参数 <i>xptr</i> 和 <i>yptr</i> 必须分别指向位于 x 数据存储区和 y 数据存储区中的数组。 |
|---|

返回值:

返回累加器减去 $a \times b$ 的结果得到的值。

汇编操作符 / 机器指令:

msc

__builtin_msc (续)

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- *Accum* 不是累加器寄存器
- *xval* 是一个空值，但 *xptr* 不为空
- *yval* 是一个空值，但 *yptr* 不为空
- *AWB_accum* 不是累加器寄存器，并且 *AWB* 不为空

`__builtin_mulss`

说明:

计算乘积 $p0 \times p1$ 。函数参数为有符号整型，函数结果为有符号长整型。可以使用命令行选项 `-Wconversions` 来检测意外的符号转换。

原型:

```
signed long __builtin_mulss(const signed int p0, const signed int p1);
```

参数:

p0 被乘数

p1 乘数

返回值:

返回乘积 $p0 \times p1$ 的有符号长整型值。

汇编操作符 / 机器指令:

`mul.ss`

`__builtin_mulsu`

说明:

计算乘积 $p0 \times p1$ 。函数参数为混合符号整型，函数结果为有符号长整型。可以使用命令行选项 `-Wconversions` 来检测意外的符号转换。该函数支持指令的所有寻址模式，包括操作数 *p1* 的立即数寻址模式。

原型:

```
signed long __builtin_mulsu(const signed int p0, const unsigned int p1);
```

参数:

p0 被乘数

p1 乘数

返回值:

返回乘积 $p0 \times p1$ 的有符号长整型值。

汇编操作符 / 机器指令:

`mul.su`

`__builtin_mulus`

说明：

计算乘积 $p0 \times p1$ 。函数参数为混合符号整型，函数结果为有符号长整型。可以使用命令行选项 `-Wconversions` 来检测意外的符号转换。该函数支持指令的所有寻址模式。

原型：

```
signed long __builtin_mulus(const unsigned int p0, const signed int p1);
```

参数：

p0 被乘数

p1 乘数

返回值：

返回乘积 $p0 \times p1$ 的有符号长整型值。

汇编操作符 / 机器指令：

`mul.us`

`__builtin_muluu`

说明：

计算乘积 $p0 \times p1$ 。函数参数为无符号整型，函数结果为无符号长整型。可以使用命令行选项 `-Wconversions` 来检测意外的符号转换。该函数支持指令的所有寻址模式，包括操作数 *p1* 的立即数寻址模式。

原型：

```
unsigned long __builtin_muluu(const unsigned int p0, const unsigned int p1);
```

参数：

p0 被乘数

p1 乘数

返回值：

返回乘积 $p0 \times p1$ 的无符号长整型值。

汇编操作符 / 机器指令：

`mul.uu`

`__builtin_nop`

说明:

生成一条 `nop` 指令。

原型:

```
void __builtin_nop(void);
```

参数:

无。

返回值:

返回无操作 (`nop`)。

汇编操作符 / 机器指令:

`nop`

`__builtin_psvoffset`

说明:

返回地址作为参数给出的对象的 `psv` 页偏移量。参数 `p` 必须是位于 `EEdata`、`PSV` 或可执行存储空间中的对象的地址；否则会产生错误消息，并且编译会失败。请参见《MPLAB® XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 `space` 属性。

原型:

```
unsigned int __builtin_psvoffset(const void *p);
```

参数:

`p` 对象的地址

返回值:

返回地址作为参数给出的对象的 `psv` 页偏移量。

汇编操作符 / 机器指令:

`psvoffset`

错误消息:

如果该函数的使用不正确，则会产生以下错误消息：

“Argument to `__builtin_psvoffset()` is not the address of an object in code, psv, or eedata section” (`__builtin_psvoffset()` 的参数不是位于代码、`psv` 或 `eedata` 段中的对象的地址)。

参数必须为显式对象地址。

例如，如果 `obj` 是一个位于可执行或只读段中的对象，则以下语法是有效的：

```
unsigned page = __builtin_psvoffset(&obj);
```

__builtin_psvpage

说明:

返回地址作为参数给出的对象的 psv 页号。参数 *p* 必须是位于 EEdata、PSV 或可执行存储空间中的对象的地址；否则会产生错误消息，并且编译会失败。请参见《MPLAB® XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 space 属性。

原型:

```
unsigned int __builtin_psvpage(const void *p);
```

参数:

p 对象的地址

返回值:

返回地址作为参数给出的对象的 psv 页号。

汇编操作符 / 机器指令:

psvpage

错误消息:

如果该函数的使用不正确，则会产生以下错误消息：

“Argument to __builtin_psvpage() is not the address of an object in code, psv, or eedata section” (__builtin_psvpage() 的参数不是位于代码、psv 或 eedata 段中的对象的地址)。

参数必须为显式对象地址。

例如，如果 *obj* 是一个位于可执行或只读段中的对象，则以下语法是有效的：

```
unsigned page = __builtin_psvpage(&obj);
```

__builtin_readsfr

说明:

读取 SFR。

原型:

```
unsigned int __builtin_readsfr(const void *p);
```

参数:

p 对象的地址

返回值:

返回 SFR。

汇编操作符 / 机器指令:

readsfr

错误消息:

如果对象地址不在 SFR 存储空间范围内，将产生错误消息。SFR 存储空间范围请查阅您所用器件的数据手册。

__builtin_return_address

说明:

返回当前函数或其调用方之一的返回地址。对于 *level* 参数，值为 0 将产生当前函数的返回地址，值为 1 将产生当前函数的调用方的返回地址，如此类推。当 *level* 超出当前堆栈深度时，将返回 0。该函数应仅使用非零参数，用于调试目的。

原型:

```
int __builtin_return_address (const int level);
```

参数:

level 要在调用堆栈中扫描的帧数。

返回值:

返回当前函数或其调用函数之一的返回地址。

汇编操作符 / 机器指令:

return_address

__builtin_sac

说明:

将值移位 *shift*（介于 -8 和 7 之间的立即数）位，并返回值。

例如:

```
register int value asm("A");
int result;
```

```
result = __builtin_sac(value,3);
```

可能生成:

```
sac A, #3, w0
```

原型:

```
int __builtin_sac(int value, int shift);
```

参数:

value 要进行移位的整型数。

shift 表示移位位数的立即数。

返回值:

将移位后的结果返回到累加器。

汇编操作符 / 机器指令:

sac

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- 移位值不是处于范围内的立即数

__builtin_sacr

说明:

将值移位 *shift*（介于 -8 和 7 之间的立即数）位，并返回使用由 CORCONbits.RND 控制位决定的舍入模式进行舍入后的值。

例如:

```
register int value asm("A");
int result;

result = __builtin_sac(value,3);
```

可能生成:

```
sac.r A, #3, w0
```

原型:

```
int __builtin_sacr(int value, int shift);
```

参数:

value 要进行移位的整型数。
shift 表示移位位数的立即数。

返回值:

将移位后的结果返回到 CORCON 寄存器。

汇编操作符 / 机器指令:

```
sacr
```

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- 移位值不是处于范围内的立即数

__builtin_sftac

说明:

将累加器移位 *shift* 位。有效移位范围为 -16 至 16。

例如:

```
register int result asm("A");
int i;

result = __builtin_sftac(result,i);
```

可能生成:

sftac A, w0

原型:

```
int __builtin_sftac(int Accum, int shift);
```

参数:

Accum 要移位的累加器。
shift 要移位的位数。

返回值:

将移位后的结果返回到累加器。

汇编操作符 / 机器指令:

sftac

错误消息:

如果出现以下情况，将显示一条错误消息:

- 结果不是累加器寄存器
- *Accum* 不是累加器寄存器
- 移位值不是处于范围内的立即数

__builtin_subab

说明:

将累加器 A 和 B 相减，并将结果回写到指定的累加器。例如：

```
register int result asm("A");
register int B asm("B");
result = __builtin_subab(result,B);
```

将生成：

```
sub A
```

原型:

```
int __builtin_subab(int Accum_a, int Accum_b);
```

参数:

Accum_a 作为被减数的累加器。

Accum_b 作为减数的累加器。

返回值:

将减法结果返回到累加器。

汇编操作符 / 机器指令:

```
sub
```

错误消息:

如果结果不是累加器寄存器，则将显示一条错误消息。

__builtin_tbladdress

说明:

返回一个表示对象在程序存储器中的地址的值。参数 *p* 必须是位于 EEdata、PSV 或可执行存储空间中的对象的地址；否则会产生错误消息，并且编译会失败。请参见《MPLAB® XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 *space* 属性。

原型:

```
unsigned long __builtin_tblpage(const void *p);
```

参数:

p 对象的地址

返回值:

返回一个表示对象在程序存储器中的地址的 unsigned long 值。

汇编操作符 / 机器指令:

```
tbladdress
```

`__builtin_tbladdress`

错误消息:

如果该函数的使用不正确, 则会产生以下错误消息:

“Argument to `__builtin_tbladdress()` is not the address of an object in code, psv, or eedata section” (`__builtin_tbladdress()` 的参数不是位于代码、psv 或 eedata 段中的对象的地址)。

参数必须为显式对象地址。

例如, 如果 `obj` 是一个位于可执行或只读段中的对象, 则以下语法是有效的:

```
unsigned long page = __builtin_tbladdress(&obj);
```

`__builtin_tbloffset`

说明:

返回地址作为参数给出的对象的表页偏移量。参数 `p` 必须是位于 EEdata、PSV 或可执行存储空间中的对象的地址; 否则会产生错误消息, 并且编译会失败。请参见《MPLAB[®] XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节 “指定变量的属性” 中的 `space` 属性。

原型:

```
unsigned int __builtin_tbloffset(const void *p);
```

参数:

`p` 对象的地址

返回值:

返回地址作为参数给出的对象的表页偏移量。

汇编操作符 / 机器指令:

`tbloffset`

错误消息:

如果该函数的使用不正确, 则会产生以下错误消息:

