

第四章 AVR 单片机指令系统

说明: 为了使读者和用户迅速掌握 AVR 指令系统的功能,边学习,边实践,希望大家先学习<<第三章 AVR 开发工具>>。根据我们的实际教学经验,有的书籍是根据英文原文翻译,程序及说明可能不合中国人习惯,又由于印刷等多种原因,内容有出入,学起来较难。我们是参考有关资料,并在实际工作中验证,并编写有关测试程序(含中文注释),在模拟调试软件窗口观察通过,或在实时仿真器或在 SL-AVR 下载开发下载实验器上验证通过,把测试实验程序刻在光盘上,保证用户学习、实验时少走弯路。所以我们先学习系统软件的使用,然后学指令系统,用户一边学习 AVR 指令系统,一边学习系统软件编程调试,这样使指令功能流向看得见摸得着,学习起来有声有色,达到事半功倍的效果。当学完所有指令,你也学会了用软件编程开发调试。我们的想法希望你能去边学边实践,并得到你的认可,我们就谢谢了。

AVR 单片机每条指令对应的实验源程序见文件夹<<指令 ASM>>

计算机的指令系统是一套控制计算机操作的代码,称之为机器语言。计算机只能识别和执行机器语言的指令。为了便于人们理解、记忆和使用,通常用汇编语言指令来描述计算机的指令系统。汇编语言指令可通过汇编器翻译成计算机能识别的机器语言。

AVR 单片机指令系统是 RISC 结构的精简指令集,是一种简明易掌握、效率高的指令系统。

AVR 单片机指令系统速查表,不同器件使用不同的指令表,见附录 3:

- (1) 89 条指令器件(AT90S1200,最基本指令);
- (2) 90 条指令器件(□):Atiny11/12/15/22; 90 条指令=□+89 条基本指令
- (3) 118 条指令器件(◇):AT90S2313/2323/2343/2333,/4414/4433/4434/8515/90S8534/8535
;118 条指令=◇+ 90 条;
- (4) 121 条指令器件(△)ATmega603/103; 121 条指令=△+ 118 条;
- (5) 130 条指令器件(☆)ATmega161; 130 条指令=☆+121 条

AVR 大多数执行时间为单个时钟周期。这一章主要分析 AVR 单片机指令系统的功能和使用方法。下表为常用 AVR 器件指令表:

AVR 器件 (指令速查表) 118 条指令器件
 AT90S2313/2323/2343/2333 ,AT90S4414/4433/4434/8515,AT90S8534/8535

算术和逻辑指令		BRCC k	C 清零转	位指令和位测试指令	
ADD Rd,Rr	加法	BRSH k	≥ 转	SBI P,b	置位 I/O 位
ADC Rd,Rr	带进位加	BRLO k	小于转(无符号)	CBI P,b	清零 I/O 位
◇ ADIW RdI,K	加立即数	BRMI k	负数转移	LSL Rd	左移
SUB Rd,Rr	减法	BRPL k	正数转移	LSR Rd	右移
SUBI Rd,Rr	减立即数	BRGE k	≥ 转(带符号)	ROL Rd	带进位左循环
SBC Rd,Rr	带进位减	BRLT k	小于转(带符号)	ROR Rd	带进位右循环
SBCI Rd,K	带 C 减立即数	BRHS k	H 置位转移	ASR Rd	算术右移
◇ SBIW RdI,K	减立即数	BRHC k	H 清零转移	SWAP Rd	半字节交换
AND Rd,Rr	与	BRTS k	T 置位转移	BSET s	置位 SREG
ANDI Rd,K	与立即数	BRTC k	T 清零转移	BCLR s	清零 SREG
OR Rd,Rr	或	BRVS k	V 置位转移	BST Rr,b	Rr 的 b 位送 T
ORI Rd,K	或立即数	BRVC k	V 清零转移	BLD Rd	T 送 Rr 的 b 位
EOR Rd,Rr	异或	BRIE k	中断位置位转移	SEC	置位 C
COM Rd	取反	BRID k	中断位清零转移	CLC	清零 C
NEG Rd	取补	数据传送指令		SEN	置位 N
SBR Rd,K	寄存器位置位	MOV Rd,Rr	寄存器传送	CLN	清零 N
CBR Rd,K	寄存器位清零	◇ LDI Rd,Rr	装入立即数	SEZ	置位 Z
INC Rd	加 1	◇ LD Rd, X	X 间接取数	CLZ	清零 Z
DEC Rd	减 1	◇ LD Rd, X+	X 间接取数后+	SEI	置位 I
TST Rd	测试零或负	◇ LD Rd, -X	X 间接取数先-	CLI	清零 I
CLR Rd	寄存器清零	◇ LD Rd, Y	Y 间接取数	SES	置位 S
SER Rd	寄存器置 FF	◇ LD Rd, Y+	Y 间接取数后+	CLS	清零 S
条件转移指令		◇ LD Rd, -Y	Y 间接取数先-	SEV	置位 V
RJMP k	相对转移	◇ LDD Rd, Y+q	Y 间接取数 + q	CLV	清零 V
◇ IJMP	间接转移(Z)	LD Rd, Z	Z 间接取数	SET	置位 T
RCALL k	相对调用	◇ LD Rd, Z+	Z 间接取数后+	CLT	清零 T
◇ ICALL	间接调用(Z)	◇ LD Rd, -Z	Z 间接取数先-	SEH	置位 H
RET	子程序返回	◇ LDD Rd, Z+q	Z 间接取数 + q	CLH	清零 H
RETI	中断返回	◇ LDS Rd,K	从 SRAM 装入	NOP	空操作
CPSE Rd,Rr	比较相等跳行	◇ ST X, Rr	X 间接存数	SLEEP	休眠指令
CP Rd,Rr	比较	◇ ST X+, Rr	X 间接存数后+	WDR	看门狗复位
CPC Rd,Rr	带进位比较	◇ ST -X, Rr	X 间接存数先-	90 条指令为 Attiny11/12/15/22= □+89 条基本指令是 AT90S1200	
CPI Rd,K	与立即数比较	◇ ST Y, Rr	Y 间接存数		
SBRC Rr,b	位清零跳行	◇ ST Y+, Rr	Y 间接存数后+		
SBRS Rr,b	位置位跳行	◇ ST -Y, Rr	Y 间接存数先-		
SBIC P,b	I/O 位清零跳行	◇ STD Y+q, Rr	Y 间接存数 + q		
SBIS P,b	I/O 位置位跳行	ST Z, Rr	Z 间接存数		
BRBS s,k	SREG 位置位转	◇ ST Z+, Rr	Z 间接存数后+	118 条指令器件= ◇+ 90 条指令器件	
BRBC s,k	SREG 位清零转	◇ ST -Z, Rr	Z 间接存数先-		
BREQ k	相等转移	◇ STD Z+q, Rr	Z 间接存数+q		
BRNE k	不相等转移	◇ STS k, Rr	数据送 SRAM		
BRCS k	C 置位转	□ LPM	从程序区取数		
		IN Rd, P	从 I/O 口取数		
		OUT P, Rdr	存数 I/O 口		
		PUSH Rr	压栈		
		POP Rd,	出栈		

4.1 指令格式

4.1.1 汇编指令

汇编语言源文件是由汇编语言代码和汇编程序指令所组成的 ASCII 字符文件。

一、汇编语言源文件

汇编语言源文件包括指令助记符、标号和伪指令。指令助记符和伪指令常带操作数。

每条程序输入行首先是标号,标号为字母数字串,并带一个冒号。使用标号的目的是为了跳转和转移指令及在程序存储器和 SRAM 中定义变量名。

程序输入行有下列四种形式:

- (1) 【标号:】 伪指令 【操作数】 【注释义】
- (2) 【标号:】 指令 【操作数】 【注释】
- (3) 注释
- (4) 空行

注释有下列形式: 【文字】

括号内的项是任选的。用于注释的分号 (;) 及到行结尾的文字,汇编器是忽略的。标号、指令和伪指令在后面有详细说明。

例子:

```
Label: .EQU Var1=100    ; 置 Var1 等于 100 (伪指令)
      .EQU Var2=200    ; 置 Var2 等于 200
test:  rjmp test      ; 无限循环 (指令)
      ; 纯注释行
      ; 另一个注释行
```

注意: 不限制有关标号、伪指令、注释或指令的列位置。

二、指令助记符

汇编器认可指令集中的指令助记符。指令集中综合了助记符并给出了参数。

操作数有下列形式:

Rd: R0~R31 或 R16~R31 (取决于指令)。

Rr: R0~R31。

b: 常数 (0~7), 可能是常数表达式。

S: 常数 (0~7), 可能是常数表达式。

p: 常数 (0~31 / 63) 可能是常数表达式。

K: 常数 (0~255) 可能是常数表达式。

k: 常数, 值范围取决于指令, 可能是常数表达式。

q: 常数 (0~63), 可能是常数表达式。

4.1.2 汇编器伪指令

汇编器提供一些伪指令,伪指令并不直接转换成操作数,而是用于调整存储器中程序的位置、定义宏、初始化存储器等。全部伪指令在表 4.2 中给出。

1. BYTE——保存字节到变量

BYTE 伪指令保存存储的内容到 SRAM 中。为了能提供所要保存的位置, BYTE 伪指令前应有标号。该伪指令带一个表征被保存字节数的参数。该伪指令仅用在数据段内 (见伪指令 CSEG, DSEG 和 ESEG)。注意: 必须带一个参数, 字节数的位置不需要初始化。

语法: LABEL: . BYTE 表达式

2. CSEG——代码段

CSEG 伪指令定义代码段的开始位置。一个汇编文件包含几个代码段，这些代码段在汇编时被连接成一个代码段。在代码段中不能使用 BYTE 伪指令，典型的缺省段为代码段。代码段有一个字定位计数器。ORG 伪指令用于放置代码段和放置程序存储器指定位置的常数。

CSEG 伪指令不带参数。

语法：. CSEG

表 4.2 伪指令表

序号	伪指令	说 明	序号	伪指令	说 明
1	BYTE	保存字节到变量	10	ESEG	E2PROM 段
2	CSEG	代码段	11	EXIT	退出文件
3	DB	定义字节常数	12	INCLUDE	从指定文件开始读
4	DEF	设置寄存器的符号名	13	LIST	打开列表文件
5	DEVICE	定义被汇编的器件	14	LISTMAC	打开宏表达式
6	DSEG	数据段	15	MACRO	宏开始
7	DW	定义字常数	16	NOLIST	关闭列表文件
8	ENDMACRO	宏结束	17	ORG	设置程序起始位置
9	EQU	符号相等于表达式	18	SET	赋值给一个标号

3. DB——在程序存储器或 E2PROM 存储器中定义字节常数

DB 伪指令保存数据到程序存储器或 E2PROM 存储器中。为了提供被保存的位置，在 DB 伪指令前必须有标号。DB 伪指令可带一个表达式表，至少有一个表达式。DB 伪指令必须放在代码段或 E2PROM 段。表达式表是一系列表达式，用逗号分隔。每个表达式必须是一 128~255 之间的有效值。如果表达式有效值是负数，则用 8 位 2 的补码放在程序存储器或 E2PROM 存储器中。如果 DB 伪指令用在代码段，并且表达式表多于一个表达式，则以两个字节组合成一个字放在程序存储器中。如果表达式表是奇数，那么最后一个表达式将独自以字格式放在程序存储器中，而不管下一行汇编代码是否是单个 DB 伪指令。

语法：LABEL: . DB 表达式

4. DEF——设置寄存器的符号名

DEF 伪指令允许寄存器用符号代替。一个定义的符号用在程序中，并指定一个寄存器，一个寄存器可以赋几个符号。符号在后面程序中能再定义。

语法：. DEF 符号—寄存器

5. DEVICE——定义被汇编的器件

DEVICE 伪指令允许用户告知编译器被执行的代码使用那种器件。如果使用该伪指令，若在代码中有指定的器件不提供的指令，则提示一个警告。如果代码段或 E2PROM 段的尺寸大于被指定器件的尺寸，也提示警告。如果不使用 DEVICE 伪指令，则假定器件提供所有的指令，也不限制存储器尺寸。

语法：.DEVICE AT90S1200 AT90S2313 AT90S4414 AT90S8515

6. DSEG 一数据段

DSEG 伪指令定义数据段的开始。一个汇编文件能包含几个数据段，这些数据段在汇编时被连接成一个数据段。一个数据段正常仅由 BYTE 伪指令（和标号）组成。数据段有自己的定位字节计数器。ORG 伪指令被用于在 SRAM 指定位置放置变量。DSEG 伪指令不带参数。

语法：.DSEG

7. DW——在程序存储器和 E2PROM 存储器中定义字常数

DW 伪指令保存代码到程序存储器或 E2PROM 存储器，为了提供被保存的位置，在 DW 伪指令前必须有标号。DW 伪指令可带一个表达式表，至少有一个表达式。DW 伪指令必须放在代码段或 E2PROM 段。表达式表是一系列表达式，用逗号分隔。每个表达式必须是一 32 768~65 535 之间的有效值。如果表达式有效值是负数，则用 16 位 2 的补码放在程序存储器中。

语法: LABEL: .DW 表达式表

8. ENDMACRO —宏结束

ENDMACRO 伪指令定义宏定义的结束。该伪指令并不带参数, 参见 MACRO 宏定义伪指令。

语法: .ENDMACRO

9. EQU—设置一个符号相等于一个表达式

EQU 伪指令赋一个值到标号, 该标号用于后面的表达式, 用 EQU 伪指令赋值的标号是一个常数, 不能改变或重定义。

语法: .EQU 标号= 表达式

10. ESEG - E2PROM 段

ESEG 伪指令定义 E2PROM 段的开始位置。一个汇编文件包含几个 E2PROM 段, 这些 E2PROM 段在汇编时被连接成一个 E2PROM 段。在 E2PROM 段中不能使用 BYTE 伪指令。E2PROM 段有一个字节定位计数器。ORG 伪指令用于放置 E2PROM 存储器指定位置的常数。ESEG 伪指令不带参数。

语法: .ESEG

11. EXIT—退出文件

EXIT 伪指令告诉编译器停止汇编该文件。正常情况下, 编译器汇编到文件的结束。如果 EXIT 出现在包括文件中, 则编译器从文件中 INCLUDE 伪指令行继续汇编。

语法: .EXIT

12. INCLUDE—包括另外的文件

INCLUDE 伪指令告诉编译器从指定的文件开始读。然后编译器汇编指定的文件, 直到文件结束或遇到 EXIT 伪指令。一个包括文件也可能自己用 INCLUDE 伪指令来表示。

语法: .INCLUDE "文件名"

13. LIST—打开列表文件生成器

LIST 伪指令告诉编译器打开列表文件生成器。编译器生成一个汇编源代码、地址和操作代码的文件列表。列表文件生成器缺省值是打开。该伪指令总是与 NOLIST 伪指令一起出现, 用于生成列表或汇编源文件有选择的列表。

语法: .LIST

14 LISTMAC—打开宏表达式

LISTMAC 伪指令告诉编译器, 当调用宏时, 用列表生成器在列表文件中显示宏表达式。缺省值仅是在列表文件中显示宏调用参数。

语法: .LISTMAC

15. MACRO —宏开始

MACRO 伪指令告诉编译器这是宏开始。MACRO 伪指令带宏名和参数。当后面的程序中写了宏名, 被表达的宏程序在指定位置被调用。一个宏可带 10 个参数。这些参数在宏定义中用 @0~@9 代表。当调用一个宏时, 参数用逗号分隔。宏定义用 ENDMACRO 伪指令结束。缺省值为编译器的列表生成器, 仅列表宏调用。为了在列表文件中包括宏表达式, 必须使用 LISTMAC 伪指令。在列表文件的操作代码域内宏用 at 作记号。

