

# SDIC 精简指令集说明


## 1. 指令集概述

本指令集包含 47 条指令，本文档后面的部分将讨论该指令集。


该指令集中的 47 条指令中，只有 2 个指令是双字指令，其余都是单字指令（16 位）。

每个单字指令都是一个 16 位字，由操作码（指明指令类型）和一个或多个操作数（指定指令操作）组成。

整个指令集具有高度的正交性，可以分为 4 种基本类型：

 **字节操作类指令**

 **位操作类指令**

 **立即数操作类指令**


 **控制操作类指令**

表 3.1 为该指令集汇总，表 2.1 给出了操作码字段的说明。

双字指令为双字（32 位），执行双字指令需要 2 个或 3 个指令周期。

每个指令周期有 4 个振荡器周期组成。因此，对于频率为 4MHz 的振荡器，其正常的指令执行时间为 1us（振荡周期为250ns）。

## 2. 操作码字段说明

字段	说明
a	快速操作 RAM 位： a = 0：快速操作 RAM 内的 RAM 单元（BSR 寄存器被忽略）； a = 1：由 BSR 寄存器指定的 RAM 快速操作存储区。
b	8 位文件寄存器内的位地址（0 到 7）。
BSR	存储区选择寄存器。用于选择当前的 RAM 存储区。
C、DC、Z	ALU 状态位：进位标志位、辅助进位标志位和全零标志位。
d	目标寄存器选择位： d = 0：结果保存至 WREG 寄存器； d = 1：结果保存至文件寄存器 f。

dest	目标寄存器：可以是 WREG 寄存器或指定的寄存器地址。
f	8 位寄存器地址（00h 到 FFh）。
k	立即数、常数或者标号（可能是 8 位、12 位或 20 位的值）。
标号	标号名称
*	表读指令的寄存器模式，只与表读指令一起使用。
n	相对调整指令的相对地址（二进制补码），或 CALL、BRA 和 RETURN 指令的直接地址。
PC	程序计数器。
PCL	程序寄存器低字节。
PCH	程序计数器高字节
PCLATH	程序计数器高字节锁存器。
PCLATU	程序计数器最高字节锁存器。
s	快速调用、返回模式选择位：  s = 0：不对影子寄存器进行更新，也不用影子寄存器的内容更新其它寄存器  s = 1：将寄存器的值存入影子寄存器或把影子寄存器中的值载入寄存器（快速模式）
PD	掉电位
T0	超时溢出位
u	未使用或未改变
WDT	看门狗寄存器
WREG	工作寄存器
x	忽略（0 或 1），汇编器将产生 x = 0 的代码。
{ }	可选参数。

表 2.1

### 3. 指令集

指令	说明	周期	受影响的状态位
ADDWF f, d, a	WREG 与 f 相加，结果放入 d 指定的内容中	1	C、DC 和 Z
ADDWFC f, d, a	WREG 与 f 带进位位相加，结果放入 d 指定的内容中	1	C、DC 和 Z
ANDWF f, d, a	WREG 与 f 做与运算，结果放入 d 指定的内容中	1	Z
CLRF f, a	f 清零	1	Z
COMF f, d, a	f 取反，结果放入由 d 指定的内容中	1	Z
DECF f, d, a	f 减 1，结果放入由 d 指定的内容中	1	C、DC 和 Z
DECFSZ f, d, a	f 减 1，结果为 0 则跳过下条指令，结果放入由 d 指定的内容中	1(2 或 3)	无
DCFSNZ f, d, a	f 减 1，结果不为 0 则跳过下条指令，结果放入由 d 指定的内容中	1(2 或 3)	无
INCF f, d, a	f 加 1，结果放入由 d 指定的内容中	1	C、DC 和 Z
INCFSZ f, d, a	f 加 1，结果为 0 则跳过下条指令，结果放入由 d 指定的内容中	1(2 或 3)	无
INFSNZ f, d, a	f 加 1，结果不为 0 则跳过下条指令，结果放入由 d 指定的内容中	1(2 或 3)	无
CPFSEQ f, a	将 f 与 WREG 做比较，相等则跳过下条指令	1(2 或 3)	无
IORWF f, d, a	WREG 与 f 做或运算，结果放入 d 指定的内容中	1	Z