“Argument to `__builtin_tbloffset()` is not the address of an object in code, psv, or eedata section” (`__builtin_tbloffset()` 的参数不是位于代码、psv 或 eedata 段中的对象的地址)。

参数必须为显式对象地址。

例如, 如果 `obj` 是一个位于可执行或只读段中的对象, 则以下语法是有效的:

```
Unsigned page = __builtin_tbloffset(&obj);
```

__builtin_tblpage

说明:

返回地址作为参数给出的对象的表页号。参数 *p* 必须是位于 EEdata、PSV 或可执行存储空间中的对象的地址；否则会产生错误消息，并且编译会失败。请参见《MPLAB® XC16 C 编译器用户指南》(DS50002071E_CN) 的第 2.3.1 节“指定变量的属性”中的 *space* 属性。

原型:

```
unsigned int __builtin_tblpage(const void *p);
```

参数:

p 对象的地址

返回值:

返回地址作为参数给出的对象的表页号。

汇编操作符 / 机器指令:

tblpage

错误消息:

如果该函数的使用不正确，则会产生以下错误消息：

“Argument to __builtin_tblpage() is not the address of an object in code, psv, or eedata section” (__builtin_tblpage() 的参数不是位于代码、psv 或 eedata 段中的对象的地址)。

参数必须为显式对象地址。

例如，如果 *obj* 是一个位于可执行或只读段中的对象，则以下语法是有效的：

```
unsigned page = __builtin_tblpage(&obj);
```

__builtin_tblrddh

说明:

发出 *tblrddh.w* 指令来从闪存或 EEDATA 存储器中读取一个字。必须将 TBLPAG 设置为指向相应的页。要实现该目的，可以使用 __builtin_tbloffset() 和 __builtin_tblpage()。

关于读取和写入程序闪存的完整详细信息，请参见具体器件的数据手册或相应的系列参考手册。

原型:

```
unsigned int __builtin_tblrddh(unsigned int offset);
```

参数:

offset 所需的存储器偏移量

返回值:

无。

汇编操作符 / 机器指令:

tblrddh

错误消息:

无。

`__builtin_tblrld`

说明:

发出 `tblrld.w` 指令来从闪存或 EEDATA 存储器中读取一个字。必须将 TBLPAG 设置为指向相应的页。要实现该目的, 可以使用 `__builtin_tbloffset()` 和 `__builtin_tblpage()`。

关于读取和写入程序闪存的完整详细信息, 请参见具体器件的数据手册或相应的系列参考手册。

原型:

```
unsigned int __builtin_tblrld(unsigned int offset);
```

参数:

offset 所需的存储器偏移量

返回值:

无。

汇编操作符 / 机器指令:

`tblrld`

错误消息:

无。

`__builtin_tblwth`

说明:

发出 `tblwth.w` 指令来向闪存或 EEDATA 存储器中写入一个字。必须将 TBLPAG 设置为指向相应的页。要实现该目的, 可以使用 `__builtin_tbloffset()` 和 `__builtin_tblpage()`。

关于读取和写入程序闪存的完整详细信息, 请参见具体器件的数据手册或相应的系列参考手册。

原型:

```
void __builtin_tblwth(unsigned int offset, unsigned int data);
```

参数:

offset 所需的存储器偏移量

data 要写入的数据

返回值:

无。

汇编操作符 / 机器指令:

`tblwth`

错误消息:

无。

__builtin_tblwtl

说明:

发出 `tblrdl.w` 指令来向闪存或 `EEDATA` 存储器中写入一个字。必须将 `TBLPAG` 设置为指向相应的页。要实现该目的, 可以使用 `__builtin_tbloffset()` 和 `__builtin_tblpage()`。关于读取和写入程序闪存的完整详细信息, 请参见具体器件的数据手册或相应的系列参考手册。

原型:

```
void __builtin_tblwtl(unsigned int offset, unsigned int data);
```

参数:

offset 所需的存储器偏移量
data 要写入的数据

返回值:

无。

汇编操作符 / 机器指令:

`tblwtl`

错误消息:

无。

例 6-1: 其他内联函数

```
#include "p33fxxxx.h"

volatile long Result_mpy1616;
volatile long Result_addab;
volatile long Result_subab;
volatile long Result_mpy3216;
volatile long Result_div3216;

register int Accu_A asm("A");
register int Accu_B asm("B");

inline static long mpy_32_16 (long, int);

inline static long mpy_32_16 (long x, int y)
{
    long result;
    int temp1, temp2;
    temp1 = (x>>1)&0x7FFF;
    temp2 = x>>16;
    Accu_A = __builtin_mpy (temp1, y, 0,0,0,0,0,0);
    Accu_A = __builtin_sftac (15);
    Accu_A = __builtin_mac (temp2, y, 0,0,0,0,0,0);
    asm("mov _ACCAL,%0\n\t"
        "mov _ACCAH,%d0" : "=r"(result) : "w"(Accu_A));
    return result;
}

int main (void)
{
    // Variable declarations
    int Input1;
    int Input2;
    int Input3;
    int Input4;
    long Input5;
    int Input6;
    long Input7;
    int Input8;

    // Enable 32-bit saturation, signed and fractional modes for both ACCA
    // and ACCB
    CORCON = 0x00C0;

    // Example of 16*16-bit fractional multiplication using ACCA
    Input1 = 32767;
    Input2 = 32767;
    Accu_A = __builtin_mpy (Input1, Input2, 0,0,0,0,0,0);
    asm("mov _ACCAL,%0\n\t"
        "mov _ACCAH,%d0" : "=r"(Result_mpy1616) : "w"(Accu_A));

    // Example of 16*16-bit fractional multiplication using ACCB
    Input3 = 16384;
    Input4 = 16384;
    Accu_B = __builtin_mpy (Input3, Input4, 0,0,0,0,0,0);
    asm("mov _ACCBL,%0\n\t"
        "mov _ACCBH,%d0" : "=r"(Result_mpy1616) : "w"(Accu_B));

    // Example of 32-bit addition using ACCA (ACCA = ACCA + ACCB)
    Accu_A = __builtin_addab();
    asm("mov _ACCAL,%0\n\t"
        "mov _ACCAH,%d0" : "=r"(Result_addab) : "w"(Accu_A));

    // Example of 32-bit subtraction using ACCB (ACCB = ACCB - ACCA)
    Accu_B = __builtin_subab();
    asm("mov _ACCBL,%0\n\t"
        "mov _ACCBH,%d0" : "=r"(Result_subab) : "w"(Accu_B));

    // Example of 32*16-bit fractional multiplication using ACCA
    Input5 = 0x7FFFFFFF;
    Input6 = 32767;
    Result_mpy3216 = mpy_32_16 (Input5, Input6);

    while(1);
}
```


例 6-2: 32 位除以 16 位除法函数

```
#include <p33Fxxxx.h>
#include "divide.h"

_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSDCMD & OSCIOFNC_OFF & POSCMD_NONE);
_FWDT(FWDTEN_OFF);

unsigned int divide_(long a, int b) {
    union convert {
        unsigned long l;
        unsigned int i[2];
    } c;

    int sign;
    unsigned int result;

    c.l = a;
    sign = c.i[1] ^ b;

    if (a < 0) a = (-a);
    if (b < 0) b = -b;
    result = __builtin_divud(a,b);
    result >>= 1;
    if (sign < 0) result = -result;
    return result;
}

int main(void)
{
    unsigned long dividend;
    unsigned int divisor;
    unsigned int quotient;

    dividend = 0x3FFFFFFF;
    divisor = 0x7FFF;

    quotient = divide_((long)dividend, (int)divisor);
    while(1);
}
```

注:

第 7 章 参考信息

目录

本章包括以下主要内容：

| | | |
|-----|--------------|-----|
| 7.1 | 指令位映射 | 484 |
| 7.2 | 指令集汇总表 | 486 |
| 7.3 | 版本历史 | 496 |

7.1 指令位映射

表 7-1 中对 16 位 MCU 和 DSC 系列器件的指令编码进行了汇总。表中包括每一条指令最高字节的编码。表中第一列表示操作码的 bit 23:20，而表中第一行表示操作码的 bit 19:16。操作码的第一个字节由第一行给出的位值附加第一列的位值而形成。例如，PUSH 指令的最高字节（最后一行，第九列）的编码为 11111000b（0xF8）。