语法: .MACRO 宏名

16. NOLIST--关闭列表文件生成器

NOLIST 伪指令告诉编译器关闭列表文件生成器。正常情况下, 编译器生成一个汇编源代码、地址和操作代码文件列表。缺省时为打开列表文件, 但是可用该伪指令禁止列表。为了使被选择的汇编源文件部分产生列表文件, 该伪指令可以与 LIST 伪指令一起使用。

语法: .NOLIST

17. ORG —设置程序起始位置

ORG 伪指令设置定位计数器一个绝对值。设置的值为一个参数。如果 ORG 伪指令放在数据段, 则设置 SRAM 定位计数器; 如果该伪指令放在代码段, 则设置程序存储器计数器; 如果该伪指令放在 E2PROM 段, 则设置 E2PROM 定位计数器。如果该伪指令前带标号 (在相同的源代码行), 则标号由参数值给出。代码和 E2PROM 定位计数器的缺省值是零, 而当汇编启

动时，SRAM 定位计数器的缺省值是 32（因为寄存器占有地址为 0~ 31）。注意，E2PROM 和 SRAM 定位计数器按字节计数，而程序存储器定位计数器按字计数。

语法：.ORG 表达式

18. SET—设置一个与表达式相等的符号

SET 伪指令赋值给一个标号。这个标号能用在后面的表达式中。用 SET 伪指令赋值的标号在后面的程序中能改变。

语法：.SET 标号 = 表达式

4.1.3 表达式

汇编器包括一些表达式，表达式由操作数、运算符和函数组成。所有的表达式内部为 32 位。

一、操作数

- (1) 用户定义的标号，该标号给出了放置标号位置的定位计数器的值。
- (2) 用户用 SET 伪指令定义的变量。
- (3) 用户用 EQU 伪指令定义的常数。
- (4) 整数常数，包括下列几种形式：
 - 十进制（缺省值）：10, 255
 - 十六进制数（二进制表示法）：0x0a,\$0a,0xff,\$ff
 - 二进制数：0b00001010,0b11111111
- (5) PC：程序存储器定位计数器的当前值。

二、函数

- (1) LOW（表达式）返回一个表达式的低字节。
- (2) HIGH（表达式）返回一个表达式的第二个字节。
- (3) BYTE2（表达式）与 HIGH 函数相同。
- (4) BYTE3（表达式）返回一个表达式的第三个字节。
- (5) BYTE4（表达式）返回一个表达式的第四个字节。
- (6) LWRD（表达式）返回一个表达式的 0~ 15 位。
- (7) HWRD（表达式）返回一个表达式的 16~ 31 位。
- (8) PAGE（表达式）返回一个表达式的 16~ 21 位。
- (9) EXP2（表达式）返回 2^{\wedge} 表达式。
- (10) LOG2（表达式）返回 LOG2（表达式）的整数部分。

三、运算符

汇编器提供的部分运算符见表 3.3。越高的运算符，优先级越高。表达式可以用括号括起来，并且与括号外任意表达式所组合的表达式总是有效的。

表 4.3 部分运算符表

序号	名称	符号	优先级	说明
1	逻辑非	!	14	一元运算符,表达式是 0 返回 1,而表达式为非 0 则返回 0
2	逐位非	~	14	一元运算符,输入表达式的所有位倒置
3	负号	-	14	一元运算符,使表达式为算术负
4	乘法	*	13	二进制运算符,两个表达式相乘
5	除法	/	13	二进制运算符,左边表达式除以右边表达式,得整数的商值
6	加法	+	12	二进制运算符,两个表达式相加
7	减法	-	12	二进制运算符,左边表达式减去右边表达式
8	左移	<<	11	二进制运算符,左边表达式左移右边表达式给出的次数
9	右移	>>	11	二进制运算符,左边表达式右移右边表达式给出的次数
10	小于	<	10	二进制运算符,左边带符号表达式小于右边带符号表达式,则为 1,否则为 0
11	小于等于	<=	10	二进制运算符,左边带符号表达式小于或等于右边带符号表达式,则为 1,否则为 0
12	大于	>	10	二进制运算符,左边带符号表达式大于右边带符号表达式,则为 1,否则为 0
13	大于等于	>=	10	二进制运算符,左边带符号表达式大于或等于右边带符号表达式,则为 1,否则为 0
14	等于	=	9	二进制运算符,左边带符号表达式等于右边带符号表达式,则为 1,否则为 0
15	不等于	!=	9	二进制运算符,左边带符号表达式不等于右边带符号表达式,则为 1,否则为 0
16	逐位与	&	8	二进制运算符,两个表达式之间逐位与
17	逐位异或	^	7	二进制运算符,两个表达式之间逐位异或
18	逐位或		6	二进制运算符,两个表达式之间逐位或
19	逻辑与	&&	5	二进制运算符,两个表达式逻辑与,非 0 则为 1,否则为 0
20	逻辑或		4	二进制运算符,两个表达式逻辑或,非 0 则为 1,否则为 0

4.2 寻址方式

指令的一个重要组成部分是操作数，指令给出参与运算的数据的方式称为寻址方式。AVR 单片机指令操作数的寻址方式有以下多种：单寄存器直接寻址、双寄存器直接寻址、I/O 直接寻址、数据直接寻址、带位移的数据间接寻址、数据间接寻址、带预减量的数据间接寻址、带后增量的数据间接寻址、常量寻址、程序直接寻址、程序间接寻址、程序相关寻址。

一、单一寄存器直接寻址

由指令指出一个寄存器的内容作为操作数，这种寻址方式称为单寄存器直接寻址。寄存器寻址所选的工作寄存器为寄存器文件中的 0-31 区域。指令字的低 5 位（D0~D4 位）指出所用的寄存器 Rd。图 4.1 为单寄存器直接寻址示意图。

二、双寄存器直接寻址

双寄存器直接寻址方式同单寄存器直接寻址方式，指令字中指出两个寄存器 Rd 和 Rr。指令字的 D0~D4 位指出 Rd 寄存器，D5~D9 位指出 Rr 寄存器。结果存在 Rd 寄存器中。图 4.2 为双寄存器直接寻址示意图。

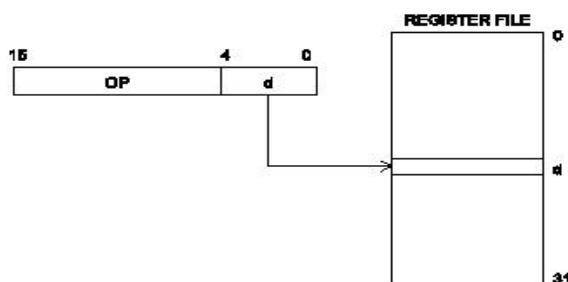


图 4.1 单寄存器直接寻址示意图

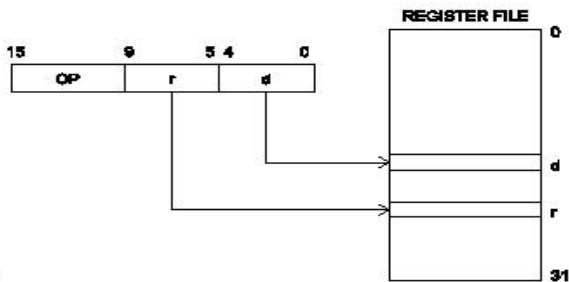


图 4.2 双寄存器直接寻址示意图

三、I/O 直接寻址

在 AVR 单片机的寄存器区中映射有 I/O 寄存器。指令可以直接对 I/O 空间进行操作。操作数包含在指令字中的 D0~D5 位中，同时指令字中还包含了目的或源寄存器地址 n。图 4.3 为 I/O 直接寻址示意图。

四、数据直接寻址

数据直接寻址方式便于直接从 SRAM 存储器中存取数据。数据直接寻址为双字指令，一个 16 位的数据地址放在低字指令中，高字指令中的 Rd/ Rr 指定了目的寄存器或源寄存器。存储器存取的范围限制在 SRAM 当前 64 字节页。图 4.4 为数据直接寻址示意图。

五、带位移的数据间接寻址

带位移的数据间接寻址方式是利用变址寄存器（Y 或 Z）及指令字中的位移量共同决定被存取 SRAM 存储器的地址。操作数的地址由 Y 或 Z 寄存器的内容加上指令字 D0~D5 位给出的位移量 a 给出。指令字中的 n 为目的或源寄存器地址。图 4.5 为带位移的数据间接寻址示意图。

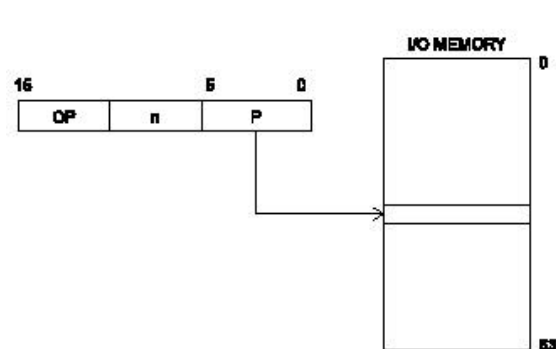


图 4.3 I/O 直接寻址示意图

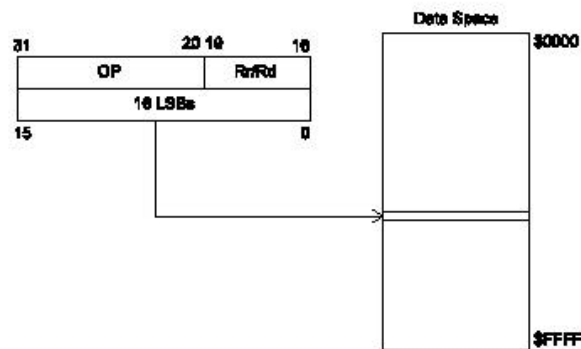


图 4.4 数据直接寻址示意图

六、数据间接寻址

由指令指出某一个寄存器的内容作为操作数的地址，该寻址方式称为寄存器间接寻址。AVR 单片机中用变址寄存器 X、Y 或 Z 作为规定的寄存器，并对 SRAM 存储器存取操作，称为数据间接寻址。操作数的地址在变址寄存器（X、Y 或 Z）中。图 4. 6 为数据间接寻址示意图。

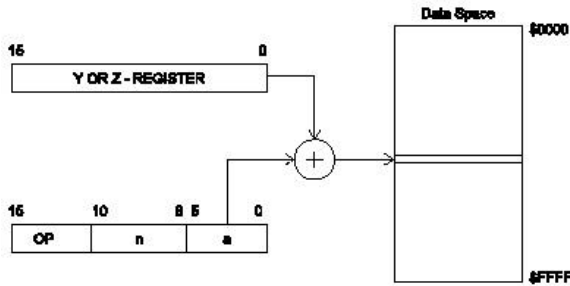


图 4.5 带位移的数据间接寻址示意图

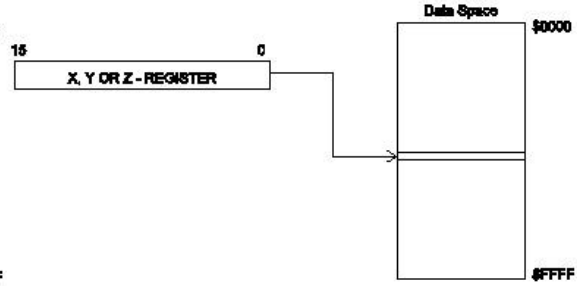


图 4.6 数据间接寻址示意图

七、带预减量数据间接寻址

同数据间接寻址，但寄存器 X、Y 或 Z 的内容在操作之前先被减 1，相减后的内容为操作数的地址。这种寻址方式特别适用于访问矩阵、查表等应用。图 4. 7 为带预减量的数据间接寻址示意图。

八、带后增量的数据间接寻址

同数据间接寻址方式，但寄存器 X、Y 或 Z 的内容在操作后被加 1，而操作数地址的内容为寄存器增量之前的内容。这种寻址方式特别适用于访问矩阵、查表等应用。图 4. 8 为带后增量的数据间接寻址示意图。

九、常量寻址

常量寻址主要从程序存储器取常量。程序存储器中放常量字节的地址由寄存器 Z 的内容确定。Z 寄存器的高 15 位用于选择字地址（0~4K），而 Z 寄存器的最低位（D0）用于写字地址的高低字节。若最低位被清除（LSB=0），则选择低字节；若最低位被置位（LSB=1），则选择高字节，例如 LPM 指令。图 4. 9 为常量寻址示意图。

十、程序直接寻址

程序直接寻址方式中操作数包含在指令字中，即操作数以指令字的形式存放于程序存储器中。程序在双指令字中操作数指定的立即地址处执行，如 JMP、CALL 指令。图 4. 10 为程序直接寻址示意图。

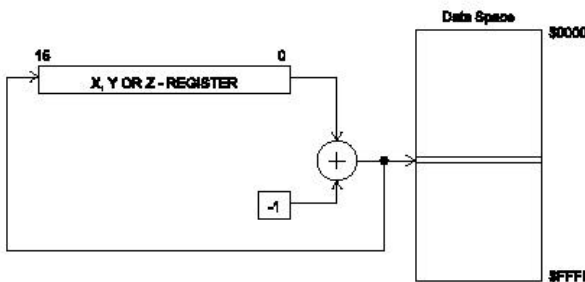


图 4.7 带预减量的数据间接寻址示意图

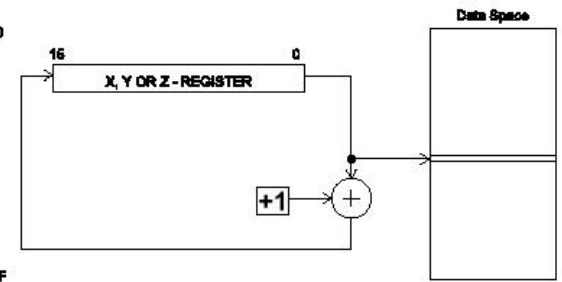


图 4.8 带后增量的数据间接寻址示意图

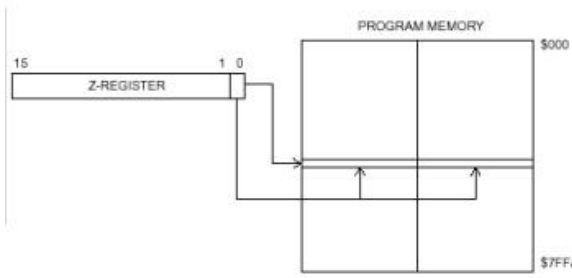


图 4.9 常量寻址示意图

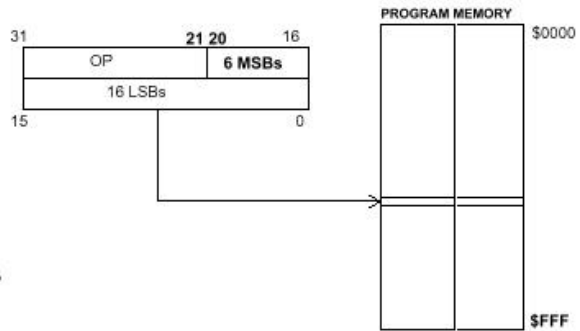


图 4.10 程序直接寻址示意图

十一、程序间接寻址

程序间接寻址方式中，使用 Z 寄存器存放要执行程序地址。程序在 Z 寄存器的内容指定的地址处继续执行，即用寄存器 Z 的内容代替 PC 的值，如 IJMP、ICALL 指令。图 4.11 为程序间接寻址示意图。