MOVF f, d, a	移动 f 到 d 指定的内容中	1	Z
MOVWF f, a	将 WREG 移入 f	1	无
RLCF f, d, a	将 f 执行带进位的循环左移，结果放入 d 指定的内容中	1	C 和 Z
RLNCF f, d, a	将 f 执行不带进位的循环左移，结果放入 d 指定的内容中	1	Z
RRCF f, d, a	将 f 执行带进位的循环右移，结果放入 d 指定的内容中	1	C 和 Z
RRNCF f, d, a	将 f 执行不带进位的循环右移，结果放入 d 指定的内容中	1	Z
SETF f, a	将 f 置为全 1	1	无
SUBWF f, d, a	f 减去 WREG，结果放入 d 指定的内容中	1	C、DC 和 Z
SUBWFB f, d, a	f 带借位减去 WREG，结果放入 d 指定的内容中	1	C、DC 和 Z
TSTFSZ f, a	测试 f，为 0 则跳过下条指令	1(2 或 3)	无
XORWF f, d, a	WREG 与 f 做异或运算，结果放入 d 指定的内容中	1	Z
BCF f, b, a	将 f 中的第 b 位清零	1	无
BSF f, b, a	将 f 中的第 b 位置 1	1	无
BTFSC f, b, a	测试 f 中的 b 位，为 0 则跳过下条指令	1(2 或 3)	无
BTFSS f, b, a	测试 f 中的 b 位，为 1 则跳过下条指令	1(2 或 3)	无

BRA	n	无条件跳转	2	无
CALL	n, s	调用子程序，双字指令	2	无
CLRWDT		看门狗定时器清零	1	T0 和 PD
GOTO	n	跳转到地址（无条件），双字指令	2	无
NOP		空操作	1	无
RCALL	n	相对调用	2	无
RESET		软件复位	1	无
RETFIE	s	中断返回	2	无
RETLW	k	返回时将立即数送入 WREG	2	无
RETURN	s	从子程序返回	2	无
SLEEP		进入待机模式	1	T0 和 PD
ADDLW	k	立即数 k 和 WREG 相加，结果放入 WREG	1	C、DC 和 Z
ANDLW	k	立即数 k 与 WREG 相与，结果放入 WREG	1	Z
IORLW	k	立即数 k 与 WREG 相或，结果放入 WREG	1	Z
MOVLB	k	将立即数 k 移入 BSR<3:0>	1	无
MOVLW	k	将立即数 k 移入 WREG	1	无
SUBLW	k	立即数 k 减去 WREG，结果放入 WREG	1	C、DC 和 Z
XORLW	k	立即数 k 与 WREG 相异或，结果放入 WREG	1	Z
TBLRD*		表读 ROM	2	无
寄存器间接寻址		对 INDF0 操作实际是以 FSR0H:FSR0L 的内容作为地址进行操作。	-	-

表 3.1

## 4. 指令说明

### 4.1 ADDLW - WREG 与立即数相加

语法:                ADDLW    k

操作数:              $0 \leq k \leq 255$

操作:                 $(WREG) + k \rightarrow WREG$

说明:                将 WREG 寄存器的内容与 8 位立即数 k 相加, 结果保存在 WREG 寄存器。

示例:                ADDLW    0x15

                      指令执行前, WREG = 0x10;

                      指令执行后, WREG = 0x25;

### 4.2 ADDWF - WREG 与 f 寄存器相加

语法:                ADDWF    f, d, a

操作数:              $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:                 $(WREG) + (f) \rightarrow dest$

说明:                将 WREG 的内容和 f 寄存器的内容相加。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例:                ADDWF    REG, 0, 0

                      指令执行前, WREG = 0x17, REG = 0xC2;

                      指令执行后, WREG = 0xD9, REG = 0xC2。

### 4.3 ADDWFC - WREG 与 f 寄存器带进位位相加

语法:                ADDWFC   f, d, a

操作数:              $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:                 $(WREG) + (f) + (C) \rightarrow dest$

说明:                将 WREG 的内容、进位标志位 C 和 f 寄存器的内容相加。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例:                   ADDWFC REG, 1, 1

指令执行前, C = 1, WREG = 0x4D, REG = 0x02;

指令执行后, C = 0, WREG = 0x4D, REG = 0x50。

#### 4.4 ANDLW - 立即数和 WREG 寄存器作逻辑与运算

语法:                   ANDLW k

操作数:                 $0 \leq k \leq 255$

操作:                   (WREG) & k  $\rightarrow$  WREG

说明:                   将 WREG 寄存器的内容与 8 位立即数 k 相与, 结果保存在 WREG 寄存器。

示例:                   ANDLW 0x00

指令执行前, WREG = 0xFF;