注： 根据第 5 章“指令描述”中的指令描述，通过使用表 5-1 至表 5-12 可确定每一条指令的完整操作码。

表 7-1: 指令编码

| | | 操作码 <19:16> | | | | | | | | | | | | | | | |
|-------------|------|--|------------------------------|-------------|----------------------|--|-------------|--|--|----------------------|---------------------|--------------------|--|----------------------------|----------------------------|--|----------------------------|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 操作码 <23:20> | 0000 | NOP | BRA CALL GOTO RCALL | CALL | — | GOTO | RETLW | RETFIE RETURN | RCALL | DO ⁽¹⁾ | REPEAT | — | — | BRA ⁽¹⁾ (OA) | BRA ⁽¹⁾ (OB) | BRA ⁽¹⁾ (SA) | BRA ⁽¹⁾ (SB) |
| | 0001 | SUBR | | | | | | | | SUBBR | | | | | | | |
| | 0010 | MOV | | | | | | | | | | | | | | | |
| | 0011 | BRA (OV) | BRA (C) | BRA (Z) | BRA (N) | BRA (LE) | BRA (LT) | BRA (LEU) | BRA | BRA (NOV) | BRA (NC) | BRA (NZ) | BRA (NN) | BRA (GT) | BRA (GE) | BRA (GTU) | — |
| | 0100 | ADD | | | | | | | | ADDC | | | | | | | |
| | 0101 | SUB | | | | | | | | SUBB | | | | | | | |
| | 0110 | AND | | | | | | | | XOR | | | | | | | |
| | 0111 | IOR | | | | | | | | MOV | | | | | | | |
| | 1000 | MOV | | | | | | | | | | | | | | | |
| | 1001 | MOV | | | | | | | | | | | | | | | |
| | 1010 | BSET | BCLR | BTG | BTST | BTSTS | BTST | BTSS | BTSC | BSET | BCLR | BTG | BTST | BTSTS | BSW | BTSS | BTSC |
| | 1011 | ADD ADDC | SUB SUBB | AND XOR | IOR MOV | ADD ADDC | SUB SUBB | AND XOR | IOR MOV | MUL.US MUL.UU | MUL.SS MUL.SU | TBLRDH TBLRDL | TBLWTH TBLWTL | MUL | SUB SUBB | MOV.D | MOV |
| | 1100 | MAC ⁽¹⁾ MPY ⁽¹⁾ MPY.N ⁽¹⁾ MSC ⁽¹⁾ | | | CLRAC ⁽¹⁾ | MAC ⁽¹⁾ MPY ⁽¹⁾ MPY.N ⁽¹⁾ MSC ⁽¹⁾ | | | MOVSAC ⁽¹⁾ | SFTAC ⁽¹⁾ | ADD ⁽¹⁾ | LAC ⁽¹⁾ | ADD ⁽¹⁾ NEG ⁽¹⁾ SUB ⁽¹⁾ | SAC ⁽¹⁾ | SAC.R ⁽¹⁾ | — | FF1L FF1R |
| | 1101 | SL | ASR LSR | RLC RLNC | RRC RRNC | SL | ASR LSR | RLC RLNC | RRC RRNC | DIV.S DIV.U | DIVF ⁽¹⁾ | — | — | — | SL | ASR LSR | FBCL |
| | 1110 | CP0 | CP CPB | CP0 | CP CPB | — | — | CPBGT ⁽²⁾ CPBLT ⁽²⁾ CPSGT CPSLT | CPBEQ ⁽²⁾ CPBNE ⁽²⁾ CPSEQ CPSNE | INC INC2 | DEC DEC2 | COM NEG | CLR SETM | INC INC2 | DEC DEC2 | COM NEG | CLR SETM |
| | 1111 | ED ⁽¹⁾ EDAC ⁽¹⁾ MAC ⁽¹⁾ MPY ⁽¹⁾ | | | | — | — | — | — | PUSH | POP | LNK ULNK | SE ZE | DISI | DAW EXCH SWAP | CLRWDT MOVPAG ⁽²⁾ PWRSV POP.S PUSH.S RESET | NOPR |

注 1: 该指令仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 系列器件中可用。
2: 该指令仅在 PIC24E 和 dsPIC33E 系列器件中可用。

7.2 指令集汇总表

表 7-2 对 16 位 MCU 和 DSC 器件的整个指令集进行了汇总。表中给出了按字母顺序排列的指令集列表。表中包括指令汇编语法、描述、指令长度（以 24 位字为单位）、执行时间（以指令周期为单位）、受影响的状态位以及指令详细描述所处的页码。表 1-2 说明了指令集汇总表中使用的符号。

注： 此处列出的指令周期数是针对 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件的。在 PIC24E 和 dsPIC33E 器件中，一些指令需要额外的指令周期。有关详细信息，请参见第 3.3 节“指令集汇总表”和第 5.4 节“指令描述”。

表 7-2: 指令集汇总表

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|------------------------|-------------------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| ADD f {, WREG} | 目标寄存器 = f + WREG | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 99 |
| ADD #lit10, Wn | Wn = lit10 + Wn | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 100 |
| ADD Wb, #lit5, Wd | Wd = Wb + lit5 | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 101 |
| ADD Wb, Ws, Wd | Wd = Wb + Ws | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 102 |
| ADD Acc ⁽²⁾ | 累加器相加 | 1 | 1 | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | — | — | — | — | — | 103 |
| ADD Wso, #Slit4, Acc | 16 位有符号数与累加器相加 | 1 | 1 | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | — | — | — | — | — | 104 |
| ADDC f {, WREG} | 目标寄存器 = f + WREG + (C) | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 106 |
| ADDC #lit10, Wn | Wn = lit10 + Wn + (C) | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 107 |
| ADDC Wb, #lit5, Wd | Wd = Wb + lit5 + (C) | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 108 |
| ADDC Wb, Ws, Wd | Wd = Wb + Ws + (C) | 1 | 1 | — | — | — | — | — | — | ↕ | ↕ | ↕ | ↕ | ↕ | 110 |
| AND f {, WREG} | 目标寄存器 = f .AND. WREG | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 112 |
| AND #lit10, Wn | Wn = lit10 .AND. Wn | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 113 |
| AND Wb, #lit5, Wd | Wd = Wb .AND. lit5 | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 114 |
| AND Wb, Ws, Wd | Wd = Wb .AND. Ws | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 115 |
| ASR f {, WREG} | 目标寄存器 = 算术右移 f, LSb → C | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | ↕ | 117 |
| ASR Ws, Wd | Wd = 算术右移 Ws, LSb → C | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | ↕ | 119 |
| ASR Wb, #lit4, Wnd | Wnd = Wb 算术右移 lit4 位, LSb → C | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 121 |

图注： ↕ 置 1 或清零；↕ 可以被清零，但永远不会被置 1；↕ 可以被置 1，但永远不会被清零；“1”总是置 1；“0”总是清零；— 不变

注

- 1: 只有在相应的饱和功能使能时，SA、SB 和 SAB 才可被修改，否则将不会发生改变。
- 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
- 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
- 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
- 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
- 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|-----------------------------|------------------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| ASR Wb, Wns, Wnd | Wnd = Wb 算术右移 Wns 位, LSb → C | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 122 |
| BCLR f, #bit4 | 将 f 中的指定位清零 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 123 |
| BCLR Ws, #bit4 | 将 Ws 中的指定位清零 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 124 |
| BRA Expr | 无条件转移 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 126 |
| BRA Wn | 计算转移 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 128 |
| BRA C, Expr | 如果进位位为 1 则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 130 |
| BRA GE, Expr | 如果大于或等于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 132 |
| BRA GEU, Expr | 如果无符号大于或等于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 134 |
| BRA GT, Expr | 如果大于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 135 |
| BRA GTU, Expr | 如果无符号大于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 136 |
| BRA LE, Expr | 如果小于或等于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 137 |
| BRA LEU, Expr | 如果无符号小于或等于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 138 |
| BRA LT, Expr | 如果小于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 139 |
| BRA LTU, Expr | 如果无符号小于则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 140 |
| BRA N, Expr | 如果为负则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 141 |
| BRA NC, Expr | 如果进位位为零则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 142 |
| BRA NN, Expr | 如果非负则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 143 |
| BRA NOV, Expr | 如果未溢出则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 144 |
| BRA NZ, Expr | 如果非零则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 145 |
| BRA OA, Expr ⁽²⁾ | 如果累加器 A 溢出则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 146 |
| BRA OB, Expr ⁽²⁾ | 如果累加器 B 溢出则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 147 |
| BRA OV, Expr | 如果溢出则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 148 |
| BRA SA, Expr ⁽²⁾ | 如果累加器 A 饱和则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 149 |
| BRA SB, Expr ⁽²⁾ | 如果累加器 B 饱和则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 150 |
| BRA Z, Expr | 如果为零则转移 | 1 | 1 (2) | — | — | — | — | — | — | — | — | — | — | — | 151 |
| BSET f, #bit4 | 将 f 中的指定位置 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 152 |
| BSET Ws, #bit4 | 将 Ws 中的指定位置 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 153 |
| BSW.C Ws, Wb | 将 C 位内容写入 Ws<Wb> | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 155 |

图注: ↕ 置 1 或清零; ↕ 可以被清零, 但永远不会被置 1; ↕ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

注:

- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
- 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
- 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
- 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
- 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
- 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|--|--|----------|--------------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| BSW.Z Ws,Wb | 将 Z 位内容写入 Ws<Wb> | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 155 |
| BTG f,#bit4 | 将 f 中的指定位翻转 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 157 |
| BTG Ws,#bit4 | 将 Ws 中的指定位翻转 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 158 |
| BTSC f,#bit4 | 测试 f 中的指定位, 为零则跳过 | 1 | 1 (2 或 3) | — | — | — | — | — | — | — | — | — | — | — | 160 |
| BTSC Ws,#bit4 | 测试 Ws 中的指定位, 为零则跳过 | 1 | 1 (2 或 3) | — | — | — | — | — | — | — | — | — | — | — | 162 |
| BTSS f,#bit4 | 测试 f 中的指定位, 为 1 则跳过 | 1 | 1 (2 或 3) | — | — | — | — | — | — | — | — | — | — | — | 164 |
| BTSS Ws,#bit4 | 测试 Ws 中的指定位, 为 1 则跳过 | 1 | 1 (2 或 3) | — | — | — | — | — | — | — | — | — | — | — | 166 |
| BTST f,#bit4 | 测试 f 中的指定位, 并将被测试位的补码存储到全零标志位 Z 中 | 1 | 1 | — | — | — | — | — | — | — | — | — | ↕ | — | 168 |
| BTST.C Ws,#bit4 | 测试 Ws 中的指定位, 并将被测试位的值存储到进位标志位 C 中 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ↕ | 169 |
| BTST.Z Ws,#bit4 | 测试 Ws 中的指定位, 并将被测试位的补码存储到全零标志位 Z 中 | 1 | 1 | — | — | — | — | — | — | — | — | — | ↕ | — | 169 |
| BTST.C Ws,Wb | 测试 Ws<Wb> 中的指定位, 并将被测试位的值存储到进位标志位 C 中 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ↕ | 171 |
| BTST.Z Ws,Wb | 测试 Ws<Wb> 中的指定位, 并将被测试位的补码存储到全零标志位 Z 中 | 1 | 1 | — | — | — | — | — | — | — | — | — | ↕ | — | 171 |
| BTSTS f,#bit4 | 测试 f 中的指定位, 并将被测试位的补码存储到全零标志位 Z 中, 然后将 f 中的该位置 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | ↕ | — | 173 |
| BTSTS.C Ws,#bit4 | 测试 Ws 中的指定位, 并将被测试位的值存储到进位标志位 C 中, 然后将 Ws 中的该位置 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ↕ | 175 |
| BTSTS.Z Ws,#bit4 | 测试 Ws 中的指定位, 并将被测试位的补码存储到全零标志位 Z 中, 然后将 Ws 中的该位置 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | ↕ | — | 175 |
| CALL Expr | 调用子程序 | 2 | 2 | — | — | — | — | — | — | — | — | — | — | — | 177 |
| CALL Wn | 间接调用子程序 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 180 |
| CALL.L Wn ⁽³⁾ | 间接长调用子程序 | 1 | 4 | — | — | — | — | — | — | — | — | — | — | — | 183 |
| CLR f | f = 0x0000 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 184 |
| CLR WREG | WREG = 0x0000 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 184 |
| CLR Wd | Wd = 0 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 185 |
| CLR Acc,[Wx],Wxd,[Wy],Wyd,AWB ⁽²⁾ | 清零累加器 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | 186 |
| CLRWDWT | 清零看门狗定时器 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 188 |
| COM f {,WREG} | 目标寄存器 = f | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 189 |
| COM Ws,Wd | Wd = Ws | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 190 |