十二、程序相关寻址

程序间接寻址方式中，在指定字中包含了相关地址 K。执行程序时，首先将 PC 值与相关地址 K 相加，得出程序需要继续执行的下一条指令的地址。然后程序在地址 PC+K 处继续执行。其范围从 -2K 到 (2K-1) 之中，如 RJMP、RCALL 指令。图 4.12 为程序相关寻址示意图。

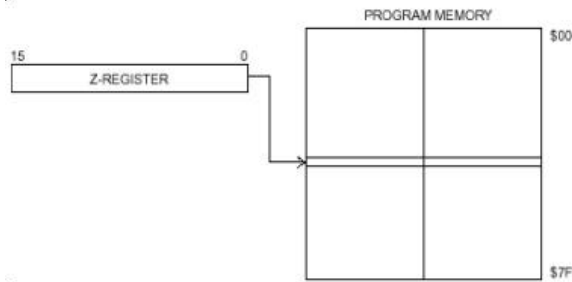


图 4.11 程序间接寻址示意图

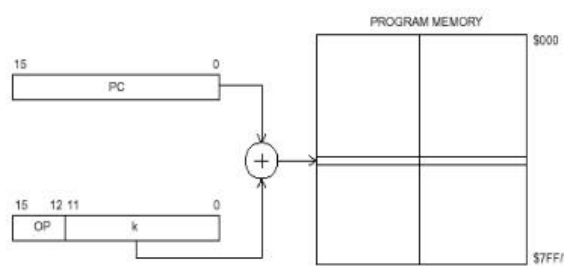


图 4.12 程序相关寻址示意图

4.3 数据操作和指令类型

4.3.1 数据操作

AVR 单片机是一个增强型 RISC 微控制器，具有高性能的数据处理能力，能对位、半字节、字节和双字节数据进行各种操作。它们包括算术和逻辑运算、数据传送、布尔处理和转移操作。

4.3.2 指令类型

AVR 单片机共有 89-130 条指令。如果 118 条指令按功能分类，则有 22 条算术和逻辑指令、34 条条件转移指令、31 条数据传送指令、31 条位指令和位测试指令。

4.3.3 指令集名词

1. 状态寄存器

SREG: 状态寄存器。	S: N V, 符号测试位。
C: 进位标志位。	H: 半进位标志位。
Z: 零标志位。	T: 用于 BLD 和 BST 指令传送位。
N: 负数标志位。	I: 全局中断触发/禁止标志位。
V: 2 的补码溢出指示位。	

2. 寄存器和操作码

Rd: 寄存器区中的目的(或源)寄存器。
Rr: 寄存器区中的源寄存器。
R: 指令执行后的结果。
K: 常数项或字节数据(8位)。
k: 程序计数器的常量地址数据。
b: 在寄存器区中或 I/O 寄存器(3位)中的位。
S: 在状态寄存器(3位)中的位。
X, Y, Z: 间接地址寄存器(X=R27,R26; Y=R29,R28; Z=R31,R30)。
P: I/O 口地址。
q: 直接寻址的偏移(6位)。

3. I/O 寄存器

RAMPX, RAMPY, RAMPZ: X、Y、Z 寄存器的级联寄存器，允许 MCU 在相连多于 64K 字节的 SRAM 整个范围内间接寻址。

4. 堆栈

STACK: 作为返回地址和压栈寄存器的堆栈。
SP: STACK 的堆栈指针。

5. 标志

↔: 由指令引起的有效标志。
0: 由指令清除的标志。
1: 由指令置位的标志。
-: 由指令引起的无效标志。

说明: 为了使读者和用户对每条指令有一个具体的了解,又有利于大家对单片机映象空间(通用工作寄存器空间、I/O 寄存器空间、片内片外 SRAM 空间、程序存储器空间、E2PROM 数据存储器空间)认识更清楚(软件即完成在单片机映象空间之间或自身之间的传送、运算、检测、处理等操作),我们对每条指令均编一些简单的测试指令,我们约定它的程序编号为第四章第四节第几段第几题,例 4.4.1 加法指令的“1.不带进位加法”,程序编号为 4411.ASM。其余依此类推。

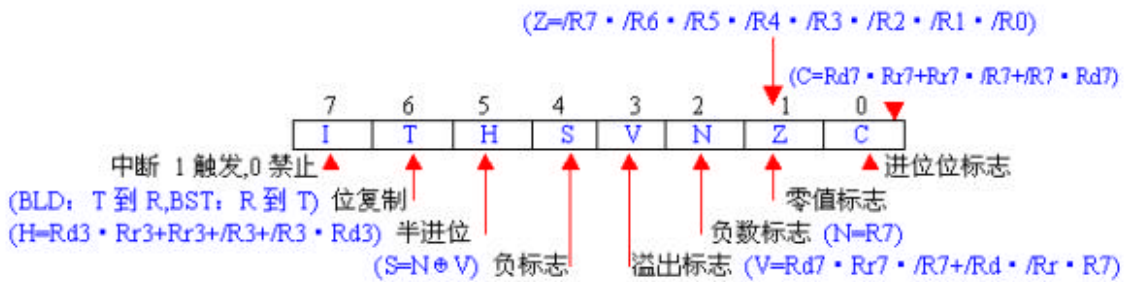
说明:AVR 单片机的指令系统对不同器件有不同指令,选用器件时应注意这一点(详情见本书附录 4),某种器件应使用那些指令,更详细资料请阅相应器件(电子书光盘中)英文指令表。

以下讲述 130 条指令功能及应用

4.4 算术和逻辑指令

AVR 的算术运算指令有加、减、乘、取反、取补、比较指令、增量和减量指令。逻辑运算指令有与、或、异或指令等。

图 4.4.1 是状态寄存器每位的功能及在运算过程中的功能示意图



复位后状态寄存器—SREG=\$00,即禁止中断,无半进位,无负标志,无溢出,无负数,无零标志,无进位,等

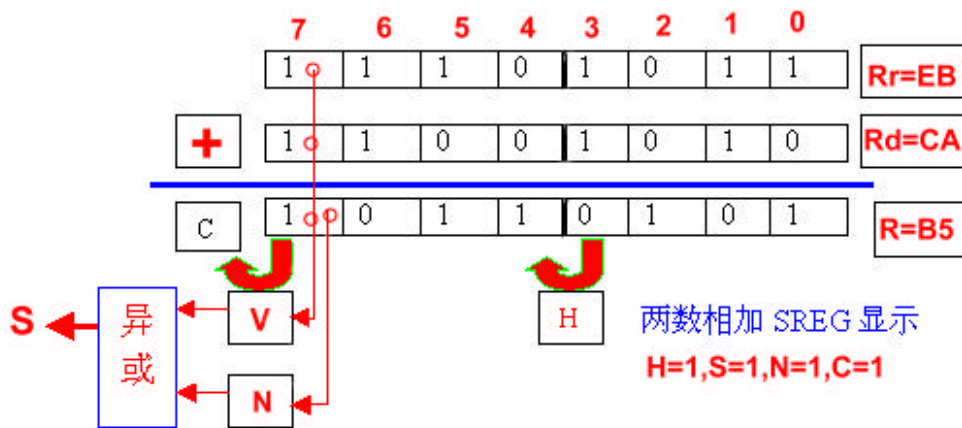


图 4.4.1 状态寄存器功能示意图

4.4.1 加法指令

1. 不带进位加法

ADD 一不带进位加

说明: 两个寄存器不带进位 C 标志加, 结果送目的寄存器 Rd。

操作: $Rd \leftarrow Rd + Rr$

语法:

操作码:

程序计数器:

ADD Rd,Rr $0 \leq d \leq 31, 0 \leq r \leq 31$

$PC \leftarrow PC + 1$

16 位操作码:

0000	11rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $Rd3 \cdot Rr3 + Rr3 \cdot /R3 + /R3 \cdot Rd3$

N: R7

S: N V,

Z: $/R7 \cdot /R6 \cdot /R5 \cdot /R4 \cdot /R3 \cdot /R2 \cdot /R1 \cdot /R0$

V: Rd7· Rr7·/R7+/Rd7·/Rr7·R7

C: Rd7.Rr7+Rr7·/R7+/R7·Rd7

重要说明:以下所有举的例子仅说明指令书写方法,该例子不能直接汇编,因为程序缺器件配制文件及程序执行地址。实践操作例子*.ASM 可以汇编,可以调试,可以修改(修改需改变文件属性,因为只读文件无法修改)!

约定:注释寄存器的内容(数据)用括号表示,例:(R16)=\$16,表示寄存器的内容(数据)为十六进制数 16H!



例子:。(实践操作程序 4411.ASM) 实践操作例子*.ASM,必须编译生成*.OBJ 文件才可调试,如要修改*.ASM,必须修改文件属性,去掉*.ASM 只读文件属性

```
ldi r16,$11 ;LDI 立即数装入指令,要求寄存器必须符合 16≤d≤31 条件
ldi r20,$22
ldi r28,0XAA ;$,0X 均为十六进制表示法
lp:add r16, r20 ;也可单步执行到此行,
;在调试窗口的对应寄存器 R16,R20 输入数据
;(R16) = , (SREG) =
add r28, r28 ;也可单步执行到此行,在调试窗口的对应寄存器 R28 输入数据
;(R28) = , (SREG) =
rjmp lp ;反复做实验
```

Words: 1 (2 bytes)
Cycles: 1

2. 带进位加法

ADC——带进位加

说明: 两个寄存器和 C 标志的内容相加, 结果送目的寄存器 Rd.

操作: Rd←Rd+Rr+C

语法: 操作码: 程序计数器:

ADC Rd,Rr 0≤d≤31, 0≤r≤31 PC←PC+1

16 位操作码:

0001	11rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: Rd3·Rr3+Rr3+/R3·Rd3

N: R7

S: N V,

Z: /Rd7./Rr7./Rr7·/R7./R7·/Rd7

V: Rd7· Rr7·/R7./Rd7·/Rr7·R7

C: Rd7.Rr7+Rr7·/R7+/R7·Rd7

例子: (实践操作程序 4412.ASM) 实践操作例子*.ASM,必须编译生成*.OBJ 文件才可调试,如

要修改*.ASM,必须修改文件属性,去掉*.ASM 只读文件属性

```
ldi r20,$77      ; LDI 立即数装入指令,要求寄存器必须符合  $16 \leq d \leq 31$  条件
ldi r21,$99
LDI R22,0X77
LDI R23,0X11
lp:add  r22, r20 ;(r22) =      ,(SREG) =
ADC  R23, R21 ; 也可单步执行到此行,在调试窗口的对应寄存器 R23,R21 输入数据
      ;(R23) =      ,(SREG) =
rjmp lp      ; 反复做实验

Words:  1 (2 bytes)
Cycles:  1
```

3. 立即数据加法 (字)

ADIW—立即数加法

说明: 寄存器对于立即数值 (0~63) 相加, 结果放到寄存器对。

操作: $Rdh:Rdl \leftarrow Rdh:Rdl + K$

语法: `ADIW Rdl, K` 操作码: 程序计数器:
 $d \in \{ 24 \ 26 \ 28 \ 30 \}$, UJ $PC \leftarrow PC + 1$
 16 位操作码:

1001	0110	KKdd	KKKK
------	------	------	------

状态寄存器 (SREG)和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	↔

S: N V

V: $Rdh7 \cdot R15$

N: R15

Z: $/R15 \cdot /R14 \cdot /R13 \cdot /R12 \cdot /R11 \cdot /R10 \cdot /R9 \cdot /R8 \cdot /R7 \cdot /R6 \cdot /R5 \cdot /R4 \cdot /R3 \cdot /R2 \cdot /R1 \cdot /R0$

C: $/R15 \cdot Rdh7$

例子: (实践操作程序 4413.ASM) 实践操作例子*.ASM,必须编译生成*.OBJ 文件才可调试,如要修改*.ASM,必须修改文件属性,去掉*.ASM 只读文件属性

```
lp:adiw r24, 1      ; 也可单步执行到此行,在调试窗口的对应寄存器 R24 输入数据
      ADIW R30, 63 ; 也可单步执行到此行,在调试窗口的对应寄存器 R30 输入数据
      rjmp lp      ; 反复测试

Words:  1 (2 bytes)
Cycles:  2
```

4. 加 1 指令

INC—加 1

说明: 寄存器 Rd 的内容加 1, 结果送目的寄存器 Rd 中。该操作不改变 SREG 中的 C 标志, 所以 INC 指令允许在多倍字长计算中用作循环计数。当对无符号数操作时, 仅有 BREQ (相等转移) 和 BRNE (不为零转移) 指令有效。当对二进制补码值操作时, 所有的带符号转移指令都有效。

操作: $Rd \leftarrow Rd + 1$

语法: `INC Rd` 操作码: 程序计数器:
 $0 \leq d \leq 31$ $PC \leftarrow PC + 1$

16 位操作码:

1001	010d	dddd	0011
------	------	------	------

状态寄存器 (SREG)和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	↔

S: N V

N: R7

V: R7·/R6·/RS·/R4·/R3·/R2·/R1·/R0

Z: /R7· /R6· /R5· /R4· /R3· /R2· /R1· /R0

例子: (实践操作程序 4414.ASM)

```

clr r22 ;寄存器 R22 清零
loop: inc r22 ;+1, 也可单步执行到此行,在调试窗口的对应寄存器 R22 输入数据
loop1: INC R22
      cpi r22,$04 ;(R22)与立即数$04 比较
      brnq loop1 ;不相等转移,相等则按顺序执行(观察状态寄存器 Z 标志变化)
      rjmp loop ;供反复测试
    
```

Word: 1 (2 bytes)

Cycles: 1

4.4.2 减法指令

1. 不带进位减法

SUB —不带进位减

说明: 两个寄存器相减, 结果送目的寄存器 Rd 中。

操作: $Rd \leftarrow Rd - Rr$

语法:

操作码:

程序计数器:

SUB Rd,Rr

$0 \leq d \leq 31, 0 \leq r \leq 31$

$PC \leftarrow PC + 1$

16 位操作码

0001	11rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: /Rd3·Rr3+Rr3·R3+R3·/Rd3

N: R7

S: N V

Z: /R7·/R6·/R5·/R4·/Rr3·/R2·/R1·/R0

V: Rd7·/Rr7·/R7+ /Rd7· Rr7· R7

C: /Rd7· Rr7+Rr7·R7+R7· /Rd7

例子: (实践操作程序 4421.ASM)

```

loop1: ldi r23,$44 ;寄存器装入立即数
      ldi r22,$11 ;寄存器装入立即数
loop: sub r23,r22 ;减法,也可单步执行到此行,
                ;在调试窗口的对应寄存器 R23,R22 输入数据
      brne loop ;r23 内容不为 0 转,为 0 顺执
      rjmp loop1 ;反复测试
    
```

Words: 1 (2 bytes)

Cycles: 1

2. 立即数减法 (字节)

SUBI—立即数减

说明: 一个寄存器和常数相减, 结果送目的寄存器 Rd。该指令工作于寄存器 R16 到 R31 之间, 非常适合 X、Y 和 Z 指针的操作。

操作: $Rd \leftarrow Rd - K$

语法:

SUBI Rd, K

16 位操作码:

操作码:

$16 \leq d \leq 31, 0 \leq k \leq 255$

程序计数器:

$PC \leftarrow PC + 1$

0101	kkkk	dddd	kkkk
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $/Rd3 \cdot K3 + K3 \cdot R3 + R3 \cdot /Rd3$

S: N V

V: $Rd7 \cdot /K7 \cdot /R7 + /Rd7 \cdot K7 \cdot R7$

例子: (实践操作程序 4422.ASM)