指令执行后, WREG = 0x00。

#### 4.5 ANDWF - 将 WREG 和 f 作逻辑与运算

语法:                   ANDWF f, d, a

操作数:                 $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:                   (WREG) & (f)  $\rightarrow$  dest

说明:                   将 WREG 的内容和 f 寄存器的内容相与。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例:                   ANDWF REG, 1, 0

指令执行前, WREG = 0x55, REG = 0xAA;

指令执行后, WREG = 0x55, REG = 0x00。

#### 4.6 BCF - 将 f 寄存器中的某位清零

语法:                   BCF f, b, a

操作数:                 $0 \leq f \leq 255$

$0 \leq b \leq 7$

$a \in [0, 1]$

操作:                   0  $\rightarrow$  f<b>

说明：将寄存器 f 中的位 b 清零。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例：BCF REG, 7, 1  
指令执行前，REG = 0xFF；  
指令执行后，REG = 0x7F。

#### 4.7 BSF - 将 f 寄存器中的某位置 1

语法：BSF f, b, a

操作数： $0 \leq f \leq 255$

$0 \leq b \leq 7$

$a \in [0, 1]$

操作： $0 \rightarrow f[b]$

说明：将寄存器 f 中的位 b 置 1。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例：BSF REG, 7, 0  
指令执行前，REG = 0x00；  
指令执行后，REG = 0x80。

#### 4.8 BRA - 无条件跳转

语法：BRA n

操作数： $-1024 \leq n \leq 1023$

操作： $(PC) + 2 + 2n \rightarrow (PC)$

说明：二进制补码“2n”与 PC 相加，由于 PC 要先递增才能取下一条指令，所以新地址将为  $PC + 2 + 2n$ 。

示例：Here:  
BRA There  
指令执行前，PC = 地址(Here)；  
指令执行后，PC = 地址(There)。

#### 4.9 BTFSC - 测试寄存器中的位，为 0 则跳过下条指令

语法：BTFSC f, b, a

操作数： $0 \leq f \leq 255$

$0 \leq b \leq 7$

$a \in [0, 1]$



操作: 如果  $f\langle b \rangle = 0$  则跳过下条指令

说明: 如果寄存器  $f$  的位  $b$  为 0, 则跳过下条指令。即在位  $b$  为 0 时, 丢弃下一条指令而执行一条 NOP 指令。如果  $a$  为 0, 选择快速操作存储区 (SFR), 如果  $a$  为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: BTFSC REG, 7, 0  
BSF REG, 6, 0  
BCF REG, 0, 0  
指令执行前, REG = 0x01;  
指令执行后, REG = 0x00。

#### 4.10 BTFSS - 测试寄存器中的位, 为 1 则跳过下条指令

语法: BTFSS  $f, b, a$

操作数:  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0, 1]$

操作: 如果  $f\langle b \rangle = 0$  则跳过下条指令

说明: 如果寄存器  $f$  的位  $b$  为 1, 则跳过下条指令。即在位  $b$  为 1 时, 丢弃下一条指令而执行一条 NOP 指令。如果  $a$  为 0, 选择快速操作存储区 (SFR), 如果  $a$  为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: BTFSS REG, 7, 1  
BSF REG, 6, 1  
BCF REG, 0, 1  
指令执行前, REG = 0x01;  
指令执行后, REG = 0x40。

#### 4.11 CALL - 调用子程序

语法: CALL  $n\{, s\}$

操作数:  $0 \leq n \leq 1048575$   
 $s \in [0, 1]$

操作:  $(PC) + 4 \rightarrow TOS, n \rightarrow PC\langle 20:1 \rangle$   
如果  $s = 1$ :  
 $(WREG) \rightarrow WS, STATUS \rightarrow STATUSS, BSR \rightarrow BSRS$

说明: 可在整个 2MB 的存储范围内进行子程序调用。首先, 将返回地址  $(PC + 4)$  压栈, 如果  $s = 1$ , 还会将 WREG、STATUS 和 BSR 存入