图注: ↕ 置 1 或清零; ↕ 可以被清零, 但永远不会被置 1; ↕ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

- 注
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|-----------------------------------|-------------------------------------|----------|------------------------------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| CP f | 比较 (f – WREG) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 191 |
| CP Wb, #lit5 | 比较 (Wb – lit5) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 192 |
| CP Wb, #lit8 | 比较 (Wb – lit8) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 193 |
| CP Wb, Ws | 比较 (Wb – Ws) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 194 |
| CP0 f | 比较 (f – 0x0000) | 1 | 1 | — | — | — | — | — | — | 1 | ⇕ | ⇕ | ⇕ | 1 | 196 |
| CP0 Ws | 比较 (Ws – 0x0000) | 1 | 1 | — | — | — | — | — | — | 1 | ⇕ | ⇕ | ⇕ | 1 | 197 |
| CPB f | 带借位比较 (f – WREG – \overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 198 |
| CPB Wb, #lit5 | 带借位比较 (Wb – lit5 – \overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 199 |
| CPB Wb, #lit8 | 带借位比较 (Wb – lit8 – \overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 200 |
| CPB Wb, Ws | 带借位比较 (Wb – Ws – \overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 201 |
| CPBEQ Wb, Wn, Expr ⁽³⁾ | 比较 Wb 和 Wn, 如果相等则转移 | 1 | $\frac{1}{(5)}$ | — | — | — | — | — | — | — | — | — | — | — | 203 |
| CPBGT Wb, Wn, Expr ⁽³⁾ | 有符号比较 Wb 和 Wn, 如果大于则转移 | 1 | $\frac{1}{(5)}$ | — | — | — | — | — | — | — | — | — | — | — | 204 |
| CPBLT Wb, Wn, Expr ⁽³⁾ | 有符号比较 Wb 和 Wn, 如果小于则转移 | 1 | $\frac{1}{(5)}$ | — | — | — | — | — | — | — | — | — | — | — | 205 |
| CPBNE Wb, Wn, Expr ⁽³⁾ | 比较 Wb 和 Wn, 如果不相等则转移 | 1 | $\frac{1}{(5)}$ | — | — | — | — | — | — | — | — | — | — | — | 206 |
| CPSEQ Wb, Wn | 比较 Wb 和 Wn, 如果相等则跳过 | 1 | $\frac{1}{(2 \text{ 或 } 3)}$ | — | — | — | — | — | — | — | — | — | — | — | 207 |
| CPSGT Wb, Wn | 有符号比较 Wb 和 Wn, 如果大于则跳过 | 1 | $\frac{1}{(2 \text{ 或 } 3)}$ | — | — | — | — | — | — | — | — | — | — | — | 211 |
| CPSLT Wb, Wn | 有符号比较 Wb 和 Wn, 如果小于则跳过 | 1 | $\frac{1}{(2 \text{ 或 } 3)}$ | — | — | — | — | — | — | — | — | — | — | — | 212 |
| CPSNE Wb, Wn | 比较 Wb 和 Wn, 如果不相等则跳过 | 1 | $\frac{1}{(2 \text{ 或 } 3)}$ | — | — | — | — | — | — | — | — | — | — | — | 214 |
| DAW.B Wn | Wn = 对 Wn 的内容进行十进制调整 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ⇕ | 216 |
| DEC f {, WREG} | 目标寄存器 = f – 1 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 217 |
| DEC Ws, Wd | Wd = Ws – 1 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 218 |
| DEC2 f {, WREG} | 目标寄存器 = f – 2 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 220 |
| DEC2 Ws, Wd | Wd = Ws – 2 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 221 |

图注: ⇕ 置 1 或清零; ⇓ 可以被清零, 但永远不会被置 1; ⇑ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

- 注
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|---|---|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| DISI #lit14 | 禁止中断, 持续时间为 lit14 个指令周期 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 223 |
| DIV.S Wm, Wn | 有符号 16/16 位整数除法, $Q \rightarrow Wo, R \rightarrow W1$ | 1 | 18 | — | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | 224 |
| DIV.SD Wm, Wn | 有符号 32/16 位整数除法, $Q \rightarrow Wo, R \rightarrow W1$ | 1 | 18 | — | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | 224 |
| DIV.U Wm, Wn | 无符号 16/16 位整数除法, $Q \rightarrow Wo, R \rightarrow W1$ | 1 | 18 | — | — | — | — | — | — | — | 0 | 0 | ⇕ | ⇕ | 226 |
| DIV.UD Wm, Wn | 无符号 32/16 位整数除法, $Q \rightarrow Wo, R \rightarrow W1$ | 1 | 18 | — | — | — | — | — | — | — | 0 | ⇕ | ⇕ | ⇕ | 226 |
| DIVF Wm, Wn ⁽²⁾ | 有符号 16/16 位小数除法, $Q \rightarrow Wo, R \rightarrow W1$ | 1 | 18 | — | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | 228 |
| DO #lit14, Expr ⁽⁶⁾ | 执行 DO 循环代码到 PC + Expr, 执行 (lit14 + 1) 次 | 2 | 2 | — | — | — | — | — | — | — | — | — | — | — | 230 |
| DO #lit15, Expr ⁽⁴⁾ | 执行 DO 循环代码到 PC + Expr, 执行 (lit15 + 1) 次 | 2 | 2 | — | — | — | — | — | — | — | — | — | — | — | 233 |
| DO Wn, Expr ⁽²⁾ | 执行 DO 循环代码到 PC + Expr, 执行 (Wn + 1) 次 | 2 | 2 | — | — | — | — | — | — | — | — | — | — | — | 235 |
| ED Wm*Wm, Acc, [Wx], [Wy], Wxd ⁽²⁾ | 求取欧几里德距离 (无累加) | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 239 |
| EDAC Wm*Wm, Acc, [Wx], [Wy], Wxd ⁽²⁾ | 求取欧几里德距离 | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 241 |
| EXCH Wns, Wnd | 交换 Wns 和 Wnd 的内容 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 243 |
| FBCL Ws, Wnd | 搜索自左 (MSb) 起第一个位变化 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ⇕ | 244 |
| FF1L Ws, Wnd | 搜索自左 (MSb) 起第一个 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ⇕ | 246 |
| FF1R Ws, Wnd | 搜索自右 (LSb) 起第一个 1 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | ⇕ | 248 |
| GOTO Expr | 跳转到地址 | 2 | 2 | — | — | — | — | — | — | — | — | — | — | — | 250 |
| GOTO Wn | 间接跳转到地址 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 251 |
| GOTO.L Wn ⁽³⁾ | 间接跳转到地址 (长地址) | 1 | 4 | — | — | — | — | — | — | — | — | — | — | — | 253 |
| INC f {, WREG} | 目标寄存器 = f + 1 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 254 |
| INC Ws, Wd | Wd = Ws + 1 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 255 |
| INC2 f {, WREG} | 目标寄存器 = f + 2 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 257 |
| INC2 Ws, Wd | Wd = Ws + 2 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 258 |
| IOR f {, WREG} | 目标寄存器 = f.IOR.WREG | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 260 |
| IOR #lit10, Wn | Wn = lit10.IOR.Wn | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 261 |
| IOR Wb, #lit5, Wd | Wd = Wb.IOR.lit5 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 262 |
| IOR Wb, Ws, Wd | Wd = Wb.IOR.Ws | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 263 |
| LAC Wso, #Slit4, Acc ⁽²⁾ | 装载累加器 | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 265 |

图注: ⇕ 置 1 或清零; ⇕ 可以被清零, 但永远不会被置 1; ⇕ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