LOOP: LDI R22, \$55

LOOP1: SUBI R22, \$11 ; 也可单步执行到此行, 在调试窗口的对应寄存器 R22 输入数据

BRNE LOOP1 ; 不为 0 转, 为 0 顺执

RJMP LOOP ; 反复测试

Words: 1 (2 bytes)

Cycles: 1

3. 带进位减法

SBC—带进位减

说明: 两个寄存器随着 C 标志相减, 结果放到目的寄存器 Rd 中。

操作: $Rd \leftarrow Rd - Rr - C$

语法:

SBC Rd Rr

16 位操作码

操作码:

$0 \leq d \leq 31, 0 \leq r \leq 31$

程序计数器:

$PC \leftarrow PC + 1$

0000	10rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $/Rd3 \cdot Rr3 + Rr3 \cdot R3 + R3 \cdot /Rd3$

N: R7

S: N V Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0·Z

V: Rd7·/Rr7·/R7+ /Rd7· Rr7· R7 C: /Rd7·Rr7+Rr7·R7+R7·/Rd7

例子: (实践操作程序 4423.ASM)

LP: LDI R22,\$80 ;寄存器装数(R22)=\$80
 LDI R20,\$80 ;(R20)=\$80
 LDI R23,\$23 ;(R23)=\$23
 LDI R21,\$11 ;(R21)=\$11
 ADD R22,R20 ;(R22)=\$00,C=1
 LP1:SBC R23,R21 ;也可单步执行到此行,在调试窗口的对应寄存器 R23,R21 输入数据
 ;(R23)-(R21)-(C)=
 CPI R23,\$00 ;R23 的内容与立即数\$00 比较,
 BRNE LP1 ;R23 的内容不为 0 为转,为 0 顺执
 RJMP LP ;反复测试

Words: 1 (2 byteS)

Cycles: 1

4. 带进位立即数减

SBCI—带进位立即数减

说明: 寄存器和立即数随着 C 标志相减,结果放到目的寄存器 Rd 中。

操作: $Rd \leftarrow Rd - K - C$

语法: 操作码: 程序计数器:

SBCI Rd K $16 \leq d \leq 31, 0 \leq K \leq 255$ $PC \leftarrow PC + 1$

16 位操作码:

0100	KKKK	dddd	KKKK
------	------	------	------

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: /Rd3·K3+K3·R3+R3·/Rd3

N: R7

S: N V

Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0·Z

V: Rd7·/K7·/R7+ /Rd7· K7· R7

C: /Rd7·K7+K7·R7+R7·/Rd7

例子: (实践操作程序 4424.ASM)

LP: SEC ;(C)=1
 LDI R16,\$44 ;(R16)=\$44
 SUBI R16,\$22 ;(R16)减立即数\$22
 LP1:SBCI R16,\$11 ;也可单步执行到此行,在调试窗口的对应寄存器 R16 输入数据
 ;(R16)-\$11-1
 CPI R16,\$00 ;比较 R16 内容是否为\$00
 BRNE LP1 ;(R16)不为 0 为转,为 0 顺执
 RJMP LP ;反复测试

Words: 1 (2 bytes)

Cycles: 1

5. 立即数减法 (字)

SBIW—立即数减法

说明: 双寄存器与立即数 (0~63) 减, 结果送双寄存器。该指令操作于四个以上的寄存器对, 比较适合对指针寄存器操作。

操作: $R_{dh} : R_{dl} \leftarrow R_{dh} : R_{dl} - K$

语法:

操作码:

程序计数器:

SBIW R_{dl}, K $dl \in \{24, 26, 28, 30\}, 0 \leq K \leq 63$ $PC \leftarrow PC + 1$

16 位操作码:

1001	0111	KKdd	KKKK
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	↔	↔	↔	↔

S: NV V: $R_{dh} \cdot /R_{15}$ N: R15 C: $R_{15} \cdot /R_{dh}7$ Z: $/R_{15} \cdot /R_{14} \cdot /R_{13} \cdot /R_{12} \cdot /R_{11} \cdot /R_{10} \cdot /R_9 \cdot /R_8 \cdot /R_7 \cdot /R_6 \cdot /R_5 \cdot /R_4 \cdot /R_3 \cdot /R_2 \cdot /R_1 \cdot /R_0 \cdot Z$

例子: (实践操作程序 4425.ASM)

```

LP: LDI  R24, 5 ; 寄存器装入立即数, 寄存器必须符合  $16 \leq R \leq 31$ 
    LDI  R28, 63 ; 5, 63 为十进制数, 十六进制为 0X3F
    sbiw r24, 1 ;
    sbiw r28, 60 ; 60 为十进制数
    RJMP LP ; 反复测试

```

Words: 1 (2 bytes)

Cycles: 1

6. 减 1 指令

DEC—减 1

说明: 寄存器 R_d 的内容减 1, 结果送目的寄存器 R_d 中。该操作不改变 SREG 中的 C 标志, 所以 DEC 指令允许在多倍字长计算中用作循环计数。当对无符号值操作时, 仅有 BREQ (不相等转移) 和 BRNE (不为零转移) 指令有效。当对二进制补码值操作时, 所有的带符号转移指令都有效。

操作: $R_d \leftarrow R_d - 1$

语法:

操作码:

程序计数器:

DEC R_d $0 \leq d \leq 31$ $PC \leftarrow PC + 1$

16 位操作码:

1001	010d	dddd	1010
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	↔	↔	↔	↔

S: N V

N: R7

V: $/R_7 \cdot /R_6 \cdot /R_5 \cdot /R_4 \cdot /R_3 \cdot /R_2 \cdot /R_1 \cdot /R_0$ Z: $/R_7 \cdot /R_6 \cdot /R_5 \cdot /R_4 \cdot /R_3 \cdot /R_2 \cdot /R_1 \cdot /R_0$

例子: (实践操作程序 4426.ASM)

```
LP: LDI R16,$04 ;多级-1,可作延时子程序用
LOOP: DEC R16 ;也可单步执行到此行,在调试窗口的对应寄存器R16输入数据
      ;(R16)-1
      BRNE LOOP ;(R16)不为0转,为0顺执
      DEC R16 ;第一次((R16)=$00)-1=$FF
      BRNE LP ;(R16)不为0转,为0顺执
      RET ;子程序返回

Words: 1 ( 2 bytes)
Cycles: 1
```

4.4.3 乘法指令

注意: AVR 单片机只有 ATmega161 和 Atmeg103L 有乘法指令

MUL —乘法

说明: 该指令完成 8 位 X 8 位 → 16 位的无符号数乘法操作。被乘数 Rr 和乘数 Rd 是两个寄存器。16 位结果放在 R1 (高字节) 和 R0 (低字节) 中。注意, 如果被乘数和乘数选择了 R0 或 R1, 则当进行乘法后, 结果将溢出。

操作: R1, R0 ← Rr * Rd

语: MUL Rd, Rr 操作码: 程序计数器:
 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1
 16 位操作码:

0001	11rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-						↔

C: R15

例子: (实践操作程序 4431.ASM, 因无相应器件配制文件 *.inc, 所以汇编时有出错提示)

```
lp: ldi r6,$04
    ldi r5,#05
    mul r6,r5 ;乘法
    mov r5,r1 ;保存乘积高位
    mov r5,r0 ;保存乘积低位
    rjmp lp
```

Words: 1 (2 bytes)
 Cycles: 2

4.4.4 取反码指令

COM — 取二进制反码

说明: 该指令完成寄存器 Rd 的二进制反码操作。

操作: Rd ← \$FF - Rd

语法: COM Rd 操作码: 程序计数器:
 0 ≤ d ≤ 31 PC ← PC + 1

16 位操作码:

1001	010d	dddd	0000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	0	↔	↔	1

S: N V Z: /R7./R6./R5./R4./R3./R2./R1./R0
V: 0 C: 1
N: R7

例子: (实践操作程序 4441.ASM)

```

lp: ldi R24,$AA
    com r24 ;取反
    cpi r24,$aa ;(r24)与$aa 比较相等吗?
    breq lp1
lp1: COM R24 ;取反
    cpi r24,$aa ;(r24)与$aa 比较相等吗?
    breq lp ;相等,反复测试
    BREQ LP
  
```

Words: 1 (2 bytes)

Cycles: 1

4.4.5 取补指令

NEG— 二进制补码

说明: 寄存器 Rd 的内容转换成二进制补码, 值\$80 是不改变的。

操作: $R \leftarrow \$00 - Rd$

语法:

NEG Rd

操作码:

$0 \leq d \leq 31$

程序计数器:

$PC \leftarrow PC + 1$

16 位操作码:

1001	010d	dddd	0001
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: R3./Rd3

S: N V

V: R7./R6./R5./R4./R3./R2./R1./R0

例子: (实践操作程序 4451.ASM)

```

LP: SUB R11,R0 ;开始在调试窗口中的对应寄存器输入数据
  
```

;设(r11)=0b1010101=\$AA (r0)=0b10011001=\$99

```

BRPL LP1 ;(r11)为正数转移
LP1: NEG R11
BRMI LP ;(r11)为负数转移
    
```

Words: 1 (2 bytes)
Cycles: 1

4.4.6 比较指令

1. 寄存器比较

CP—比较

说明：该指令完成两个寄存器 Rd 和 Rr 相比较操作，而寄存器的内容不改变。该指令后能使用所有条件转移指令。

操作：Rd—Rr

语法：

CP Rd Rr

16 位操作码：

操作码：

$0 \leq d \leq 31, 0 \leq r \leq 31$

程序计数器：

$PC \leftarrow PC + 1$

1001	01rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

-	-	↔	↔	↔	↔	↔	↔
---	---	---	---	---	---	---	---

H: /Rd3.Rr3+Rr3.R3+R3./Rd3

N: R7

S: N V

Z: Rd7.Rr7.Rr7.R7+ R7.Rd7

V: Rd7./Rd7./R7+/Rd7.Rr7.R7

C: /Rd7.Rr7+Rr7.R7+R7./Rd7

例子：(实践操作程序 4461.ASM)

```

lp:cp r24,r19 ;单步执行到此行,开始时在调试窗口的对应寄存器输入数据
;第一次操作设:(r24)=$AA (r19)=$55
;第二次操作设:(r14)=$11 (r19)=$55
brsh lp1 ;(r24)≥(r19)则转 lp1, (r24)小于(r19)顺执
rjmp lp2 ;
lp1:sub r24,r19 ;(r24)相减(r19)
brne lp ;(r24)不为 0 转,为 0 顺执
lp2:adiw r24,$11 ;立即数加,要求 d∈{ 24 26 28 30 }, 0≤K≤63
rjmp lp ;反复测试
    
```

Words: 1 (2 bytes)

CyCICS: 2

2. 带进位比较

CPC 一带进位比较

说明：该指令完成寄存器 Rd 的值和寄存器 Rr 加前位进位的值相比较操作。而寄存器的内容不改变。该指令后能使用所有条件转移指令。

操作：Rd—Rr—C

语法：

CPC Rd Rr

操作码：

$0 \leq d \leq 31, 0 \leq r \leq 31$

程序计数器：

$PC \leftarrow PC + 1$

16 位操作码:

0000	01rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $/Rd3 \cdot Rr3 + Rr3 \cdot R3 + R3 \cdot /Rd3$ N: R7
 S: N V Z: $/R7 \cdot /R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0 \cdot Z$
 V: $Rd7 \cdot /Rr7 \cdot /R7 + /Rd7 \cdot Rr7 \cdot R7$ C: $/Rd7 \cdot Rr7 + Rr7 \cdot R7 + R7 \cdot /Rd7$

例子: (实践操作程序 4462.ASM)

```

cp r2,r0      ; 单步执行到此行,开始在调试窗口的对应寄存器输入数据
               ; 设:(R2)=$AA,(R0)=$55
lp:sec        ;(c)=1
cpc r3,r1     ;第一次操作设:(r3)=$11,(r1)=$10,
               ;第二次操作设:设:(R3)=$12,(R1)=$10,
brne lp1     ;比较不相等转移,相等顺执
rjmp lp2     ;相对转移
lp1:inc r1    ;+1
rjmp lp
lp2:dec r1    ; -1
rjmp lp      ;反复测试

words:       1 ( 2 bytes)
Cycles:      1

```

3. 立即数比较

CPI——带立即数比较

说明: 该指令完成寄存器 Rd 和常数的比较操作。寄存器的内容不改变。该指令后能使用所有条件转移指令。

操作: Rd—K

语法:

操作码:

程序计数器:

CPI Rd, K $16 \leq d \leq 31, 0 \leq K \leq 255$ $PC \leftarrow PC + 1$

16 位操作码:

0011	KKKK	dddd	KKKK
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $/Rd3 \cdot K3 + K3 \cdot R3 + R3 \cdot /Rd3$ N: R7
 S: N V Z: $/R7 \cdot /R6 \cdot /R5 \cdot /R4 \cdot /R3 \cdot /R2 \cdot /R1 \cdot /R0$
 V: $Rd7 \cdot /K7 \cdot /R7 + /Rd7 \cdot K7 \cdot R7$ C: $/Rd7 \cdot K7 + K7 \cdot R7 + R7 \cdot /Rd7$

例子: (实践操作程序 4463.ASM)

```

LP: CPI R19,3 ; 单步执行到此行,开始在调试窗口的对应寄存器输入数据
; 设(R19)=4
BRNE LP1 ; 不相等转,相等顺执
INC R19 ; (R19)+1
RJMP LP ; 反复测试
LP1: DEC R19 ; (R19)-1
RJMP LP ; 反复测试
    
```

Words: 1 (2 bytes)
Cycles: 1

4.4 .7 逻辑与指令

逻辑与输入		输出
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

1. 寄存器逻辑与

AND 一逻辑与 ; 全 1 为 1, 有 0 即 0,
说明: 寄存器 Rd 和寄存器 Rr 的内容为逻辑与, 结果送目的寄存器 Rd。
应用: 清 0, 使某位为 0, 用 0 去与; 保留, 用 1 去逻辑与; 代硬件与门
操作: $Rd \leftarrow Rd \ \& \ Rr$;

语法: AND Rd,Rr 操作码: $0 \leq d \leq 31 \quad 0 \leq r \leq 31$ 程序计数器: $PC \leftarrow PC + 1$
16 位操作码:

0010	00rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	∇	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	

S: N V N: R7
V: 0 Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4471.ASM)

```

LP: add r2,r3 ; 单步执行到此行,开始在调试窗口的对应寄存器输入数据
; 设: (R2)=0B1000 0000, (R3)=0B0001 0011
LDI R16,1 ; (R16)=0B0000 0001,
and r2,r16 ; (R2)=
RJMP LP ; 反复测试
    
```

Words: 1 (2 bytes)
Cycles: 1

V: 0 Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4473.ASM)

```

lp:cbr R16,$F0   ; 单步执行到此行,在调试窗口的对应寄存器输入数据
                 ;(r16)=$FF
CBR R18,1       ;(r18)=$80
INC R16
INC R18
rjmp lp         ;反复输入数据测试

```

Words: 1 (2 bytes)

Cycles: 1

4. 测试零或负

TST 测试零或负

说明: 测试寄存器是否是零或是负。完成同一寄存器之间的逻辑与操作, 而寄存器内容不改变。

操作: Rd←Rd Rd

语法: 操作码: 程序计数器

TST Rd 0≤d≤31 PC←PC+1

16 位操作码:

0011	00dd	dddd	dddd
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	0	↔	↔	

S: N V

N: R7

V: 0

Z: /R7·/R6·/RS·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4474.ASM)

```

sub r0,r2       ; 单步执行到此行,在调试窗口的对应寄存器输入数据
                 ;并打开状态寄存器 SREG 观察窗口
                 ;(r0)=$aa,(r2)=$aa
                 ;(r0)=$77,(r2)=$80
lp: tst   r0    ; 测试寄存器 r0 是否是零或是负,
   breq  lp1   ; 如果零标志(Z)位为 1,则转移,为 0 顺执
   dec  r0    ; -1
   rjmp lp    ; 再测试
lp1: inc  r0    ; +1
   rjmp lp    ; 再测试