对应的影子寄存器，如果  $s = 0$ （默认），则不会，然后将 20 位的值  $n$  装入  $PC<20:1>$ 。

示例：

Here：

CALL            There

指令执行前， $PC = \text{地址(Here)}$ ；

指令执行后， $PC = \text{地址(There)}$

$TOS = \text{地址}(\text{Here} + 4)$

其中 TOS 为栈顶。

#### 4.12 CLRF - 将 f 清零

语法：CLRF f, a

操作数： $0 \leq f \leq 255$

$a \in [0, 1]$

操作： $0x00 \rightarrow f$

说明：清零寄存器 f 中。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例：CLRF REG, 0

指令执行前， $REG = 0xFF$ ；

指令执行后， $REG = 0x00$ 。

#### 4.13 CLRWD - 清零看门狗定时器

语法：CLRWD

操作数：无

操作： $0x00 \rightarrow WDT$

说明：清零看门狗定时器（喂狗）。

示例：CLRWD

指令执行前，WDT 计数器 =  $0xFF$ ；

指令执行后，WDT 计数器 =  $0x00$ 。

#### 4.14 COMF - 将 f 取反

语法：COMF f, d, a

操作数： $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(\sim f) \rightarrow \text{dest}$

说明: 将寄存器 f 的内容取反。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区(SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区(SRAM)。

示例: `COMF REG, 0, 0`

指令执行前, WREG = 0x00, REG = 0x55;

指令执行后, WREG = 0xAA, REG = 0x55。

#### 4.15 CPFSEQ - 比较 f 和 WREG, 如果 f = WREG 则跳过下条指令

语法: `CPFSEQ f, a`

操作数:  $0 \leq f \leq 255$

$a \in [0, 1]$

操作:  $(f) - \text{WREG}$ , 如果  $f = \text{WREG}$  则跳过下条指令 (无符号比较)

说明: 通过执行无符号的减法, 将寄存器 f 的内容与 WREG 进行比较, 相等则跳过下条指令, 即丢弃下一条指令而执行一条 NOP 指令。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: `CFPSEQ REG, 1`

`BSF REG, 6, 1`

`BCF REG, 0, 1`

指令执行前, WREG = 0x00, REG = 0x00;

指令执行后, WREG = 0x00, REG = 0x00。

#### 4.16 DECF - f 减 1

语法: `DECF f, d, a`

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) - 1 \rightarrow \text{dest}$

说明: 将寄存器 f 的内容减 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: `DECF REG, 1, 1`

指令执行前, WREG = 0x00, REG = 0x00;

指令执行后, WREG = 0x00, REG = 0xFF。

#### 4.17 DECFSZ - f 减 1, 为 0 则跳过下条指令

语法: DECFSZ f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) - 1 \rightarrow \text{dest}$ , 结果为 0 时跳过下条指令

说明: 将寄存器 f 的内容减 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果结果为 0, 则跳过下条指令, 即丢弃下条指令, 改为执行 NOP。如果 a 为 0, 选择快速操作存储区(SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区(SRAM)。

示例: DECFSZ REG, 0, 1

BSF REG1, 7, 0

...

指令执行前, WREG = 0x00, REG = 0xFF, REG1 = 0x00;

指令执行后, WREG = 0xFE, REG = 0xFF, REG1 = 0x80。

#### 4.18 DCFSNZ - f 减 1, 不为 0 则跳过下条指令

语法: DCFSNZ f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) - 1 \rightarrow \text{dest}$ , 结果不为 0 时跳过下条指令

说明: 将寄存器 f 的内容减 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果结果不为 0, 则跳过下条指令, 即丢弃下条指令, 改为执行 NOP。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: DCFSNZ REG, 1, 1

BSF REG1, 7, 0

...

指令执行前, WREG = 0xFF, REG = 0x01, REG1 = 0x00;

指令执行后, WREG = 0xFF, REG = 0x00, REG1 = 0x00。

#### 4.19 GOTO - 无条件跳转

语法: GOTO n

操作数:  $0 \leq n \leq 1048575$

操作:  $n \rightarrow PC\langle 20:1 \rangle$

说明: GOTO 指令运行无条件跳转到整个 2MB 存储范围中的任何位置。  
将 20 位值  $n$  装入  $PC\langle 20:1 \rangle$ 。

示例: Here:

GOTO     There

指令执行前,  $PC = \text{地址(Here)}$ ;

指令执行后,  $PC = \text{地址(There)}$ 。

#### 4.20 INCF - f 加 1

语法: INCF f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) + 1 \rightarrow \text{dest}$

说明: 将寄存器 f 的内容加 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: INCF     REG, 1, 1

指令执行前,  $WREG = 0xFF$ ,  $REG = 0xFF$ ;

指令执行后,  $WREG = 0xFF$ ,  $REG = 0x00$ 。

#### 4.21 INCFSZ - f 加 1, 为 0 则跳过下条指令

语法: INCFSZ f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) + 1 \rightarrow \text{dest}$ , 结果为 0 时跳过下条指令

说明: 将寄存器 f 的内容加 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果结果为 0, 则跳过下条指令, 即丢弃下条指令, 改为执行 NOP。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: INCFSZ   REG, 0, 1

BSF        REG1, 7, 0

...