- 注
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|--|---------------------------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| LNK #lit14 | 分配堆栈帧 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 267 |
| LSR f {, WREG} | 目标寄存器 = 逻辑右移 f, MSb → C | 1 | 1 | — | — | — | — | — | — | — | 0 | — | ⇕ | ⇕ | 269 |
| LSR Ws, Wd | Wd = 逻辑右移 Ws, MSb → C | 1 | 1 | — | — | — | — | — | — | — | 0 | — | ⇕ | ⇕ | 271 |
| LSR Wb, #lit4, Wnd | Wnd = Wb 逻辑右移 lit4 位, MSb → C | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 273 |
| LSR Wb, Wns, Wnd | Wnd = Wb 逻辑右移 Wns 位, MSb → C | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 274 |
| MAC Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd, AWB ⁽²⁾ | 相乘并累加 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 275 |
| MAC Wm*Wm, Acc, [Wx], Wxd, [Wy], Wyd ⁽²⁾ | 平方并累加 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 277 |
| MOV f {, WREG} | 将 f 中内容传送到目标寄存器 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 279 |
| MOV WREG, f | 将 WREG 中内容传送到 f | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 280 |
| MOV f, Wnd | 将 f 中内容传送到 Wnd | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 281 |
| MOV Wns, f | 将 Wns 中内容传送到 f | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 282 |
| MOV.B #lit8, Wnd | 将 8 位无符号立即数传送到 Wnd | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 283 |
| MOV #lit16, Wnd | 将 16 位立即数传送到 Wnd | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 284 |
| MOV [Ws+Slit10], Wnd | 将 [Ws + Slit10] 中内容传送到 Wnd | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 285 |
| MOV Wns, [Wd+Slit10] | 将 Wns 中内容传送到 [Wd + Slit10] | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 286 |
| MOV Wso, Wdo | 将 Wso 中内容传送到 Wdo | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 287 |
| MOV.D Wns, Wnd | 双字传送 Wns 内容至 Wnd:Wnd + 1 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 289 |
| MOV.D Wns, Wnd | 双字传送 Wns:Wns + 1 内容至 Wnd | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 289 |
| MOVPAG #lit10, DSRPAG(3) | 将 10 位立即数传送到 DSRPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 291 |
| MOVPAG #lit9, DSWPAG(3) | 将 9 位立即数传送到 DSWPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 291 |
| MOVPAG #lit8, TBLPAG(3) | 将 8 位立即数传送到 TBLPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 291 |
| MOVPAG Wn, DSRPAG ⁽³⁾ | 将 Wn 中内容传送到 DSRPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 292 |
| MOVPAG Wn, DSWPAG ⁽³⁾ | 将 Wn 中内容传送到 DSWPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 292 |
| MOVPAG Wn, TBLPAG ⁽³⁾ | 将 Wn 中内容传送到 TBLPAG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 292 |
| MOVSAC Acc, [Wx], Wxd, [Wy], Wyd, AWB ⁽²⁾ | 将 [Wx] 中内容传送到 Wxd, 并将 [Wy] 中内容传送到 Wyd | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 293 |
| MPY Wm*Wn, Acc, [Wx], Wxd, [Wy], Wyd ⁽²⁾ | Wn 和 Wm 相乘, 结果存入累加器 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 295 |

图注: ⇕ 置 1 或清零; ⇕ 可以被清零, 但永远不会被置 1; ↑ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

- 注:
- 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|---|--|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| MPY $Wm * Wn, Acc, [Wx], Wxd, [Wy], Wyd^{(2)}$ | 求平方, 结果存入累加器 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 297 |
| MPY.N $Wm * Wn, Acc, [Wx], Wxd, [Wy], Wyd^{(2)}$ | Wn 和 Wm 相乘并取反, 结果存入累加器 | 1 | 1 | 0 | 0 | — | — | 0 | — | — | — | — | — | — | 299 |
| MSC $Wm * Wn, Acc, [Wx], Wxd, [Wy], Wyd, AWB^{(2)}$ | 相乘并从累加器中减去乘积 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 301 |
| MUL f | W3:W2 = f * WREG | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 303 |
| MUL.SS Wb, Ws, Wnd | {Wnd + 1, Wnd} = 有符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 305 |
| MUL.SS $Wb, Ws, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 307 |
| MUL.SU $Wb, #lit5, Wnd$ | {Wnd + 1, Wnd} = 有符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 308 |
| MUL.SU Wb, Ws, Wnd | {Wnd + 1, Wnd} = 有符号 (Wb) * 无符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 310 |
| MUL.SU $Wb, Ws, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 无符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 312 |
| MUL.SU $Wb, #lit5, Acc^{(4)}$ | 累加器 = 有符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 314 |
| MUL.US Wb, Ws, Wnd | {Wnd + 1, Wnd} = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 315 |
| MUL.US $Wb, Ws, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 317 |
| MUL.UU $Wb, #lit5, Wnd$ | {Wnd + 1, Wnd} = 无符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 319 |
| MUL.UU Wb, Ws, Wnd | {Wnd + 1, Wnd} = 无符号 (Wb) * 无符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 320 |
| MUL.UU $Wb, Ws, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 无符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 322 |
| MUL.UU $Wb, #lit5, Acc^{(4)}$ | 累加器 = 无符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 323 |
| MULW.SS $Wb, Ws, Wnd^{(3)}$ | Wnd = 有符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 324 |
| MULW.SU $Wb, Ws, Wnd^{(3)}$ | Wnd = 有符号 (Wb) * 无符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 326 |
| MULW.SU $Wb, #lit5, Wnd^{(3)}$ | Wnd = 有符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 328 |
| MULW.US $Wb, Ws, Wnd^{(3)}$ | Wnd = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 329 |
| MULW.UU $Wb, Ws, Wnd^{(3)}$ | Wnd = 无符号 (Wb) * 有符号 (Ws) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 331 |
| MULW.UU $Wb, #lit5, Wnd^{(3)}$ | Wnd = 无符号 (Wb) * 无符号 (lit5) | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 332 |
| NEG $f \{, WREG\}$ | 目标寄存器 = $\bar{f} + 1$ | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 333 |
| NEG Ws, Wd | Wd = $\overline{Ws} + 1$ | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 334 |
| NEG $Acc^{(2)}$ | 对累加器内容求补 | 1 | 1 | ⇕ | ⇕ | ↑ | ↑ | ⇕ | ↑ | — | — | — | — | — | 335 |
| NOP | 空操作 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 336 |

图注: ⇕ 置 1 或清零; ⇕ 可以被清零, 但永远不会被置 1; ↑ 可以被置 1, 但永远不会被清零; “1”总是置 1; “0”总是清零; — 不变

- 注**
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|-------------------------------------|-------------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| NOPR | 空操作 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 336 |
| POP f | 栈顶内容弹出至 f | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 337 |
| POP Wdo | 栈顶内容弹出至 Wdo | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 338 |
| POP.D Wnd | 将栈顶内容双字弹出至 Wnd:Wnd + 1 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 339 |
| POP.S | 弹出影子寄存器内容 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 340 |
| PUSH f | 将 f 内容压入栈顶 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 341 |
| PUSH Wso | 将 Wso 内容压入栈顶 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 342 |
| PUSH.D Wns | 将 Wns:Wns + 1 内容双字压入栈顶 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 343 |
| PUSH.S | 压入影子寄存器 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 345 |
| PWRSV #lit1 | 进入低功耗模式 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 346 |
| RCALL Expr | 相对调用 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 347 |
| RCALL Wn | 计算调用 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 351 |
| REPEAT #lit14 ⁽⁵⁾ | 重复执行下一条指令 (lit14 + 1) 次 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 355 |
| REPEAT #lit15 ⁽³⁾ | 重复执行下一条指令 (lit15 + 1) 次 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 357 |
| REPEAT Wn | 重复执行下一条指令 (Wn + 1) 次 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 359 |
| RESET | 软件器件复位 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 363 |
| RETFIE | 从中断返回 | 1 | 3 (2) | — | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | 365 |
| RETLW #lit10, Wn | 从子程序返回并将 lit10 存入 Wn | 1 | 3 (2) | — | — | — | — | — | — | — | — | — | — | — | 367 |
| RETURN | 从子程序返回 | 1 | 3 (2) | — | — | — | — | — | — | — | — | — | — | — | 371 |
| RLC f {, WREG} | 目标寄存器 = 带进位位循环左移 f 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 373 |
| RLC Ws, Wd | Wd = 带进位位循环左移 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 375 |
| RLNC f {, WREG} | 目标寄存器 = 不带进位位循环左移 f 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 377 |
| RLNC Ws, Wd | Wd = 不带进位位循环左移 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 379 |
| RRC f {, WREG} | 目标寄存器 = 带进位位循环右移 f 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 381 |
| RRC Ws, Wd | Wd = 带进位位循环右移 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 383 |
| RRNC f {, WREG} | 目标寄存器 = 不带进位位循环右移 f 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 385 |
| RRNC Ws, Wd | Wd = 不带进位位循环右移 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 387 |
| SAC Acc, #Slit4, Wdo ⁽²⁾ | 保存累加器内容 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 389 |

图注: ⇕ 置 1 或清零; ⇓ 可以被清零, 但永远不会被置 1; ⇑ 可以被置 1, 但永远不会被清零; “1”总是置 1; “0”总是清零; — 不变

- 注
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|---------------------------------------|---------------------------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| SAC.R Acc, #Slit4, Wdo ⁽²⁾ | 保存舍入后的累加器内容 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 391 |
| SE Ws, Wd | Wd = 符号扩展 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 393 |
| SETM f | f = 0xFFFF | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 395 |
| SETM WREG | WREG = 0xFFFF | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 395 |
| SETM Wd | Wd = 0xFFFF | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 396 |
| SFTAC Acc, #Slit6 ⁽²⁾ | 算术移位累加器内容 Slit6 位 | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 397 |
| SFTAC Acc, Wb ⁽²⁾ | 算术移位累加器内容 (Wb) 位 | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 398 |
| SL f {, WREG} | 目标寄存器 = 算术左移 f 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 399 |
| SL Ws, Wd | Wd = 算术左移 Ws 内容 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | ⇕ | 401 |
| SL Wb, #lit4, Wnd | Wnd = Wb 左移 lit4 位 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 403 |
| SL Wb, Wns, Wnd | Wnd = Wb 左移 Wns 位 | 1 | 1 | — | — | — | — | — | — | — | ⇕ | — | ⇕ | — | 404 |
| SUB f {, WREG} | 目标寄存器 = f - WREG | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 405 |
| SUB #lit10, Wn | Wn = Wn - lit10 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 406 |
| SUB Wb, #lit5, Wd | Wd = Wb - lit5 | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 407 |
| SUB Wb, Ws, Wd | Wd = Wb - Ws | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 408 |
| SUB Acc ⁽²⁾ | 累加器相减 | 1 | 1 | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | — | — | — | — | — | 410 |
| SUBB f {, WREG} | 目标寄存器 = f - WREG - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 411 |
| SUBB #lit10, Wn | Wn = Wn - lit10 - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 412 |
| SUBB Wb, #lit5, Wd | Wd = Wb - lit5 - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 413 |
| SUBB Wb, Ws, Wd | Wd = Wb - Ws - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 415 |
| SUBBR f {, WREG} | 目标寄存器 = WREG - f - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 417 |
| SUBBR Wb, #lit5, Wd | Wd = lit5 - Wb - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 418 |
| SUBBR Wb, Ws, Wd | Wd = Ws - Wb - (\overline{C}) | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 420 |
| SUBR f {, WREG} | 目标寄存器 = WREG - f | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 422 |
| SUBR Wb, #lit5, Wd | Wd = lit5 - Wb | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 423 |
| SUBR Wb, Ws, Wd | Wd = Ws - Wb | 1 | 1 | — | — | — | — | — | — | ⇕ | ⇕ | ⇕ | ⇕ | ⇕ | 424 |
| SWAP Wn | Wn = Wn 内容字节或半字节交换 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 426 |