```

Words: 1 (2 bytes)

Cycles: 1

4.4.8 逻辑或指令

逻辑或输入		输出
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

1. 寄存器逻辑或

OR 一逻辑或 ;有 1 即 1,全 0 为 0,

应用: 置数,使某位为 1,用 1 去或;保留,用 0 去逻辑或;代硬件或门

说明: 完成寄存器 Rd 与寄存器 Rr 的内容逻辑或操作,结果进目的寄存器 Rd 中。

操作: $Rd \leftarrow Rd \vee Rr$

语法:

OR Rd Rr

16 位操作码:

操作码:

$0 \leq d \leq 31, 0 \leq r \leq 31$

程序计数器:

$PC \leftarrow PC + 1$

0010	10rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	0	↔	↔	

S: N V

V: 0

例子: (实践操作程序 4481.ASM)

or r19,r16 ;逻辑或,单步执行到此行,在调试窗口的对应寄存器输入数据

; (R19)=\$AA , (R16)=\$44 , (R18)=0B0100 1111=\$4F

; (R18)=0B0000 1111=\$0F

LP:bst r18,6 ;R18 中的 6 位内容到 SREG 中 T 标志,观察 SREG 中 T 标志的变化

brts ok ;SREG 中标志为 1 转移,为 0 顺执

SER R18 ;置位 R18

RJMP LP ;反复测试

ok:CLR R18 ;清零 R18

RJMP LP ;反复测试

Words: 1 (2 bytes)

Cycles: 1

2. 带立即数或

ORI—立即数逻辑或 ;功能: 保留(屏蔽)数据,置数(使某位为 1)

说明: 完成寄存器 Rd 的内容与常量逻辑或操作,结果送目的寄存器 Rd 中。

操作: $Rd \leftarrow Rd \vee K$

语法:

ORI Rd K

16 位操作码:

操作码:

$16 \leq d \leq 31, 0 \leq K \leq 255$

程序计数器:

$PC \leftarrow PC + 1$

0110	KKKK	dddd	KKKK
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	0	↔	↔	

S: N V

N: R7

V: 0

Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4482.ASM)

```

LP: ori r19, $F0 ;立即数或, 单步执行到此行, 在调试窗口的对应寄存器输入数据
                ;(R19)=$A0 , (R17)=$45
                ;(R19)=0B0110 1111=$6F , (R17)=0B1111 0000=$F0
    ori r17, 1   ;立即数或
    lsl r19      ;r19 逻辑左移
    lsr r17      ;r17 逻辑右移
    RJMP LP     ;反复测试
  
```

Words: 1 (2 bytes)

Cycles: 1

3. 置寄存器位

SBR—寄存器位置位

说明: 对寄存器 Rd 中指定位置位。完成寄存器 Rd 和常数表征码 K 之间的逻辑直接数或 (ORI), 结果送目的寄存器 Rd。

操作: $Rd \leftarrow Rd \vee K$

p95

语法:

操作码:

程序计数器:

SBR Rd K $16 \leq d \leq 31, 0 \leq K \leq 255$

$PC \leftarrow PC + 1$

16 位操作码:

0110	KKKK	dddd	KKKK
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	0	↔	↔	

S: N V

N: R7

V: 0

Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4483.ASM)

```

                ;设: (R16)=$F0 , (R17)=$80
lp: sbr r16, 3 ;对 R16 的(0B0000 0011) 第 1、 0 位置为 1, 执行后 (R16 )=
    sbr r17, $17 ;对 R17 的(0B0001 0111) 第 4、 2、 1、 0 位置 1, 执行后 (R17)=
    INC R16     ;+1
  
```

DEC R17 ; -1
rjmp lp ; 反复试验

Words: 1 (2 bytes)

Cycles: 1

4. 置寄存器

SER—置位寄存器的所有位

说明: 直接装入\$FF 到寄存器 Rd.

操作: $Rd \leftarrow \$FF$

语法:

操作码:

程序计数器:

SER Rd $16 \leq d \leq 31$

$PC \leftarrow PC + 1$

16 位操作码:

1110	1111	dddd	1111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4484.ASM)

```
LP:CLR R16 ;(R16)清零
ser r17 ;(R17)置$FF
out $18,r16 ;输出到 B 口,打开 I/O 寄存器窗口看$18 变化
out $18,r17 ;输出到 B 口,打开 I/O 寄存器窗口看$18 变化
INC R16 ;+1
INC R17 ;+1
RJMP LP ;反复试验
Words: 1 ( 2 bytes)
Cycles: 1
```

4.4.9 逻辑异或指令

1. 寄存器异或

EOR—异或 ;输入相同输出为 0,输入不同输出为 1;也称同或(清零);也称互斥(置 1);见下真值表

说明: 完成寄存器 Rd 和寄存器 Rr 的内容相逻辑异或操作,结果送目的寄存器 Rd.

异或输入		输出
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

操作: $Rd \leftarrow Rd \oplus Rr$

语法:

操作码:

程序计数器:

EOR Rd,Rr $0 \leq d \leq 31, 0 \leq r \leq 31$

$PC \leftarrow PC + 1$

16 位操作码:

0010	01rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
			↔	0	↔	↔	

S: N V

N: R7

V: 0

Z: /R7·/R6·/R5·/R4·/R3·/R2·/R1·/R0

例子: (实践操作程序 4491.ASM)

LP:eor r4,r4 ;设:(R4)=0B1010 0011,相同数异或为清零,也可称同或清零

eor r0,r22 ;设:(R0)=0B1010 0101 设:(R22)=0B0101 0011

SWAP R4 ;半字节交换

SWAP R0 ;半字节交换

SWAP R22 ;半字节交换

RJMP LP ;反复实验

Words: 1 (2 bytes)

Cycles: 1

2. 清除寄存器

CLR——寄存器清零

说明: 寄存器清零。该指令采用寄存器 Rd 与自己的内容相异或实现的。寄存器的所有位都被清零。

操作: $Rd \leftarrow Rd \oplus Rd$

语法: 操作码:

程序计数器:

CLR Rd $0 \leq d \leq 31$ $PC \leftarrow PC + 1$

16 位操作码:

0001	11rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
			0	0	0	1	

S: 0

N: 0

V: 0

Z: 1

例子: (实践操作程序 4492.ASM)

LP:CLR R18 ;R18 清零

Lp1:inc r18 ;+1

CPI R18,\$05 ;R18 内容与立即数比较

brne lp1 ;不相等转,相等顺执

RJMP LP ;循环测试

Words: 1 (2 bytes)

Cycles: 1

4. 5 转移指令

4.5.1 无条件转移指令

1. 相对跳转

RJMP —相对跳转

说明：相对跳转到 PC -2K 和 PC+2K（字）范围内的地址。在汇编程序中，标号用于替代相对操作。AVR 微控制器的程序存储器空间不超过 4K 字（8K 字节），该指令能寻址整个存储器空间的每个地址位置。

严格地讲：在 PC+1 后的 -2K ≤ k ≤ 2K 范围内转移才能正确执行！

操作： PC ← (PC+1) + k

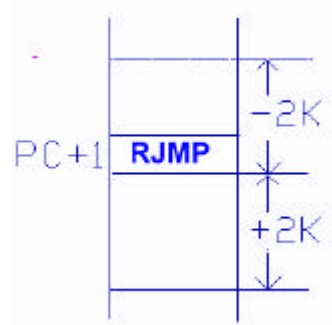
语法： RJMP k 操作码： 程序计数器：
 RJMP k - 2K ≤ k ≤ 2K PC ← (PC+1) + k

16 位操作码：

1100	kkkk	kkkk	kkkk
------	------	------	------

状态寄存器（SREG）和布尔格式：

I	T	H	S	V	N	Z	C



例子：（实践操作程序 4511.ASM）

```
.ORG 0X0010
LP: cpi r16, $42      ; 设: (R16)=$42
   brne LP2         ; 不相等转移, 相等顺执
   rjmp LP1         ; 转移不超过+2K
LP2: ADD R16, R17
   inc r16
.ORG $080F          ; ORG 为$0810, 则超过-2K, ORG 为$080F 不超过-2K
                    ; 因为 PC+1 ($080F+1=$0810), 从$0010 到$0810 正好等于-2K
                    ; 如 PC+1 ($0810+1=$0811), 从$0010 到$0811 正好超过-2K,
                    ; 汇编时出错提示,
LP1: RJMP LP        ; 请修改 ORG 为$0810 一试, 打开程序存储器窗口观察
                    ; 用 RJMP 一相对转移好处: 作为子程序模块搬家(移动)较方便, 不必修改转移指令, 缺点: 子程序
                    ; 大小限在 PC ← (PC+1) + k
```

Words: 1 (2 bytes)
 Cycles: 2

2. 间接跳转

IJMP —间接跳转

说明：间接跳转到由寄存器区中的 Z（16 位）指针寄存器指向的地址。Z 指针寄存器是 16 位宽，允许在当前程序存储器空间 64K 字（128K 字节）内跳转。

IJMP—间接跳转优点: 转移范围大, 缺点: 作为子程序模块, 移值时需修改转移地址, 希望在子程序中不要使用! 不要给自己带来麻烦!

注意: 只能到你设计的硬件电路所具有的空间, 你的器件可有这条指令吗?

操作: $PC \leftarrow Z (15-0)$

$PC (15-0) \leftarrow Z (15-0)$

语法: IJMP 操作码: None 程序计数器: See Operation
16 位操作码:

1001	0100	XXXX	1001
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4512.ASM)

```

ORG $0010
LP:MOV R30,R0 ;设(R0)为 Z 寄存器低 8 位地址,R30 为 Z 间接地址低 8 位
MOV R31,R1 ;设(R1)为 Z 寄存器高 8 位地址,R31 为 Z 间接地址高 8 位
ijmp ;根据 Z 寄存器的内容跳转
.ORG $0200
LDI R20,$11 ;地址$0201 ,如设(R0)=$01,(R1)=02,执行 IJMP 到此行
LDI R21,$22 ;地址$0202 ,如设(R0)=$02,(R1)=02,执行 IJMP 到此行
LDI R22,$33 ;地址$0203 ,如设(R0)=$03,(R1)=02,执行 IJMP 到此行
LDI R23,$44 ;地址$0204 ,如设(R0)=$04,(R1)=02,执行 IJMP 到此行
LDI R24,$55 ;地址$0205 ,如设(R0)=$05,(R1)=02,执行 IJMP 到此行
LDI R25,$66 ;地址$0206 ,如设(R0)=$06,(R1)=02,执行 IJMP 到此行
IJMP ;地址$0207 ,如设(R0)=$07,(R1)=02,执行 IJMP 到此行
;又根据 Z 寄存器的内容跳转到指定地址
    
```

Words: 1 (2 bytes)
Cycles: 1

3. 长跳转

JMP—跳转

说明: 在整个程序存储空间 4M (字) 内跳转, 见 RJMP。

JMP—跳转优点: 可跳转到程序存储器空间 4M (字) 任何地方;

缺点: 不适宜子程序模块中使用;

注意: 只能到你设计的硬件电路所具有的空间, 你的器件可有这条指令吗!

操作: $PC \leftarrow k$

语法: JMP k 操作码: 程序计数器: $PC \leftarrow k$
 $0 \leq k \leq 4M$

32 位操作码:

1001	010k	kkkk	110k
------	------	------	------

kkkk	kkkk	kkkk	kkkk
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4513.ASM)

```
.equ lp1=$0f00
lp:mov r0,r1 ;设(r1)=12
    jmp lp2 ;长跳转
;AT90S8515 等 AVR 单片机无此指令,
;现在用 AVR 的 MEG103 单片机可使用该条指令

.org $0f00
lp2:swap r0 ;半字节交换
    jmp lp ;跳回反复实验
```

Words: 2 (4 bytes)
Cycles: 3

4.5.2 条件转移指令

条件转移指令是依某种特定的条件转移的指令。条件满足则转移，条件不满足时则顺序执行下面的指令。

一、测试条件符合转移指令

1. 状态寄存器中位置位转移

BRBS—SREG 中的位被置位转移

说明: 条件相对转移, 测试 SREG 的某一位, 如果该位被置位, 则相对 PC 值转移。这条指令相对 PC 转移的方向为: PC -64≤目的寄存器≤PC+63。参数 K 为 PC 的偏移, 用 2 的补码表示。

操作: If SREG(S)=1 then PC←(PC+1)+k,else PC←PC+1
语法: 操作码: 程序计数器:
BRBS S, k 0≤S≤7 -64≤k≤+63 PC←(PC+1)+k
PC←PC+1

16 位操作码:

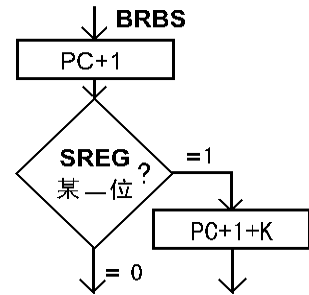
1111	00kk	kkkk	ksss
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4521.ASM)

```
LP:bst r0,3 ;设:(R0)=0B0000 1000;R0 中第 3 位到 SREG 中的 T 标志
;设:(R0)=0B0100 0100;R0 中第 3 位到 SREG 中的 T 标志
brbs 6,LP1 ; SREG 中的 6 位被置位(T=1)则转移,若为零顺执
SET ;置 T 标志为 1
```




```
RJMP LP      ;反复测试
LP1:CLT     ;T 标志清零
RJMP LP      ;反复测试
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true

2. 状态寄存器中位清零转移

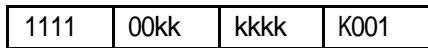
BRBC——SREG 中的位被清零转移

说明：条件相对转移，测试 SREG 的某一位，如果该位被清零，则相对 PC 值转移。这条指令相对 PC 转移的方向为： $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移，用 2 的补码表示。

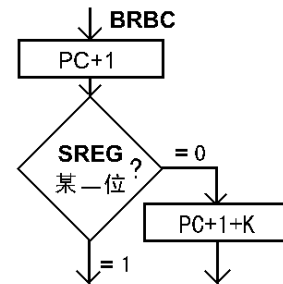
操作：If SREG(S)=0 then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$

语法：操作码：程序计数器：
 BRBC S, k $0 \leq S \leq 7, -64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码：



状态寄存器 (SREG) 和布尔格式：



例子：(实践操作程序 4522.ASM)

```
CPI R20,5    ;R20 中内容与立即数 05 比,设 (R20)=6 或 (R20)=5
LP:BRBC 1,LP1 ;SREG 中位被清零则转移,为 1 顺执
CLZ
RJMP LP      ;反复测试
LP1:SEZ      ;Z=1
RJMP LP      ;反复测试
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true

3. 相等转移

BREQ 相等转移

说明：条件相对转移，测试零标志 (Z)，如果 Z 位被置位，则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令，且当寄存器 Rd 中无符号或有符号二进制数与寄存器 Rr 中无符号或有符号 H 进制数相等时，转移将发生。该指令相对 PC 转移的方向为： $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移，用 2 的补码表示 (相当于指令 BRBSI)，

K)。

操作: If Rd=Rr(z=1)then PC←(PC+1)+k, PC←PC+1

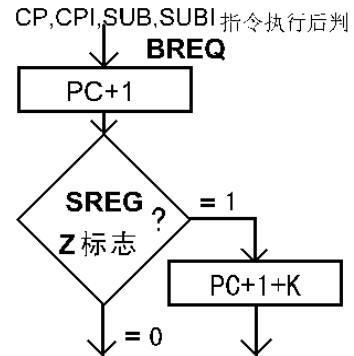
语法: 操作码: 程序计数器:
BREQ k -64≤k≤+63 PC←(PC+1)+k
 PC←PC+1