指令执行前, WREG = 0x00, REG = 0xFE, REG1 = 0x00;

指令执行后, WREG = 0xFF, REG = 0xFE, REG1 = 0x80。

#### 4.22 INFSNZ - f 加 1, 不为 0 则跳过下条指令

语法: INFSNZ f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作:  $(f) - 1 \rightarrow \text{dest}$ , 结果不为 0 时跳过下条指令

说明: 将寄存器 f 的内容加 1。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果结果不为 0, 则跳过下条指令, 即丢弃下条指令, 改为执行 NOP。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: INFSNZ REG, 1, 1

BSF REG1, 7, 0

...

指令执行前, WREG = 0x55, REG = 0x00, REG1 = 0x00;

指令执行后, WREG = 0x55, REG = 0x01, REG1 = 0x00。

#### 4.23 IORLW - 将立即数与 W 作逻辑或运算

语法: IORLW k

操作数:  $0 \leq k \leq 255$

操作:  $(\text{WREG}) \mid k \rightarrow \text{WREG}$

说明: 将 WREG 寄存器的内容与 8 位立即数 k 相或, 结果保存在 WREG 寄存器。

示例: IORLW 0xAA

指令执行前, WREG = 0x55;

指令执行后, WREG = 0xFF。

#### 4.24 IORWF - 将 WREG 和 f 作逻辑或运算

语法: IORWF f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作: (WREG) | (f)  $\rightarrow$  dest

说明: 将 WREG 的内容和 f 寄存器的内容相或。如果 d 为 0，结果存储到 WREG 中，如果 d 为 1，结果存回寄存器 f（默认）。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例: IORWF REG, 1, 0

指令执行前，WREG = 0x55，REG = 0xAA；

指令执行后，WREG = 0x55，REG = 0xFF。

#### 4.25 MOVF - 移动 f

语法: MOVF f, d, a

操作数:  $0 \leq f \leq 255$

$d \in [0, 1]$

$a \in [0, 1]$

操作: (f)  $\rightarrow$  dest

说明: 移动寄存器 f，如果 d 为 0，结果存储到 WREG 中，如果 d 为 1，结果存回寄存器 f（默认）。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例: MOVF REG, 0, 1

指令执行前，WREG = 0x55，REG = 0xAA；

指令执行后，WREG = 0xAA，REG = 0xAA。

#### 4.26 MOVLB - 将立即数移入 BSR 的低半字节

语法: MOVLB k

操作数:  $0 \leq k \leq 255$

操作: k  $\rightarrow$  BSR

说明: 将 8 位立即数 k 移入存储区选择寄存器 BSR，不管 k7:k4 的值如何，BSR<7:4>的值保持为 0。

示例: MOVLB 0x01

指令执行前，BSR = 0x00；

指令执行后，BSR = 0x01。

#### 4.27 MOVLW - 将立即数移入 WREG

语法: MOVLW k

操作数:  $0 \leq k \leq 255$

操作:  $k \rightarrow \text{WREG}$

说明: 将 8 位立即数  $k$  移入工作寄存器 WREG。

示例: `MOVLW 0x01`

指令执行前,  $\text{WREG} = 0x00$ ;

指令执行后,  $\text{WREG} = 0x01$ 。

#### 4.28 MOVWF - 将 WREG 内容移入 f

语法: `MOVWF f, a`

操作数:  $0 \leq f \leq 255$

$a \in [0, 1]$

操作:  $(\text{WREG}) \rightarrow f$

说明: 将 WREG 中的数据移入寄存器 f。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: `MOVWF REG, 1`

指令执行前,  $\text{WREG} = 0x55$ ,  $\text{REG} = 0x00$ ;