图注: ⇕ 置 1 或清零; ⇓ 可以被清零, 但永远不会被置 1; ⇑ 可以被置 1, 但永远不会被清零; “1” 总是置 1; “0” 总是清零; — 不变

- 注
- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
 - 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
 - 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
 - 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
 - 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
 - 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

表 7-2: 指令集汇总表 (续)

| 汇编语法 助记符, 操作数 | 描述 | 指令 字数 | 指令 周期数 | OA ⁽²⁾ | OB ⁽²⁾ | SA ^(1,2) | SB ^(1,2) | OAB ⁽²⁾ | SAB ^(1,2) | DC | N | OV | Z | C | 页码 |
|------------------|--------------------|----------|-----------|-------------------|-------------------|---------------------|---------------------|--------------------|----------------------|----|---|----|---|---|-----|
| TBLRDH [Ws],Wd | 将高位程序字读入 Wd | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 427 |
| TBLRDL [Ws],Wd | 将低位程序字读入 Wd | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 429 |
| TBLWTH Ws,[Wd] | 将 Ws 中的内容写入高位程序字 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 431 |
| TBLWTL Ws,[Wd] | 将 Ws 中的内容写入低位程序字 | 1 | 2 | — | — | — | — | — | — | — | — | — | — | — | 433 |
| ULNK | 释放堆栈帧 | 1 | 1 | — | — | — | — | — | — | — | — | — | — | — | 435 |
| XOR f {,WREG} | 目标寄存器 = f.XOR.WREG | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 437 |
| XOR #lit10,Wn | Wn = lit10.XOR.Wn | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 438 |
| XOR Wb,#lit5,Wd | Wd = Wb.XOR.lit5 | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 439 |
| XOR Wb,Ws,Wd | Wd = Wb.XOR.Ws | 1 | 1 | — | — | — | — | — | — | — | ↕ | — | ↕ | — | 440 |
| ZE Ws,Wnd | Wnd = 零扩展 Ws | 1 | 1 | — | — | — | — | — | — | — | 0 | — | ↕ | 1 | 442 |

图注: ↕ 置 1 或清零; ↕ 可以被清零, 但永远不会被置 1; ↕ 可以被置 1, 但永远不会被清零; “1”总是置 1; “0”总是清零; — 不变

注

- 1: 只有在相应的饱和功能使能时, SA、SB 和 SAB 才可被修改, 否则将不会发生改变。
- 2: 该指令 / 操作数仅在 dsPIC30F、dsPIC33F 和 dsPIC33E 器件中可用。
- 3: 该指令 / 操作数仅在 PIC24E 和 dsPIC33E 器件中可用。
- 4: 该指令 / 操作数仅在 dsPIC33E 器件中可用。
- 5: 该指令 / 操作数仅在 PIC24F、PIC24H、dsPIC30F 和 dsPIC33F 器件中可用。
- 6: 该指令 / 操作数仅在 dsPIC30F 和 dsPIC33F 器件中可用。

7.3 版本历史

版本 A（2005 年 5 月）

这是本文档的初始版本。

版本 B（2005 年 9 月）

该版本包括在本文档更新时的所有已知勘误。

版本 C（2008 年 2 月）

该版本包括以下修正和更新：

- 指令更新：
 - 更新了 BRA 指令（见 **“BRA”**）
 - 更新了 DIVF 指令（见 **“DIVF”**）
 - 更新了 DO 指令（见 **“DO”**）
 - 更新了 SUB 指令（见 **“SUB”**）

版本 D（2009 年 11 月）

该版本包括以下修正和更新：

- 将文档从《dsPIC30F/33F 程序员参考手册》更名为《16 位 MCU 和 DSC 程序员参考手册》
- 完全重新设计了文档以适合目前所有的 16 位系列：dsPIC30F、dsPIC33F、PIC24F 和 PIC24H

版本 E（2010 年 6 月）

该版本包括以下修正和更新：

- 在整篇文档中增加了特定于 dsPIC33E 和 PIC24E 器件的信息

版本 F（2011 年 7 月）

该版本包括以下修正和更新：

- 增加了一个新章节**第 6 章 “内建函数”**
- 整篇文档中增加并更新了交叉引用
- 在**寄存器 2-4**和**寄存器 2-6**中，将位特性从 U 更新为 U-0
- 整篇文档中增加了一个注，指定在 dsPIC33E 和 PIC24E 器件中对非 CPU 特殊功能寄存器的读操作和读 - 修改 - 写操作需要一个额外周期的要求
- 在整篇文档中，对格式进行了更新并更改了少量文本内容

索引

符号

| | |
|----------------------------|----------|
| __builtin_addab | 447 |
| __builtin_btg | 447, 448 |
| __builtin_divmodud | 454 |
| __builtin_divsd | 454 |
| __builtin_edsoffset | 458 |
| __builtin_edspage | 458 |
| __builtin_mac | 460 |
| __builtin_modsd | 462 |
| __builtin_modud | 462 |
| __builtin_mpy | 464 |
| __builtin_mpy_n | 465 |
| __builtin_mulss | 468 |
| __builtin_muluu | 469 |
| __builtin_nop | 470 |
| __builtin_psvoffset | 470 |
| __builtin_sac | 472 |
| __builtin_subab | 475 |
| __builtin_tbladdress | 475 |
| __builtin_tblwth | 478 |
| __builtin_tblwtl | 479 |

B

| | |
|--------------|----|
| 本手册的宗旨 | 6 |
| 编程模型 | 14 |
| 寄存器说明 | 18 |

C

| | |
|---|-----|
| 产生 16 位结果的 16x16 位无符号短立即数整数乘法 | 332 |
| 产生 16 位结果的 16x16 位有符号 - 无符号短立即数整数乘法 | 328 |
| 程序计数器 | 19 |
| 程序空间寻址模式 | 61 |
| 改变程序流的方法 | 61 |

D

| | |
|-----------------------|----|
| DCOUNT 寄存器 | 20 |
| DOEND 寄存器 | 21 |
| DOSTART 寄存器 | 20 |
| DSP MAC 间接寻址模式 | 57 |
| DSP MAC 类指令 | 84 |
| dsPIC30F/33F 概述 | 10 |
| DSP 累加器类指令 | 88 |
| DSP 数据格式 | 81 |

代码示例

| | |
|------------------------------------|----|
| 32 位加法的 Z 状态位操作 | 77 |
| 带有效地址更新的间接寻址 | 55 |
| 堆栈指针的使用 | 71 |
| 非法的字传送操作 | 68 |
| 合法的字传送操作 | 67 |
| 基本 MAC 语法 | 85 |
| 寄存器偏移量间接寻址 | 56 |
| 寄存器直接寻址 | 54 |
| 立即数寻址 | 59 |
| MAC 累加器回写语法 | 87 |
| MAC 预取语法 | 86 |
| 使用 10 位立即数作为字节操作数 | 69 |
| 使用 FBCL 进行换算 | 89 |
| 使用默认工作寄存器 WREG | 79 |
| 文件寄存器寻址 | 53 |
| 文件寄存器寻址和 WREG | 53 |
| 无符号 f 和 WREG 相乘（传统 MULWF 指令） | 80 |
| 用立即数偏移量进行传送的指令 | 56 |

| | |
|----------------------------|----|
| 帧指针用法 | 73 |
| 执行两次预取和累加器回写的 MSC 指令 | 87 |
| 字节数学操作示例 | 65 |
| 字节传送操作示例 | 64 |
| 堆栈指针限制寄存器（SPLIM） | 19 |
| 多周期指令 | 39 |
| 多字指令 | 39 |

F

| | |
|----------------|---|
| 风格和符号的约定 | 7 |
| 文档约定 | 7 |

G

| | |
|------------------|----|
| 工作寄存器阵列 | 18 |
| 规定的工作寄存器用法 | 78 |

J

寄存器

| | |
|-----------------------|--------|
| CORCON（内核控制） | 32, 35 |
| CORCON（内核控制）寄存器 | 31, 33 |
| SR（CPU 状态） | 26, 29 |
| SR（状态）寄存器 | 27 |
| 寄存器间接寻址 | 54 |
| 模式 | 54 |
| 寄存器间接寻址和指令集 | 57 |
| 寄存器直接寻址 | 53 |
| 简介 | 6 |

K

| | |
|------------|---|
| 开发支持 | 6 |
|------------|---|

L

| | |
|-------------------|----|
| 累加器 A，累加器 B | 19 |
| 累加器的使用 | 83 |
| 累加器访问 | 84 |
| 累加器选择 | 97 |
| 立即数寻址 | 58 |
| 指令集中的操作数 | 58 |

M

MAC

| | |
|-------------------------------|--------|
| 操作 | 85 |
| 回写 | 85 |
| 语法 | 85 |
| 预取 | 84 |
| 预取寄存器更新 | 85 |
| MAC 或 MPY 源操作数（不同工作寄存器） | 97 |
| MAC 或 MPY 源操作数（同一工作寄存器） | 97 |
| MAC 累加器回写选择 | 97 |
| MOPAG 目的寄存器选择 | 97 |
| 默认工作寄存器（WREG） | 18, 79 |
| 模寻址和位反转寻址模式 | 57 |

N

| | |
|---------------------------|----------|
| 内核控制寄存器 | 24 |
| 内建函数 | |
| __builtin_addab | 447 |
| __builtin_btg | 447, 448 |
| __builtin_divmodud | 454 |
| __builtin_divsd | 454 |
| __builtin_edsoffset | 458 |
| __builtin_edspage | 458 |