16 位操作码:

1111	00kk	kkkk	K001
------	------	------	------

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C



例子: (实践操作程序 4523.ASM)

```

lp:cp r1,r2 ;设:(R1)=$AA.(R2)=$AA
breq lp1 ;相等转移,不相等顺执,请同时观察 Z 标志
inc r1 ;+1
rjmp lp ;反复验证
lp1:dec r1 ;-1
rjmp lp ;反复验证
  
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
 2 if condition is true

4. 不相等转移

BRNE—不相等转移

说明: 条件相对转移, 测试零标志 (Z), 如果 Z 位被清零, 则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令, 且当在寄存器 Rd 中的无符号或带符号二进制数不等于寄存器 Rr 中的无符号或带符号二进制数时, 转移将发生。该指令相对 PC 转移的方向为: PC-64 ≤目的≤PC+63。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBCIK)。

操作: If Rd≠Rr(Z=0) then PC←(PC+1)+k, elsePC←PC+1

语法: 操作码: 程序计数器:
BRNE k -64≤k≤+63 PC←(PC+1)+k
 PC←PC+1

16 位操作码:

1111	01kk	kkkk	k001
------	------	------	------

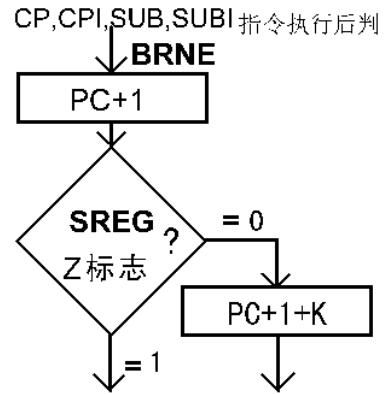
状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4524.ASM)

```
lp:cp r1,r2 ;设:(R1)=$AB.(R2)=$AA
brne lp1 ;不相等转移,相等顺执,请同时观察 Z 标志
inc r1 ;+1
rjmp lp ;反复验证
lp1:dec r1 ;-1
rjmp lp ;反复验证
```

Words: 1 (2bytes)
Cycles: 1 if condition is false
2 if condition is true



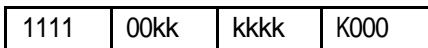
5. C 标志位置位转移

BRCS-进位位置位转移

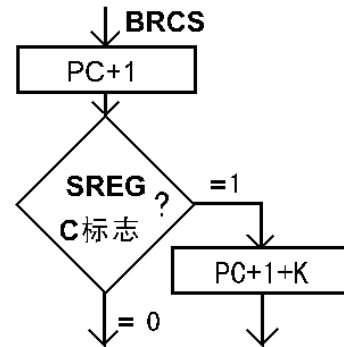
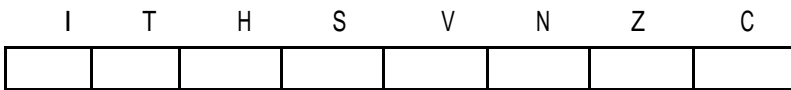
说明: 条件相对转移, 测试进位标志 (C), 如果 C 位被置位, 则相对 PC 值转移。这条指令相对 PC 转移的方向为: $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBS 0, K)。

操作: If C=1 then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$
语法: 操作码: 程序计数器:
BRCS k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码:



状态寄存器 (SREG) 和布尔格式:



例子: (实践操作程序 4525.ASM)

```
SEC ;C=1,请同时观察 C 标志

LP:BRCS LP1 ;C=1 转,C=0 顺执
SEC ;C=1
RJMP LP ;重复试验
LP1:CLC ;C=0
RJMP LP ;重复试验
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true

6. C 标志位清除转移

BRCC——进位位清除转移

说明: 条件相对转移, 测试进位标志 (C), 如果 C 位被清除, 则相对 PC 值转移。这条指令相对 PC 转移的方向为: $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示

(相当于指令 BRBC 0, K)。

操作: If C = 0 then PC ← (PC+1)+k, else PC ← PC+1

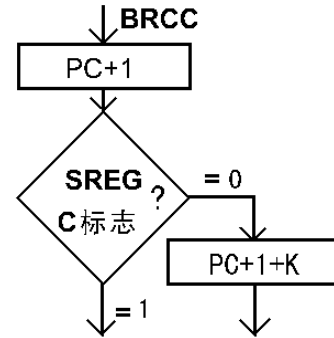
语法: 操作码: 程序计数器:
BRCC k -64 ≤ k ≤ +63 PC ← (PC+1)+k
 PC ← PC+1

16 位操作码:

1111	01kk	kkkk	K000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C



例子: (实践操作程序 4526.ASM)

```
CLC           ;C=0, 请同时观察 C 标志
LP:BRCC LP1   ;C=0 转,C=1 顺执
CLC           ;C=0
Rjmp LP       ;重复试验
LP1:SEC       ;C=0
Rjmp LP       ;重复试验
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
 2 if condition is true

7. 大于或等于转移

BRSH-大于等于转移 (无符号)

说明: 条件相对转移, 测试进位标志 (C), 如果 C 位被清零, 则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令, 且当在寄存器 Rd 中无符号二进制数大于等于寄存器 Rr 中无符号二进制数时, 转移将发生。该指令相对 PC 转移的方向为: PC - 64 ≤ 目的 ≤ PC+63。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBS 0, K)。

操作: If Rd ≥ Rr (C=0) then PC ← (PC+1)+k, else PC ← PC+1

语法: 操作码: 程序计数器:
BRSH k -64 ≤ k ≤ +63 PC ← (PC+1)+ k
 PC ← PC+1

16 位操作码:

1111	01kk	kkkk	K000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

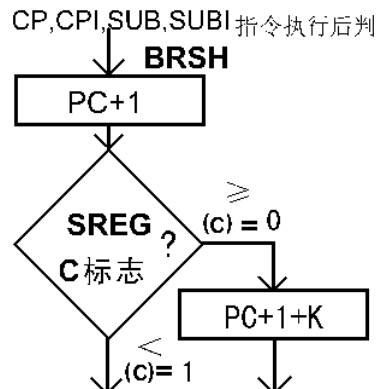
例子: (实践操作程序 4527.ASM)

```
lp:subi r19,2 ;带进位位立即数减,设:(r19)=1
```

```

brsh lp1      ;大于等于转,小于顺执,请同时观察 C 标志
inc r19      ;(r19)+1
inc r19
inc r19
rjmp lp      ;反复实验
lp1:inc r19  ;(r19)+1
rjmp lp      ;反复实验
    
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true
 3



8. 低于转移

BRLO — 小于转移 (无符号)

说明: 条件相对转移, 测试进位标志 (C), 如果 C 位被置位, 则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令, 且当在寄存器 Rd 中无符号二进制数小于在寄存器 Rr 中无符号二进制数时, 转移将发生。该指令相对 PC 转移的方向为: $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBS 0, K)。

操作: If $Rd < Rr (C=1)$ then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$

语法: 操作码: 程序计数器:
 BRLO k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码:

1111	00kk	kkkk	k000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

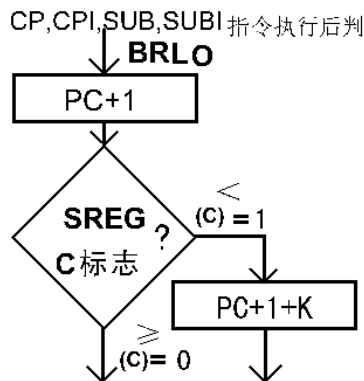
例子: (实践操作程序 4528.ASM)

```

lp:subi r19,2 ;带进位位立即数减,设:(r19)=2
brlo lp      ;小于转,大于等于顺执,
            ;请同时观察 C 标志

inc r19     ;(r19)-1
rjmp lp     ;反复实验
lp1:inc r19 ;(r19)+1
inc r19
inc r19
inc r19
rjmp lp     ;反复实验
    
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true



9. 负数转移

BRMI — 负数转移

说明：条件相对转移，测试负号标志 (N)，如果 N 被置位，则相对 PC 值转移。该指令相对 PC 转移的方向为： $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBS 2, K）。

操作：If N=1 then $PC \leftarrow (PC+1) + k$, else $PC \leftarrow PC+1$

语法：操作码：程序计数器：
BRMI k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1) + k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	00kk	kkkk	K010
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

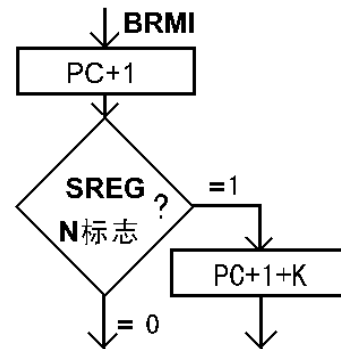
I	T	H	S	V	N	Z	C

例子：(实践操作程序 4529.ASM)

```

lp:subi r18,2 ; 带进位立即数减,设:(r18)=2
brmi lp1 ; 为负转 N=1,为正顺执 N=0
; 请同时观察 N 标志

inc r18 ; (r18)+1
rjmp lp ; 反复实验
lp1:inc r18 ; (r18)+1
inc r18
inc r18
inc r18
rjmp lp ; 反复实验
Words: 1 (2 bytes)
Cycles: 1 if condition is false
        2 if condition is true
    
```



10. 正数转移

BRPL — 正数转移

说明：条件相对转移，测试负号标志 (N)，如果 N 被清零，则相对 PC 值转移。该指令相对 PC 转移的方向为： $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBC 2, K）。

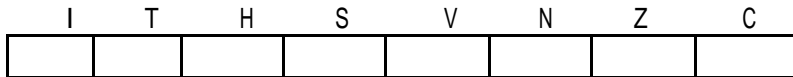
操作：If N=0 then $PC \leftarrow (PC+1) + k$, else $PC \leftarrow PC+1$

语法：操作码：程序计数器：
BRPL k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1) + k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	01kk	kkkk	K010
------	------	------	------

状态寄存器 (SREG) 和布尔格式：



```

例子: (实践操作程序 45210.ASM)
lp:subi r18,2 ; 带进位位立即数减,设:(r18)=2
BRPL lp1 ; 为正转 N=0,为负顺执 N=1 请同时观察 N 标志
inc r18 ;(r18)+1
inc r18
inc r18
inc r18
rjmp lp ;反复实验
lp1:dec r18 ;(r18)+1
dec r18
rjmp lp ;反复实验
    
```

positive: nop

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true

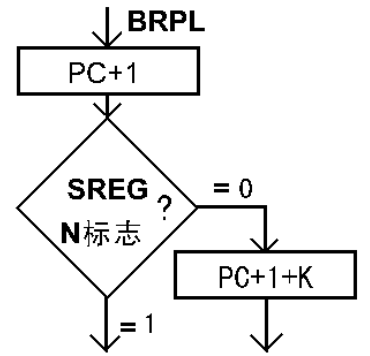
11. 大于或等于转移

BRGE—大于或等于转移 (带符号)

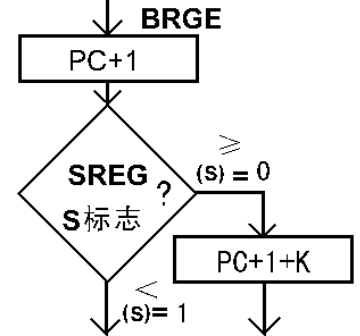
说明: 条件相对转移, 测试符号标志 (S), 如果 S 位被清零, 则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令, 且当在寄存器 Rd 中带符号二进制数大于或等于寄存器 Rr 中带符号二进制数时, 转移将发生。该指令相对 PC 转移的方向为: $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBC 4, K)。

操作: If $Rd \geq Rr$ ($N \vee = 0$) then $PC \leftarrow (PC+1)+k$, eles $PC \leftarrow PC+1$

语法: 操作码: 程序计数器:
 BRGE k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1) + k$
 $PC \leftarrow PC + 1$



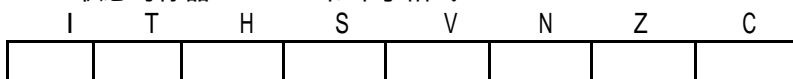
CP,CPI,SUB,SUBI 指令执行后判



16 位操作码:



状态寄存器 (SREG) 和布尔格式:



```

例子: (实践操作程序 45211.ASM)
lp:subi r18,2 ; 带进位位立即数减,设:(r18)=2
brge lp1 ; 为大于或等于转 S=0,小于顺执 S=1 请同时观察 S 标志
    
```

```
inc r18          ;(r18)+1
inc r18
inc r18
inc r18
rjmp lp         ;反复实验
lp1:dec r18     ;(r18)+1
dec r18
rjmp lp         ;反复实验
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true

12. 小于转移

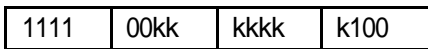
BRLT—小于转移（有符号）

说明：条件相对转移，测试符号标志（S），如果 S 位被置位。则相对 PC 值转移。如果在执行 CP、CPI、SUB 或 SUBI 指令后立即执行该指令，且当在寄存器 Rd 中带符号二进制数小于在寄存器 Rr 中带符号二进制数时，转移将发生。该指令相对 PC 转移的方向为： $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBS 4, K）。

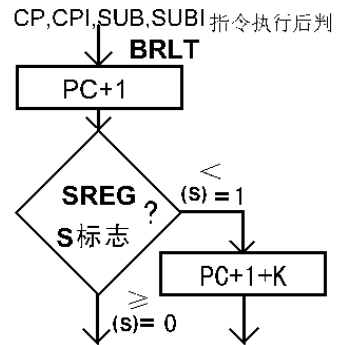
操作：If $R_d < R_r$ (N V=1) then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$

语法：操作码：程序计数器：
 BRLT k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1) + k$
 $PC \leftarrow PC+1$

16 位操作码：



状态寄存器（SREG）和布尔格式：



例子：（实践操作程序 45212.ASM）

```
lp:subi r18,2    ;带进位位立即数减,设:(r18)=2
brlt lp1        ;为小于于转 S=0,大于或等顺执 S=1 请同时观察 S 标志
inc r18         ;(r18)+1
inc r18
inc r18
inc r18
rjmp lp         ;反复实验
lp1:dec r18     ;(r18)-1
dec r18
rjmp lp         ;反复实验
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true

13. 半进位标志置位转移

BRHS--半进位标志置位转移

说明：条件相对转移，测试半进位标志（H），如果 H 被置位，则相对 PC 值转移。该指令相对 PC 转移的方向为： $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBS 5, K）。

操作： If H=1 then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$
 语法： 操作码： 程序计数器：
 BRHS k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	00kk	kkkk	k101
------	------	------	------

状态寄存器（SREG）和布尔格式：

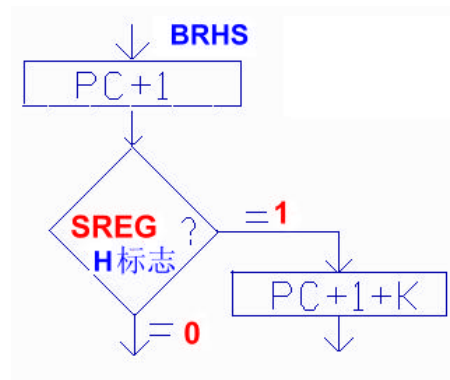
I	T	H	S	V	N	Z	C

例子：（实践操作程序 45213.ASM）