指令执行后,  $\text{WREG} = 0x55$ ,  $\text{REG} = 0x55$ 。

#### 4.29 NOP - 执行空操作

语法: `NOP`

操作数: 无

操作: 空操作

说明: 不进行任何操作。

示例: 无。

#### 4.30 RCALL - 相对调用

语法: `RCALL n`

操作数:  $-1024 \leq n \leq 1023$

操作:  $(\text{PC}) + 2 \rightarrow \text{TOS}$

$(\text{PC}) + 2 + 2n \rightarrow (\text{PC})$

说明: 从当前地址跳转 (最多 1K) 来调用子程序。首先, 将返回地址  $(\text{PC} + 2)$  压入返回堆栈, 然后, 将二进制补码 “ $2n$ ” 与 PC 相加, 因为 PC 要先递增才能取下条指令, 因此新的地址将为  $\text{PC} + 2 + 2n$ 。

示例: Here:



RCALL          There

指令执行前, PC = 地址(Here);

指令执行后, PC = 地址(There)

TOS = 地址(Here +2)

#### 4.31 RESET - 复位

语法:            RESET

操作数:          无

操作:            将所有受复位影响的寄存器和标志位复位, 0x00 → PC。

说明:            软件复位。

示例:            Here:

RESET

指令执行前, PC = 地址 (Here);

指令执行后, PC = 0x00

寄存器 = 复位值

标志位 = 复位值

#### 4.32 RETFIE - 从中断返回

语法:            RETFIE {s}

操作数:           $s \in [0, 1]$

操作:            (TOS) → PC

如果  $s = 1$ :

(WREGS) → WREG

(STATUS) → STATUS

(BSRS) → BSR

如果  $s = 0$  (默认), 则不更新。

PCLATU 和 PCLATH 保持不变。

说明:            从中断返回, 执行出栈操作, 将栈顶(TOS)的内容装入 PC。

通过将全局中断使能位置 1, 来重新使能中断。

示例:            无

#### 4.33 RETLW - 将立即数返回给 WREG

语法:            RETLW    k

操作数:  $0 \leq k \leq 225$

操作:  $k \rightarrow WREG$   
 $(TOS) \rightarrow PC$   
PCLATU 和 PCLATH 保持不变。

说明: 将 8 位立即数 k 装入 WREG, 将栈顶(TOS)的内容装入 PC。

示例: 无

#### 4.34 RETURN - 从子程序返回

语法: RETURN {s}

操作数:  $s \in [0, 1]$

操作:  $(TOS) \rightarrow PC$   
如果  $s = 1$ :  
 $(WREGS) \rightarrow WREG$   
 $(STATUSS) \rightarrow STATUS$   
 $(BSRS) \rightarrow BSR$   
PCLATU 和 PCLATH 保持不变。  
如果  $s = 0$  (默认), 则不更新。

说明: 从子程序返回, 执行出栈操作, 将栈顶(TOS)的内容装入 PC。

示例: 无

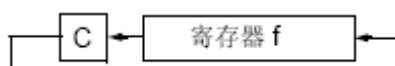
#### 4.35 RLCF - f 带进位循环左移

语法: RLCF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$   
 $a \in [0, 1]$

操作:  $(f\langle n \rangle) \rightarrow dest\langle n+1 \rangle$   
 $(f\langle 7 \rangle) \rightarrow C$   
 $(C) \rightarrow dest\langle 0 \rangle$

说明: 将寄存器 f 的内容连同进位标志位一起循环左移 1 位, 如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。



示例: RLCF REG, 0, 0

指令执行前, C = 0, WREG = 0x00, REG = 0xAA(1010 1010);

指令执行后, C = 1, WREG = 0x55(0101 0101), REG = 0xAA。

#### 4.36 RLNCF - f 循环左移 (不带进位)

语法: RLNCF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$   
 $a \in [0, 1]$

操作:  $(f\langle n \rangle) \rightarrow \text{dest}\langle n+1 \rangle$   
 $(f\langle 7 \rangle) \rightarrow \text{dest}\langle 0 \rangle$

说明: 将寄存器 f 的内容循环左移 1 位, 如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。



示例: RLNCF REG, 1, 0

指令执行前, WREG = 0x00, REG = 0xAA(1010 1010);

指令执行后, WREG = 0x00, REG = 0x55(0101 0101)。

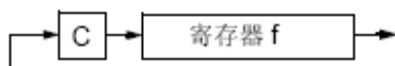
#### 4.37 RRCF - f 带进位循环右移

语法: RRCF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$   
 $a \in [0, 1]$

操作:  $(f\langle n \rangle) \rightarrow \text{dest}\langle n-1 \rangle$   
 $(f\langle 0 \rangle) \rightarrow C$   
 $(C) \rightarrow \text{dest}\langle 7 \rangle$

说明: 将寄存器 f 的内容连同进位标志位一起循环右移 1 位, 如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。