16 位 MCU 和 DSC 程序员参考手册

| | |
|----------------------------|-----|
| __builtin_mac | 460 |
| __builtin_modsd | 462 |
| __builtin_modud | 462 |
| __builtin_mpy | 464 |
| __builtin_mpy_n | 465 |
| __builtin_mulss | 468 |
| __builtin_muluu | 469 |
| __builtin_nop | 470 |
| __builtin_psvoffset | 470 |
| __builtin_sac | 472 |
| __builtin_subab | 475 |
| __builtin_tbladdress | 475 |
| __builtin_tblwth | 478 |
| __builtin_tblwtl | 479 |

P

| | |
|------------------------|----|
| PRODH:PRODL 寄存器对 | 79 |
| PSVPAG 寄存器 | 19 |

R

| | |
|------------------|--------|
| RCOUNT 寄存器 | 20 |
| 软件堆栈帧指针 | 18, 72 |
| 上溢 | 74 |
| 示例 | 73 |
| 下溢 | 75 |
| 软件堆栈指针 | 19, 70 |
| 示例 | 71 |

S

| | |
|----------------------------|----|
| 使用 10 位立即数操作数 | 69 |
| 10 位立即数编码 | 69 |
| 使用 FBCL 指令对数据进行换算 | 88 |
| 换算实例 | 89 |
| 使用 FBCL 指令将累加器中内容归一化 | 90 |
| 数据空间寻址模式树 | 59 |
| 数据寻址模式 | 52 |

T

| | |
|------------------|----|
| TBLPAG 寄存器 | 19 |
| 条件转移指令 | 76 |

W

| | |
|---------------|----|
| 文件寄存器寻址 | 52 |
|---------------|----|

X

| | |
|---------------------|----|
| X 数据空间预取操作 | 96 |
| X 数据空间预取目的寄存器 | 96 |

Y

| | |
|------------------------------------|----|
| Y 数据空间预取操作 | 96 |
| Y 数据空间预取目的寄存器 | 97 |
| 隐含的 DSP 操作数 | 78 |
| 隐含的帧指针和堆栈指针 | 78 |
| 影子寄存器 | 24 |
| 自动使用 | 24 |
| 用于 Wd 目的寄存器的偏移量寻址模式（带寄存器偏移量） | 95 |
| 用于 Wd 目的寄存器的寻址模式 | 95 |
| 用于 Ws 源寄存器的偏移量寻址模式（带寄存器偏移量） | 95 |
| 用于 Ws 源寄存器的寻址模式 | 95 |
| 与 PIC 单片机的兼容性 | 79 |

Z

| | |
|---------------|----|
| Z 状态位 | 77 |
| 整数和小数数据 | 81 |

| | |
|-----------------------------|-----|
| 表示 | 82 |
| 指令编码字段描述符介绍 | 94 |
| 指令符号 | 94 |
| 指令集符号 | 8 |
| (text) | 8 |
| [text] | 8 |
| { } | 8 |
| [label:] | 8 |
| #text | 8 |
| <n:m> | 8 |
| Acc | 8 |
| AWB | 8 |
| bit4 | 8 |
| Expr | 8 |
| f | 8 |
| lit1 | 8 |
| lit10 | 8 |
| lit14 | 8 |
| lit16 | 8 |
| lit23 | 8 |
| lit4 | 8 |
| lit5 | 8 |
| lit8 | 8 |
| Slit10 | 8 |
| Slit16 | 8 |
| Slit4 | 8 |
| Slit6 | 8 |
| TOS | 8 |
| Wb | 8 |
| Wd | 8 |
| Wm, Wn | 8 |
| Wm*Wm | 8 |
| Wm*Wn | 8 |
| Wn | 8 |
| Wnd | 8 |
| Wns | 8 |
| WREG | 8 |
| Ws | 8 |
| Wx | 8 |
| Wxd | 8 |
| Wy | 8 |
| Wyd | 8 |
| 指令集概述 | 38 |
| 比较 / 跳过指令 | 46 |
| 程序流指令 | 47 |
| dsPIC30F/33F 指令组 | 38 |
| DSP 类指令 | 50 |
| 控制指令 | 49 |
| 逻辑指令 | 43 |
| 数学指令 | 41 |
| 位操作指令 | 45 |
| 循环 / 移位指令 | 44 |
| 影子 / 堆栈指令 | 49 |
| 传送指令 | 40 |
| 指令集汇总表 | 486 |
| 指令描述 | 99 |
| ADD（16 位有符号数与累加器相加） | 104 |
| ADD（f 与 WREG 相加） | 99 |
| ADD（累加器相加） | 103 |
| ADD（立即数和 Wn 逻辑与） | 113 |
| ADD（立即数与 Wn 相加） | 100 |
| ADD（Wb 和短立即数逻辑与） | 114 |
| ADD（Wb 和 Ws 逻辑与） | 115 |
| ADD（Wb 和 Ws 相加） | 102 |
| ADD（Wb 与短立即数相加） | 101 |
| ADDC（f 与 WREG 带进位位相加） | 106 |
| ADDC（立即数与 Wn 带进位位相加） | 107 |

| | |
|--|----------|
| ADDC (Wb 和短立即数带进位位相加) | 108 |
| ADDC (Wb 和 Ws 带进位位相加) | 110 |
| AND (f 和 WREG 逻辑与) | 112 |
| ASR (算术右移, 移位位数由短立即数确定) | 121 |
| ASR (算术右移, 移位位数由 Wns 确定) | 122 |
| ASR (算术右移 f) | 117 |
| ASR (算术右移 Ws) | 119 |
| BCLR.B (将 f 中的指定位清零) | 123 |
| BCLR (将 Ws 中的指定位清零) | 124 |
| BRA C (如果进位位为 1 则转移) | 130 |
| BRA GE (如果有符号大于或等于则转移) | 132 |
| BRA GEU (如果无符号大于或等于则转移) | 134 |
| BRA GT (如果有符号大于则转移) | 135 |
| BRA GTU (如果无符号大于则转移) | 136 |
| BRA LE (如果有符号小于或等于则转移) | 137 |
| BRA LEU (如果无符号小于或等于则转移) | 138 |
| BRA LT (如果有符号小于则转移) | 139 |
| BRA LTU (如果进位位为零则转移) | 142 |
| BRA LTU (如果无符号小于则转移) | 140 |
| BRA N (如果为负则转移) | 141 |
| BRA NN (如果非负则转移) | 143 |
| BRA NOV (如果未溢出则转移) | 144 |
| BRA NZ (如果非零则转移) | 145 |
| BRA OA (如果累加器 A 溢出则转移) | 146 |
| BRA OB (如果累加器 B 溢出则转移) | 147 |
| BRA OV (如果溢出则转移) | 148 |
| BRA SA (如果累加器 A 饱和则转移) | 149 |
| BRA SB (如果累加器 B 饱和则转移) | 150 |
| BRA Z (如果为零则转移) | 151 |
| BRA (计算转移) | 128, 129 |
| BRA (无条件转移) | 126 |
| BSET (将 f 中的指定位置 1) | 152 |
| BSET (将 Ws 中的指定位置 1) | 153 |
| BSW (写 Ws 中的某位) | 155 |
| BTG (将 f 中的指定位翻转) | 157 |
| BTG (将 Ws 中的指定位翻转) | 158 |
| BTSC (测试 f 中的指定位, 为零则跳过) | 160 |
| BTSC (测试 Ws 中的指定位, 为零则跳过) | 162 |
| BTSS (测试 f 中的指定位, 为 1 则跳过) | 164 |
| BTSS (测试 Ws 中的指定位, 为 1 则跳过) | 166 |
| BTST (测试 f 中的指定位) | 168 |
| BTST (测试 Ws 中的指定位) | 169, 171 |
| BTSTS (测试 / 置 1 f 中的指定位) | 173 |
| BTSTS (测试 / 置 1 Ws 中的指定位) | 175 |
| CALL.L (间接调用子程序长地址) | 183 |
| CALL (调用子程序) | 177, 178 |
| CALL (间接调用子程序) | 180, 181 |
| CBSLT (有符号比较 Wb 和 Wn, 如果小于 则转移) | 205 |
| CLR (清零 f 或 WREG) | 184 |
| CLR (清零累加器, 预取操作数) | 186 |
| CLR (清零 Wd) | 185 |
| CLRWD (清零看门狗定时器) | 188 |
| COM (对 f 的内容取反) | 189 |
| COM (对 Ws 的内容取反) | 190 |
| CP (比较 f 和 WREG 并设置状态标志位) | 191 |
| CP (比较 Wb 和 lit5 并设置状态标志位) | 192 |
| CP (比较 Wb 和 lit8 并设置状态标志位) | 193 |
| CP (比较 Wb 和 Ws 并设置状态标志位) | 194 |
| CP0 (比较 f 和 0x0 并设置状态标志位) | 196 |
| CP0 (比较 Ws 和 0x0 并设置状态标志位) | 197 |
| CPB (带借位比较 f 和 WREG 并设置状态 标志位) | 198 |
| CPB (带借位比较 Wb 和 lit5 并设置状态 标志位) | 199 |
| CPB (带借位比较 Wb 和 lit8 并设置状态 标志位) | 200 |
| CPB (带借位比较 Ws 和 Wb 并设置状态 标志位) | 201 |
| CPBEQ (比较 Wb 和 Wn, 如果相等则转移) | 203 |
| CPBGT (有符号比较 Wb 和 Wn, 如果大于 则转移) | 204 |
| CPBNE (比较 Wb 和 Wn, 如果不等于 则转移) | 206 |
| CPSEQ (比较 Wb 和 Wn, 如果相等 则跳过) | 207, 208 |
| CPSGT (有符号比较 Wb 和 Wn, 如果大于 则跳过) | 210, 211 |
| CPSLT (有符号比较 Wb 和 Wn, 如果小于 则跳过) | 212, 213 |
| CPSNE (比较 Wb 和 Wn, 如果不等于 则跳过) | 214, 215 |
| DAW.B (对 Wn 的内容进行十进制调整) | 216 |
| DEC (f 内容递减 1) | 217 |
| DEC (Ws 内容递减 1) | 218 |
| DEC2 (f 内容递减 2) | 220 |
| DEC2 (Ws 内容递减 2) | 221 |
| DISI (暂时禁止中断) | 223 |
| DIV.S (有符号整数除法) | 224 |
| DIV.U (无符号整数除法) | 226 |
| DIVF (小数除法) | 228 |
| DO (初始化硬件循环立即数) | 230, 233 |
| DO (初始化硬件循环 Wn) | 235, 237 |
| ED (求取欧几里德距离, 无累加) | 239 |
| EDAC (求取欧几里德距离) | 241 |
| EXCH (交换 Wns 和 Wnd 的内容) | 243 |
| FBCL (搜索自左起第一个位变化) | 244 |
| FF1L (搜索自左起第一个 1) | 246 |
| FF1R (搜索自右起第一个 1) | 248 |
| GOTO.L (无条件间接跳转长地址) | 253 |
| GOTO (无条件间接跳转) | 251, 252 |
| GOTO (无条件跳转) | 250 |
| INC (f 内容递增 1) | 254 |
| INC (Ws 内容递增 1) | 255 |
| INC2 (f 内容递增 2) | 257 |
| INC2 (Ws 内容递增 2) | 258 |
| IOR (f 和 WREG 逻辑或) | 260 |
| IOR (立即数和 Wn 逻辑或) | 261 |
| IOR (Wb 和短立即数逻辑或) | 262 |
| IOR (Wb 和 Ws 逻辑或) | 263 |
| LAC (装载累加器) | 265 |
| LNK (分配堆栈帧) | 267, 268 |
| LSR (逻辑右移, 移位位数由短立即数确定) | 273 |
| LSR (逻辑右移, 移位位数由 Wns 确定) | 274 |
| LSR (逻辑右移 f) | 269 |
| LSR (逻辑右移 Ws) | 271 |
| MAC (平方并累加) | 277 |
| MAC (相乘并累加) | 275 |
| MOV.B (将 8 位立即数传送到 Wnd) | 283 |
| MOV.D (双字传送源寄存器内容至 Wnd) | 289 |
| MOV (将 [Ws + 偏移量] 中内容传送到 Wnd) | 285 |
| MOV (将 16 位立即数传送到 Wn) | 284 |
| MOV (将 f 中内容传送到目的寄存器) | 279 |
| MOV (将 f 中内容传送到 Wnd) | 281 |
| MOV (将 Wns 中内容传送到 [Wd + 偏移量]) | 286 |
| MOV (将 Wns 中内容传送到 f) | 282 |
| MOV (将 WREG 中内容传送到 f) | 280 |
| MOV (将 Ws 中内容传送到 Wd) | 287 |
| MOVPAG (将立即数传送到页寄存器) | 291 |
| MOVPAG (将 Ws 中内容传送到页寄存器) | 292 |
| MOVSAC (预取操作数并保存累加器内容) | 293 |
| MPY.N (-Wm 和 Wn 相乘, 结果存入累加器) | 299 |
| MPY (求平方, 结果存入累加器) | 297 |
| MPY (Wm 和 Wn 相乘, 结果存入累加器) | 295 |