```

lp:add r10,r11
;加法设:(r10)=5 ,(r11)=5 ,(r12)=5
brhs lp1 ;半进位标志(H)=1 转移
; (H)=0 顺执
dec r10 ;(R10)-1
dec r10
rjmp lp ;反复实验
lp1:add r10,r12 ;加法
rjmp lp ;反复实验
    
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true



14. 半进位标志清零转移

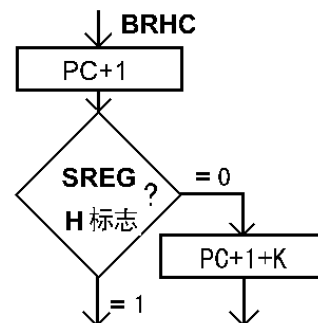
BRHC--半进位标志被清零转移

说明：条件相对转移，测试半进位标志（H），如果 H 位被清零，则相对 PC 值转移。该指令相对 PC 转移的方向为： $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBC 5, K）。

操作： If H=0 then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$
 语法： 操作码： 程序计数器：
 BRHC k $-64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	01kk	kkkk	k101
------	------	------	------



状态寄存器 (SREG)和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45214.ASM)

```
lp:add r10,r11      ;加法设:(r10)=5 ,(r11)=5 ,(r12)=5
brhc lp1           ;半进位标志(H)=0 转移,(H)=1 顺执
dec r10            ;(R10)-1
dec r10
rjmp lp           ;反复实验
lp1:add r10,r12    ;加法
rjmp lp           ;反复实验
```

```
Words:    1 ( 2 bytes)
Cycles:   1 if condition is false
          2 if conditlon is true
```

15. T 标志置位转移

BRTS—T 标志被置位转移

说明: 条件相对转移, 测试 T 标志, 如果 T 被置位, 则相对 PC 值转移。该指令相对 PC 转移的方向为: $PC - 64 \leq \text{目的} \leq PC + 63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBS 6, K)。

操作: If T=1 then $PC \leftarrow (PC+1)+k$, else $PC \leftarrow PC+1$

```
语法:          操作码:          程序计数器:
BRTS  k      -64 ≤ k ≤ +63      PC ← (PC+1) + k
                                      PC ← PC + 1
```

16 位操作码:

1111	00kk	kkkk	k110
------	------	------	------

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45215.ASM)

```
lp:bst r3,5        ;R3 中的 5 位到 SREG 中 T 标志, 设:(R3)=$AA
brts LP1          ;(T)=1 转移,(T)=0 顺执
rol r3            ;r3 通过进位位左循环
rjmp lp          ;反复实验
LP1:rol r3        ;r3 通过进位位左循环
rjmp lp          ;反复实验
```

```
Words:    1 ( 2 bytes)
Cycles :  1 if condition is false
          2 if conditlon is true
```

16. T 标志清零转移

BRBC—T 标志被清零转移

说明：条件相对转移，测试 T 标志，如果 T 被清零，则相对 PC 值转移。该指令相对 PC 转移的方向为： $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBC 6, K）。

操作：If T=0 then PC ← (PC+1)+k, else PC ← PC+1

语法： 操作码： 程序计数器：
BRBC k $64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	01kk	kkkk	k110
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

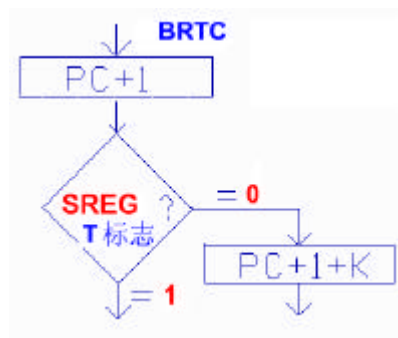
I	T	H	S	V	N	Z	C

例子：（实践操作程序 45216.ASM）

```
bst r3,5      ;R3 中的 5 位到 SREG 中 T 标志, 设 R3=$55
lp:brtc lp1   ;T 标志置 0 转移, 置 1 顺执
  clt         ;(T)=0
  rjmp lp    ;反复测试
lp1:set       ;(T)=1
  rjmp lp    ;反复测试
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
 2 if condition is true

**17. 溢出标志置位转移**

BRVS—溢出标志被置位转移

说明：条件相对转移，测试溢出标志 (V)，如果 V 被置位，则相对 PC 值转移。该对 PC 转移的方向为： $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移，用 2 的补码表示（相当于指令 BRBS3, K）。

操作：If V=1 then PC ← (PC+1)+k, else PC ← PC+1

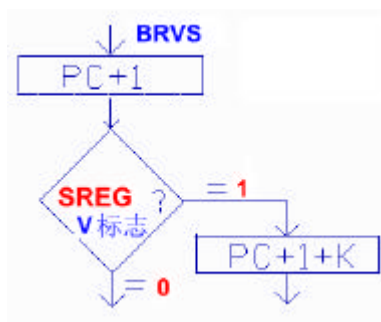
语法： 操作码： 程序计数器：
BRVS k, $-64 \leq k \leq +63$ $PC \leftarrow (PC+1)+k$
 $PC \leftarrow PC+1$

16 位操作码：

1111	00kk	kkkk	k011
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

I	T	H	S	V	N	Z	C



例子: (实践操作程序 45217.ASM)

```
lp:brvs lp1      ;溢出标志位(V)=1 转移,(V)=0 顺执
    sev         ;溢出标志位(V)置 1
    rjmp lp     ;反复测试
lp1:clv         ;溢出标志位(V)置 0
    rjmp lp     ;反复测试
```

```
words:    1 ( 2 bytes)
Cycles:   1 if condition is false
          2 if condition is true
```

18. 溢出标志清零转移

BRVC——溢出标志被清零转移

说明: 条件相对转移, 测试溢出标志 (V), 如果 V 被清零, 则相对 PC 值转移。该对 PC 转移的方向为: $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBC3, K)。

```
操作: If V=0 then PC ← (PC+1)+k, else PC ← PC+1
语法:          操作码:          程序计数器:
BRVC k        -64 ≤ k ≤ +63      PC ← (PC+1)+k
                                   PC ← PC+1
```

16 位操作码:

1111	01kk	kkkk	k011
------	------	------	------

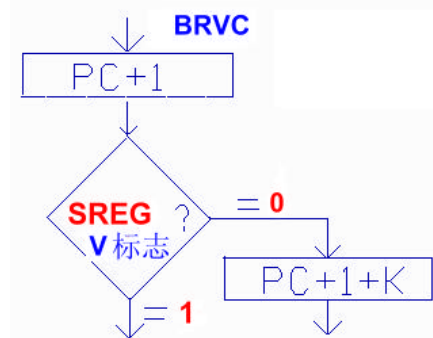
状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45218.ASM)

```
lp:brvc lp1     ;溢出标志位(V)=0 转移,(V)=1 顺执
    clv         ;溢出标志位(V)置 0
    rjmp lp     ;反复测试
lp1:sev         ;溢出标志位(V)置 1
    rjmp lp     ;反复测试
```

```
Words:    1 (2 bytes)
Cyclse:   lif condition is false
          2 if conition is true
```



19. 中断标志触发转移

BRIF——全局中断被触发转移

说明: 条件相对转移, 测试全局中断标志 (I), 如果 I 被置位 1, 则相对 PC 值转移。该指令相对 PC 转移的方向为: $PC-64 \leq \text{目的} \leq PC+63$ 。参数 K 为 PC 的偏移, 用 2 的补码表示 (相当于指令 BRBS7, K)。

```
操作: If I=1 then PC ← (PC+1)+k, else PC ← PC+1
语法:          操作码:          程序计数器:
BRIF k        -64 ≤ k ≤ +63      PC ← (PC+1)+k
```

PC←PC+1

16 位操作码:

1111	00kk	kkkk	k111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

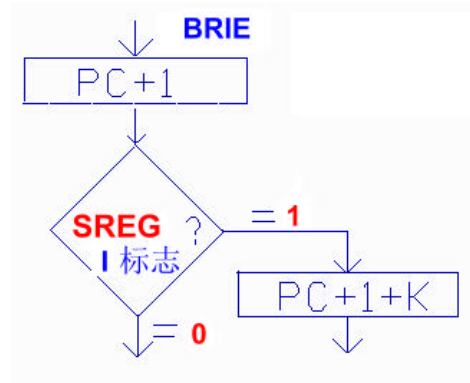
例子: (实践操作程序 45219.ASM)

LP:BRIE LP1 ;全局中断标志(I)=1 转移,
;用鼠标在状态寄存器窗口中 I 标志点一下



; (I)=0 顺执
SEI ;(I)=1
RJMP LP ;反复测试
LP1:CLI ;(I)=0
RJMP LP ;反复测试

...
words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true



20. 中断标志禁止转移

BRID 一全局中断被禁止转移

说明. 条件相对转移. 测试全局中断标志 (I), 如果 I 被清零, 则相对 PC 但转移. 该指令相对 PC 转移的方向为: PC-64≤目的≤PC+ 63. 参数 K 为 PC 的 b 偏移, 用 2 的补码表示 (相当于指令 BRBC 7, K)。

操作: If I=0 then PC←(PC+1)+ k, else PC←PC+ 1

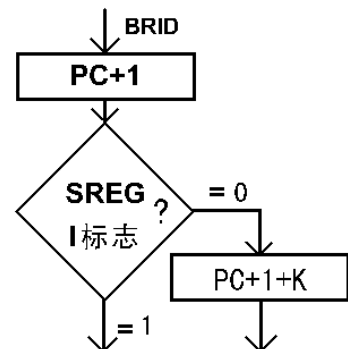
语法: 操作码: 程序计数器:
BRID k -64≤ k ≤+63 PC←(PC+1)+ k
PC←PC+1

16 位操作码:

1111	01kk	kkkk	k111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

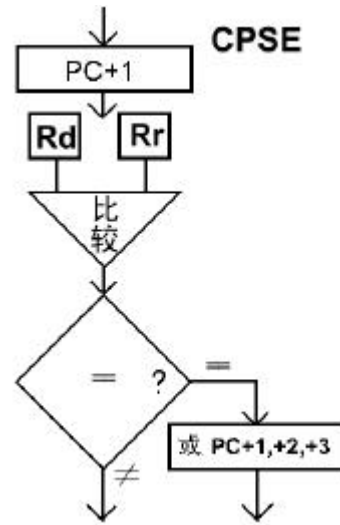


例子: (实践操作程序 45220.ASM)

```

LP:BRID LP1 ;全局中断标志(I)=0 转移, (I)=1 顺执
  CLI      ;(I)=0
  RJMP LP  ;反复测试
LP1:SEI   ;(I)=1
  RJMP LP  ;反复测试
    
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false
 2 if condition is true



二、测试条件符合跳行转移指令

21. 相等跳行

CPSE—比较相等跳行

说明: 该指令完成两个寄存器 Rd 和 Rr 的比较,
 若 Rd=Rr, 则跳行执行指令。

操作: If Rd=Rr then PC←PC+2 (or 3) else PC←PC+1

语法: 操作码: 程序计数器:
 CPSE Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1
 PC ← PC + 2
 PC ← PC + 3

16 位操作码:

0001	00rd	dddd	rrrr
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45221.ASM)

```

lp:inc r4 ;设 (r4)=$77 ,(r0)=$77
      ;设 (r4)=$76 ,(r0)=$77
lp1:cpse r4,r0 ;r4 与 r0 比较相等跳行,不等顺执
lds r10,$0100 ;这是四字节指令把 SRAM 地址 $0100 的数据装入 R10
nop
dec r4 ;(r4)-1
cpse r4,r0 ;r4 与 r0 比较相等跳行,不等顺执
rjmp lp1 ;这是二字节指令
mov r10,r0 ;拷贝
rjmp lp ;反复测试
    
```

Words: 1 (2 bytes)
 Cycles: 1

22. 寄存器位清零跳行

SBRC —寄存器位被清零跳行

说明: 该指令测试寄存器某位, 如果该位被清零, 则跳下一行执行指令。

操作: If Rd (b) = 0 then PC←PC+2 (or 3) eles PC←PC+ 1

语法: 操作码: 程序计数器:
 SBRC Rr, b 0≤r≤31, 0≤b≤7 PC←PC+1
 pC←pC+2
 PC←PC+3

16 位操作码:

1111	110r	rrrr	Xbbb
------	------	------	------

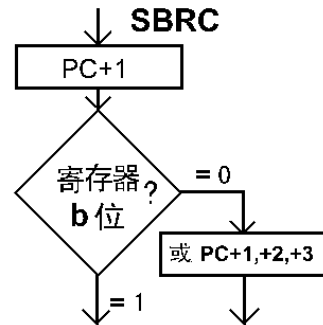
状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45222.ASM)

```
lp:sub r0,r1 ;设:(r0)=$80,(r1)=$70
sbrc r0,7 ;r0 的 7 位清零跳行,为 1 顺执
sub r0,r1 ;这是二字节指令
swap r0 ;r0 半字节交换
rjmp lp ;反复测试
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false(no skip)
 2 if condition is true (skip is exected)



23. 寄存器位置位跳行

SBRS — 寄存器位置位跳行

说明: 该指令测试寄存器某位, 如果该位被置位, 则跳下一行执行指令。

操作: If Rr(b) =1 then PC←PC+ 2 (or 3) eles PC←PC+ 1

语法: 操作码: 程序计数器:
 SBRS Rr, b ≤r≤31, 0≤b≤7 PC←PC+1
 PC←PC+2
 PC←PC+3

16 位操作码:

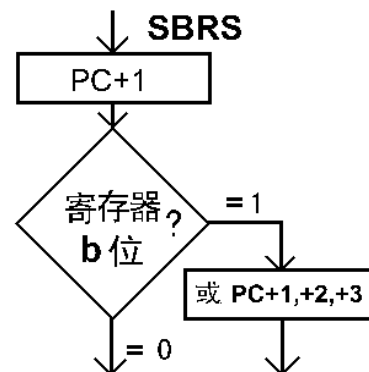
1111	111r	rrrr	Xbbb
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45223.ASM)

```
lp:sub r0,r1 ;设:(r0)=$80,(r1)=$70
sbrs r0,7 ;r0 的 7 位置位跳行,为 0 顺执
sub r0,r1 ;这是二字节指令
swap r0 ;r0 半字节交换
```



rjmp lp ;反复测试

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

24. I/O 寄存器位清零跳行

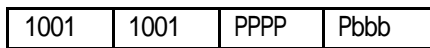
SBIC—I/O 寄存器的位清零跳行

说明: 该指令测试 I/O 寄存器某位, 如果该位被清零, 则跳一行执行指令。该指令在低 32 个 I/O 寄存器内操作, 地址为. 0~31。

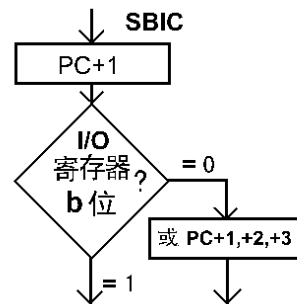
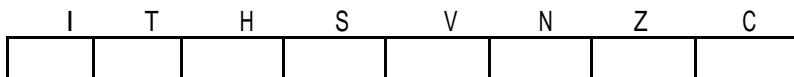
操作: If I/OP, b= 0 then PC<PC+2 (or 3) eles PC<C+1

语法: SBIC P,b 操作码: 程序计数器:
0≤P≤31, 0≤b≤7 PC<PC+1
PC<PC+2
PC<PC+3

16 位操作码:

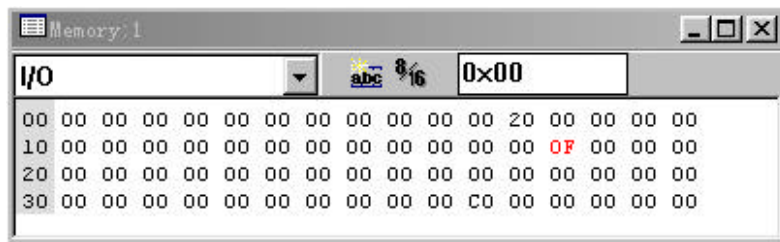


状态寄存器 (SREG) 和布尔格式:



例子: (实践操作程序 45224.ASM)