示例: RRCF REG, 0, 1

指令执行前, C = 1, WREG = 0x00, REG = 0xAA(1010 1010);

指令执行后, C = 0, WREG = 0xD5(1101 0101), REG = 0xAA。

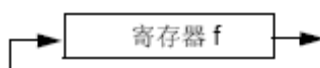
#### 4.38 RRNCF - f 循环右移 (不带进位)

语法: RRNCF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$   
 $a \in [0, 1]$

操作:  $(f\langle n \rangle) \rightarrow \text{dest}\langle n-1 \rangle$   
 $(f\langle 0 \rangle) \rightarrow \text{dest}\langle 7 \rangle$

说明: 将寄存器 f 的内容循环右移 1 位, 如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。



示例: RRNCF REG, 1, 0

指令执行前, WREG = 0x00, REG = 0xAA(1010 1010);

指令执行后, WREG = 0x00, REG = 0x55(0101 0101)。

#### 4.39 SETF - 将 f 的内容置为全 1

语法: SETF f, a

操作数:  $0 \leq f \leq 255$   
 $a \in [0, 1]$

操作:  $0xFF \rightarrow f$

说明: 将寄存器 f 的内容置为 0xFF。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: SETF REG, 0

指令执行前, REG = 0x00;

指令执行后, REG = 0xFF。

#### 4.40 SLEEP - 进入休眠模式

语法: SLEEP

操作数: 无

操作: 0x00 → WDT  
MCU 停止工作。

说明: 如果设置为睡眠模式 (OSCCON 寄存器的 IDLEN 为 0), 则 MCU 停止工作, 但振荡器仍然振荡;  
如果设置为空闲模式 (IDLEN 为 1), 则 MCU 停止工作, 振荡器停止振荡。

示例: 无。

#### 4.41 SUBLW - 立即数减去 WREG

语法: SUBLW k

操作数:  $0 \leq k \leq 255$

操作:  $k - (WREG) \rightarrow WREG$

说明: 将 8 位立即数 k 减去 WREG, 结果保存在 WREG 寄存器。

示例: SUBLW 0x15  
指令执行前, WREG = 0x14;  
指令执行后, WREG = 0x01;

#### 4.42 SUBWF - f 减去 WREG (不带借位)

语法: SUBWF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$   
 $a \in [0, 1]$

操作:  $(f) - (WREG) \rightarrow \text{dest}$

说明: 将 f 寄存器的内容减去 WREG 中的内容。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: SUBWF REG, 0, 0  
指令执行前, WREG = 0x17, REG = 0xC9;  
指令执行后, WREG = 0xB2, REG = 0xC9。

#### 4.43 SUBWFB - f 减去 WREG (带借位)

语法: SUBWFB f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1]$

$a \in [0, 1]$

操作: (f) - (WREG) - ( $\sim C$ )  $\rightarrow$  dest

说明: 将 f 寄存器的内容减去 WREG 的内容和进位（借位）（通过二进制补码进行运算）。如果 d 为 0，结果存储到 WREG 中，如果 d 为 1，结果存回寄存器 f（默认）。如果 a 为 0，选择快速操作存储区（SFR），如果 a 为 1，使用 BSR 选择 GPR 存储区（SRAM）。

示例: SUBWFB REG, 1, 1

指令执行前, C = 1, WREG = 0x02, REG = 0x05;

指令执行后, C = 1, WREG = 0x02, REG = 0x03。

#### 4.44 TBLRD

语法: TBLRD \*

操作数: 无

操作: (程序存储区(TBLPTR))  $\rightarrow$  TABLAT

说明: 该指令用于读取程序存储器的内容，使用表指针(TBLPTR)对程序存储器进行寻址。

TBLPTR(21 位指针)，寻址范围为 2MB。

示例: MOVLW 0x00

MOVWF TBLPTRU, 0 ;假设程序地址不超过 16 位

MOVLW high(Here) ;取高字节

MOVWF TBLPTRH, 0

MOVLW low(Here) ;取低字节

MOVWF TBLPTRL, 0

TBLRD \*

MOVF TABLAT, 0, 0

NOP

...