| | | | |
|---|----------|---------------------------------|----------|
| MSC (相乘并从累加器中减去乘积) | 301 | SETM (将 f 或 WREG 置全 1) | 395 |
| MUL.SS (16x16 位有符号整数乘法) | 305 | SETM (将 Ws 置全 1) | 396 |
| MUL.SS (使用累加器作为目的寄存器的 16x16 位 有符号整数乘法) | 307 | SFTAC (算术移位累加器内容 Slit6 位) | 397 |
| MUL.SU (16x16 位有符号 - 无符号短立即数整数 乘法) | 308 | SFTAC (算术移位累加器内容 Wb 位) | 398 |
| MUL.SU (16x16 位有符号 - 无符号整数乘法) | 310 | SL (左移, 移位位数由短立即数确定) | 403 |
| MUL.SU (使用累加器作为目的寄存器的 16x16 位 有符号 - 无符号短立即数整数乘法) | 314 | SL (左移, 移位位数由 Wns 内容确定) | 404 |
| MUL.SU (使用累加器作为目的寄存器的 16x16 位 有符号 - 无符号整数乘法) | 312 | SL (左移 f 内容) | 399 |
| MUL.US (16x16 位无符号 - 有符号整数乘法) | 315 | SL (左移 Ws 内容) | 401 |
| MUL.US (使用累加器作为目的寄存器的 16x16 位 无符号 - 有符号整数乘法) | 317 | SUB (f 减 WREG) | 405 |
| MUL.UU (16x16 位无符号短立即数整数乘法) | 319 | SUB (累加器相减) | 410 |
| MUL.UU (16x16 位无符号整数乘法) | 320 | SUB (Wb 减短立即数) | 407 |
| MUL.UU (使用累加器作为目的寄存器的 16x16 位 无符号短立即数整数乘法) | 323 | SUB (Wb 减 Ws) | 408 |
| MUL.UU (使用累加器作为目的寄存器的 16x16 位 无符号整数乘法) | 322 | SUB (Wn 减立即数) | 406 |
| MUL (f 与 WREG 无符号整数乘法) | 303 | SUBB (f 减 WREG, 带借位) | 411 |
| MULW.SS (产生 16 位结果的 16x16 位有符号整数 乘法) | 324 | SUBB (立即数减 Wn, 带借位) | 412 |
| MULW.SU (产生 16 位结果的 16x16 位有符号 - 无符号短立即数整数乘法) | 328 | SUBB (Wb 减短立即数, 带借位) | 413 |
| MULW.SU (产生 16 位结果的 16x16 位有符号 - 无符号整数乘法) | 326 | SUBB (Wb 减 Ws, 带借位) | 415 |
| MULW.US (产生 16 位结果的 16x16 位无符号 - 有符号整数乘法) | 329 | SUBBR (短立即数减 Wb, 带借位) | 418 |
| MULW.UU (产生 16 位结果的 16x16 位无符号 短立即数整数乘法) | 332 | SUBBR (WREG 减 f, 带借位) | 417 |
| MULW.UU (产生 16 位结果的 16x16 位无符号 整数乘法) | 331 | SUBBR (Ws 减 Wb, 带借位) | 420 |
| NEG (对 f 内容求补) | 333 | SUBR (短立即数减 Wb) | 423 |
| NEG (对累加器内容求补) | 335 | SUBR (WREG 减 f) | 422 |
| NEG (对 Ws 内容求补) | 334 | SUBR (Ws 减 Wb) | 424 |
| NOP (空操作) | 336 | SWAP (Wn 内容字节或半字节交换) | 426 |
| NOPR (空操作) | 336 | TBLRDH (表读高位字) | 427 |
| POP.D (将栈顶内容双字弹出至 Wnd/Wnd+1) | 339 | TBLRDL (表读低位字) | 429 |
| POP.S (弹出影子寄存器内容) | 340 | TBLWTH (表写高位字) | 431 |
| POP (栈顶内容弹出至 f) | 337 | TBLWTL (表写低位字) | 433 |
| POP (栈顶内容弹出至 Wd) | 338 | ULNK (释放堆栈帧) | 435, 436 |
| PUSH.D (将 Wns:Wns+1 内容双字压入栈顶) | 343 | XOR (f 和 WREG 逻辑异或) | 437 |
| PUSH.S (压入影子寄存器) | 345 | XOR (立即数和 Wn 逻辑异或) | 438 |
| PUSH (将 f 内容压入栈顶) | 341 | XOR (Wb 和短立即数逻辑异或) | 439 |
| PUSH (将 Ws 内容压入栈顶) | 342 | XOR (Wb 和 Ws 逻辑异或) | 440 |
| PWRSV (进入低功耗模式) | 346 | ZE (零扩展 Wn) | 442 |
| RCALL (计算相对调用) | 351, 353 | 指令描述示例 | 98 |
| RCALL (相对调用) | 347, 349 | 指令停顿 | 62 |
| REPEAT (重复执行下一条指令 (lit14 + 1) 次) | 355 | DO/REPEAT 循环 | 63 |
| REPEAT (重复执行下一条指令 (lit15 + 1) 次) | 357 | 改变程序流的指令 | 63 |
| REPEAT (重复执行下一条指令 (Wn + 1) 次) | 359, 361 | PSV | 63 |
| RESET (复位) | 363 | RAW 相关性检测 | 62 |
| RETFIE (从中断返回) | 365, 366 | 异常 | 63 |
| RETLW (返回并将立即数存入 Wn) | 367, 369 | 指令位映射 | 484 |
| RETURN (返回) | 371, 372 | 中断优先级 | 23 |
| RLC (带进位位循环左移 f 内容) | 373 | 状态寄存器 | 21 |
| RLC (带进位位循环左移 Ws 内容) | 375 | DSP ALU 状态位 | 23 |
| RLNC (不带进位位循环左移 f 内容) | 377 | MCU ALU 状态位 | 21 |
| RLNC (不带进位位循环左移 Ws 内容) | 379 | 循环状态位 | 21 |
| RRC (带进位位循环右移 f 内容) | 381 | 字节操作 | 64 |
| RRC (带进位位循环右移 Ws 内容) | 383 | 字传送操作 | 66 |
| RRNC (不带进位位循环右移 f 内容) | 385 | 存储区中数据对齐方式 | 66 |
| RRNC (不带进位位循环右移 Ws 内容) | 387 | | |
| SAC.R (保存舍入后的累加器内容) | 391 | | |
| SAC (保存累加器内容) | 389 | | |
| SE (符号扩展 Ws 内容) | 393 | | |

注:

全球销售及服务中心

美洲

公司总部 **Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277

技术支持:
<http://www.microchip.com/support>

网址: www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 905-695-1980
Fax: 905-695-2078

亚太地区

亚太总部 **Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

亚太地区

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 - 高雄
Tel: 886-7-213-7828

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Venice
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

波兰 Poland - Warsaw
Tel: 48-22-3325737

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

06/23/16