Here:

db 0x55 ;定义字节 0x55 在 Here 地址处

指令执行前, WREG = 0x00, TABLAT = 0x00;

指令执行后, WREG = 0x55, TABLAT = 0x55。

#### 4.45 TSTFSZ - 测试 f，为 0 则跳过后条指令

语法: TSTFSZ f, a

操作数:  $0 \leq f \leq 255$   
 $a \in [0, 1]$

操作: 如果  $f = 0x00$  则跳过下条指令。

说明: 如果  $f = 0x00$ , 丢弃下一条指令而执行一条 NOP 指令。如果  $a$  为 0, 选择快速操作存储区 (SFR), 如果  $a$  为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: TSTFSZ REG, 1  
 BSF REG, 6, 1  
 BCF REG, 0, 1  
 指令执行前, REG = 0x00;  
 指令执行后, REG = 0x00。

#### 4. 46 XORLW - 立即数与 WREG 做逻辑异或运算

语法: XORLW k

操作数:  $0 \leq k \leq 255$

操作:  $(WREG) \wedge k \rightarrow WREG$

说明: 将 WREG 寄存器的内容与 8 位立即数 k 相异或, 结果保存在 WREG 寄存器。

示例: XORLW 0xAF  
 指令执行前, WREG = 0xB5;  
 指令执行后, WREG = 0x1A。

#### 4. 47 XORWF - 将 WREG 和 f 作逻辑异或运算

语法: XORWF f, d, a

操作数:  $0 \leq f \leq 255$   
 $d \in [0, 1], a \in [0, 1]$

操作:  $(WREG) \wedge (f) \rightarrow dest$

说明: 将 WREG 的内容和 f 寄存器的内容相异或。如果 d 为 0, 结果存储到 WREG 中, 如果 d 为 1, 结果存回寄存器 f (默认)。如果 a 为 0, 选择快速操作存储区 (SFR), 如果 a 为 1, 使用 BSR 选择 GPR 存储区 (SRAM)。

示例: XORWF REG, 1, 0  
 指令执行前, WREG = 0xB5, REG = 0xAF;  
 指令执行后, WREG = 0xB5, REG = 0x1A。

## 5. 汇编器简介

### 5.1 汇编器语法说明

- ✚ 一个指令代码必须在同一行中描述完毕；
- ✚ 汇编指令不能在每行的起始处编写，至少在行首留有一个空格符，用 Tab 键保留多个更佳；
- ✚ 程序跳转用的语句标号和用户定义的变量符号必须顶格，语句标号后的“:”可选；
- ✚ 任何标号或变量不能以数字开头；
- ✚ 汇编器内的保留字（指令码或伪指令）其大小写一视同仁；
- ✚ 程序中立即数字的描述方法有以下几种：
  - 16 进制：0x12、0xFF 或 12h、0FFh 或 H ‘1234’、H ‘FFFF’；
  - 10 进制：.123 或 d ‘123’；
  - 2 进制：b ‘10100101’
- ✚ 注解信息用“;”（分号）引导；
- ✚ 源程序中必须出现伪指令 end，代表汇编结束。

### 5.2 伪指令

- ✚ #include 或 include：包含头文件和其它文件用，例如#include “SD3003.INC”，又例如#include “sqrt.asm”；
- ✚ #define/#undefine：#define 作用是定义常数符号，例如#define DELAY 100，又例如#define KEY1 PROT1,7,0；而#undefine 可以注销一个已定义的符号变量；
- ✚ equ 和#define 类似；
- ✚ cblock/endc：需成对使用，给多个变量分配地址，cblock 声明变量块的起始地址，endc 声明变量块定义结束；  
例如：

```
cblock      0x120
    w_temp          ;0x120
    buffer:8        ;0x121~0x128
    var              ;0x129
endc
```
- ✚ org：定义程序代码的绝对地址，一般用来定义程序开头地址，例如 org 0x0000；
- ✚ \$：取当前地址，例如 bra \$即在本地址循环；
- ✚ High/low：取 16 位立即数的高、低字节；
- ✚ dw/db：dw 为定义 1 个字在程序空间，db 为定义 1 个字节在程序空间，例如 dw 0x55AA，又例如 db 0x5A；
- ✚ end：汇编结束。

### 5.3 宏定义

使用 macro/endm 伪指令来定义宏定义。

例如，定义宏指令实现寄存器和立即数比较大小。

```
FL_JGE      macro    filereg , litval , jumpto
              MOVLW  litval
              SUBWF  filereg , 0,1
```



```

        BTFSC    STATUS,C,0
        GOTO     jumpto
    endm

```

定义完成后，就可以使用如下：

```

Val1    equ    0x20
        FL_JGE      val1 , .100 , val1_over
        nop
        ...
Val1_over:
        Nop
        ...

```