```
lp:sbic $1c,1 ;复位后(EECR)=$00, .equ EECR=$1c,测试 1 位,为跳 0 行,为 1 顺执
;请打开 I/O 窗口观察,注意:这条指令只能连续测试 1 次,
;第 2 次会改变$1C 数据(反复测试不能用该指令)
rjmp lp ;这是二字节指令
ldi R16,$0f ;
OUT $1C,R16 ;置 i/o 寄存器 1 位为 1
rjmp lp
```



Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

25. I/O 寄存器位置位跳行

SBIS—I/O 寄存器的位置位跳行

说明: 该指令测试 I/O 寄存器某位, 如果该位被置位, 则跳一行执行指令。该指令在低

32 个 I/O 寄存器内操作，地址为 0~31。

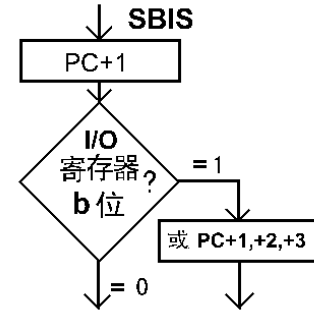
操作：If I/O, b=1 then PC←PC+2 (or 3) else PC←PC+1

语法：操作码：程序计数器：
 SBIS p , b 0 ≤ P ≤ 31, 0 ≤ b ≤ 7 PC←PC+ 1
 PC←PC+ 2
 16 位操作码： PC←PC+ 3

1001	1011	PPPP	Pbbb
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C



例子: (实践操作程序 45225.ASM)

```

lp: ldi R16,$0f      ;
    OUT $1C,R16      ;置 i/o 寄存器 1 位为 1
lp1: sbis $1c,1     ;复位后(EECR)=$00, .equ EECR=$1c,测试 1 位,为跳 1 行,为 0 顺执
    ;请打开 I/O 窗口观察,
;注意:这条指令只能连续测试 1 次,第 2 次会改变$1C 数据(反复测试不能用该指令),见 45225B.ASM
    rjmp lp          ;这是二字节指令
    ldi R16,$00     ;
    OUT $1C,R16     ;置 i/o 寄存器 1 位为 0
    rjmp lp1
    
```

Words: 1 (2 bytes)
 Cycles: 1 if condition is false (no skip)
 2 if condition is true (skip is executed)

三、调用和返回指令

在程序设计中通常把具有一定功能模块的公用程序段定义为子程序。为了实现调用子程序的功能，指令系统中都有调用指令。也称转移子程序指令，它与转移指令的区别如下。执行调用子程序时，把下一条指令地址 (PC 值) 保留到堆栈中，即断点保护，然后把子程序的起始地址置入 PC。子程序执行完毕，再从断点返回，从断点处继续执行原程序。而转移指令即不保护断点，也不返回原程序。在每个子程序中都必须有返回指令，返回指令的功能就是把调用前压入堆栈的断点弹出置入 PC，恢复调用前的原程序。

在一个程序中，子程序中还会调用别的子程序，这称为子程序嵌套。每次调用子程序时必须将下条指令地址保存起来，返回时按后进先出原则依次取出旧 PC 值。堆栈就是按后进先出规律存取数据的，调用指令和返回指令具有自动的保存和恢复 PC 内容的功能，即自动进栈，自动出栈。

26. 相对调用

RCALL—相对调用于程序

说明：在 PC+1 后 (2K 字 (4K 字节) 范围内调用子程序。返回地址 (RCALL 后的指令地址) 存储到堆栈 (见 CALL)。

操作：PC←(PC+1)+k
 语法：操作码：程序计数器：
 RCALL k -2K ≤ k ≤ 2K PC←(PC+1)+ k
 16 位操作码：

1101	kkkk	kkkk	kkkk
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

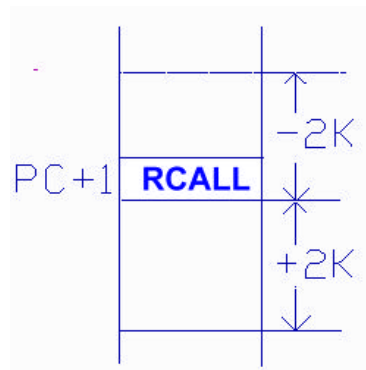
I	T	H	S	V	N	Z	C

例子: (实践操作程序 DIP40LED.ASM)

```

ser temp          ;直接装入$FF,
out DDRA, temp   ;口的方向寄存器设定,为输出
forever:
clr temp          ;硬件设低电平 LED 灯亮
out PORTA, temp  ;PORTA 口 LED 灯亮
ldi temp,1200    ;装延时常数(十进制),灯亮延时,
rcall delay_mS   ;调用延时子程序
ser temp          ;硬件设高电平 LED 灯灭
out PORTA, temp  ;PORTA 口 LED 灯灭
ldi temp,1200    ;装延时常数,灯灭延时,可修改该参数
rcall delay_mS   ;调用延时子程序
rjmp forever     ;无限循环

```



```

delay_mS:        ;延时子程序
ldi zl,low(2500) ;装入延时参数,参数可修改
ldi zh,high(2500);参数可修改
delay_loop:
dec zl           ;-1
brne delay_loop ;不相等转移,相等按顺序执行
dec zh           ;-1
brne delay_loop ;不相等转移,相等按顺序执行
dec temp         ;-1
brne delay_mS    ;不相等转移,相等按顺序执行
ret              ;子程序返回

Words:          1 ( 2 bytes )
Cycles:         3

```

27. 间接调用

ICALL—间接调用于程序

说明: 间接调用由寄存器区中的 Z (16 位) 指针寄存器指向的子程序。Z 指针寄存器是 16 位宽, 允许调用当前程序存储空间 64K 字 (128 字节) 内的子程序。

操作: PC (15—0) ← Z (15—0)
PC (15—0) ← Z (15—0)

语法: ICALL 操作码: None 程序计数器: See operation


16 位操作码:

1001	0101	XXXX	1001
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45227.ASM)

```
.ORG $0010
LP:MOV R30,R0 ;设(R0)为Z寄存器低8位地址,R30为Z间接地址低8位
    MOV R31,R1 ;设(R1)为Z寄存器高8位地址,R31为Z间接地址高8位
    ICALL      ;根据Z寄存器的内容调用,该步用进入子程序图标  调试
.ORG $0200
LDI R20,$11 ;地址$0200,如设(R0)=$00,(R1)=02,执行ICALL到此行
LDI R21,$22 ;地址$0201,如设(R0)=$01,(R1)=02,执行ICALL到此行
LDI R22,$33 ;地址$0202,如设(R0)=$02,(R1)=02,执行ICALL到此行
LDI R23,$44 ;地址$0203,如设(R0)=$03,(R1)=02,执行ICALL到此行
LDI R24,$55 ;地址$0204,如设(R0)=$04,(R1)=02,执行ICALL到此行
LDI R25,$66 ;地址$0205,如设(R0)=$05,(R1)=02,执行ICALL到此行
ICALL      ;地址$0205,如设(R0)=$06,(R1)=02,执行ICALL到此行
            ;又根据Z寄存器的内容调用到指定地址
    move r30, r0
    icall
Words:     1 ( Zbytes)
Cycles:    3
```

28. 长调用

CALL—子程序长调用

说明: 在整个程序存储器区内调用子程序。返回地址 (调用后返回的指令地址) 将存储在堆栈 (见 RCALL 指令) 中。

操作: PC←k
PC←k

语法: 操作码:
CALL k 0 ≤ k ≤ 64K

CALL k 0 ≤ k ≤ 4M

程序计数器:
PC←k STACK←PC+2
SP←SP-2
PC←k STACK←PC+2
SP←SP-3

32 位操作码:


1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 45228.ASM)

```

    move    r16,r0
    call    delay_mS      ;长调用延时子程序,该步用进入子程序图标  调试
    .ORG $0200
delay_mS:                ;延时子程序
    ldi    zl,low(1990)
    ldi    zh,high(1990)
delay_loop:
    dec    zl
    brne   delay_loop
    dec    zh
    brne   delay_loop
    dec    temp
    brne   delay_mS
    ret                    ;子程序返回

Words:    2 (4 bytes)
Cycles:    4
    
```

29. 从子程序返回

RET—子程序返回

说明: 从子程序返回。返回地址从堆栈中弹出。

操作: PC (15—0) ← STACK
 PC (21—0) ← STACK

语法:	操作码:	程序计数器:	堆栈:
RET	None	See Operation	SP←SP+ 2
RET	None	See Operation	SP←SP+ 3

16 位操作码:

1001	0101	0XX0	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序见 45228.ASM)

```

RCALL LP
LP:LDI R16,0X56      ;装入延时常数,延时 1 秒
RCALL DELY          ;调用通用延时子程序
RET                 ;子程序返回
Words:              1 ( Zbytes)
Cyclse:             4
    
```

30. 从中断程序返回

RETI—中断返回

说明: 从中断程序中返回。返回地址从堆栈中弹出,且全局中断标志被置位。

- 注意: 1. 主程序应跳过中断区,防止修改补充中断程序带来麻烦;
 2. 不用的中断入口地址写上 RETI - 中断返回,有抗干扰作用;

操作: PC (15—0) ← STACK
 PC (21—0) ← STACK

语法:	操作码:	程序计数器:	堆栈:
RETI	None	See Operation	SP←SP + 2
RETI	None	See Operation	SP←SP + 3

16 位操作码:

1001	0101	0XX0	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (程序 45230.ASM,摘自”乐曲.ASM”部分程序,仅供参考),能执行程序请阅”乐曲.ASM”程序及 AVR 单片机在儿童智能玩具中的应用--音乐玩具(电脑放音机)一文

```
.cseg
.org 0x06 ;timer1 中断入口地址
intt1: RJMP OUTPM ;转中断服务子程序
.cseg
.org 0x010 ;
OUTPM: OUT TCNT1H,TONH ; 中断服务子程序
      OUT TCNT1L,TONL ;
      SBIS PORTC,00 ;
      RJMP SETOP1 ;
SETOP0: CBI PORTC,00 ;
      LDI MUSN,$00 ;
      RETI ;中断返回

SETOP1: SBI PORTC,00 ;
      LDI MUSN,$01 ;
      RETI ;中断返回
      extint: push r0
                        pop r0
                        reti

WOrds: 1 (Zbytes)
Cyclse: 4
```

4.6 数据传送指令

数据传送指令是在编程时使用最频繁的一类指令。数据传送指令是否灵活快速对程序的执行速度产生很大影响。数据传送指令执行操作是寄存器与寄存器，寄存器与数据存储器（SRAM），寄存器与 I/O 端口之间的数据传送。另外还有从程序存储器直接取数指令 LPM 以及 PUSH（压栈）和 POP（出栈）的堆栈指令。

4.6.1 直接数据传送指令

1. 寄存器拷贝数据

MOV 寄存器拷贝

说明：该指令将一个寄存器拷贝到另一个寄存器。源寄存器 Rr 的内容不改变，而目的寄存器 Rd 拷贝了 Rr 的内容。

操作：Rd ← Rr

语法：

MOV Rd Rr $0 \leq d \leq 31, 0 \leq r \leq 31$

16 位操作码：

操作码：

程序计数器：

PC ← PC + 1

0010	11rd	dddd	rrrr
------	------	------	------

状态寄存器（SRE）和布尔格式：

I	T	H	S	V	N	Z	C

例子：（实践操作程序 4611.ASM）

```
lp:mov r16,r0    ;把 R0 的内容传送到 R16 中
rcall check    ;调用子程序
rjmp lp        ;反复实验
.org $0100     ;子程序首址
check:swap r0  ; r0 半字节交换
ret           ;子程序返回
```

Words: 1 (2 bytes)

Cycles: 1

2. SRAM 数据直接送寄存器

LDS 直接从 SRAM 装入

说明：把 SRAM 中 1 个字节装入到寄存器。必须提供一个 16 位地址。存储器访问被限制在当前 64K 字节的 SRAM 页。超过 64K 字节，LDS 指令使用 RAMPZ 寄存器访问。

操作：Rd ← (k)

语法：

LDS Rd k $0 \leq d \leq 31, 0 \leq k \leq 65535$

32 位操作码：

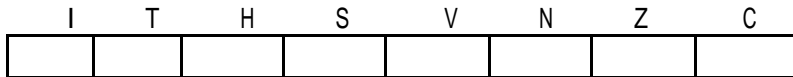
操作码：

程序计数器：

PC ← PC + 2

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

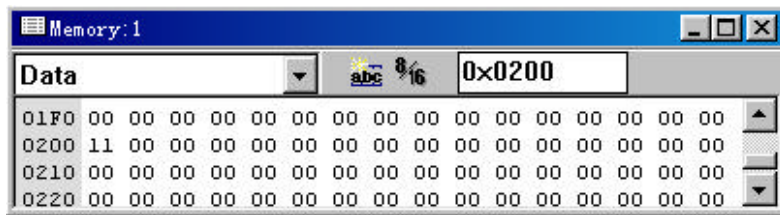
状态寄存器（SREG）和布尔格式：



例子: (实践操作程序 4612.ASM)

```

LP: lds r2,$0200 ;把片内 SRAM 地址$0200 数据装入, 设: ($0200)=$11
    add r2,r1    ;R2 与 R0 内容相加, 结果存于 R2 中, 设: (r1)=$88
    sts $0200,r2 ;R2 的内容送到片内 SRAM 地址$0200
    RJMP LP     ;打开片内 SRAM 窗口观察, 反复测试
    
```



Words: 2 (4 bytes)
Cycles: 3

3. 寄存器数据直接送 SRAM

STS 寄存器数据直接送 SRAM

说明: 将寄存器的内容直接存储到 SRAM。必须提供一个 16 位的地址。存储器访问被限制在当前 64K 字节的 SRAM 页。STS 指令使用 RAMPZ 寄存器访问存储器可超过 64K 字节。

操作: (k) ←Rr

语法:

STS k, Rr

32 位操作码:

操作码:

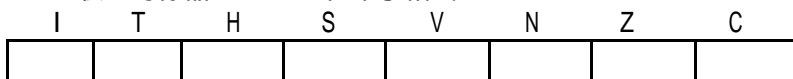
0 ≤ r ≤ 31, 0 ≤ k ≤ 65535

程序计数器:

PC ← PC + 2

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

状态寄存器 (SREG) 和布尔格式:



例子: (实践操作程序 4613.ASM 与 4612.ASM 相同)

```

LP: lds r2,$0200 ;把片内 SRAM 地址$0200 数据装入, 设: ($0200)=$11
    add r2,r1    ;R2 与 R0 内容相加, 结果存于 R2 中, 设: (r1)=$88
    sts $0200,r2 ;R2 的内容送到片内 SRAM 地址$0200
    RJMP LP     ;打开片内 SRAM 窗口观察, 反复测试
    
```

Words: 2 (4 bytes)
Cycles: 3

4. 立即数送寄存器

LDI—装入立即数

说明: 装入一个 8 位立即数到寄存器 R16~R31 中。

操作: $Rd \leftarrow K$

语法:

LDI Rd K

16 位操作码:

操作码:

$16 \leqq d \leqq 31, 0 \leqq K \leqq 255$

程序计数器:

$PC \leftarrow PC + 2$

1110	KKKK	dddd	KKKK
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4614.ASM)

clr r31 ;R31 清零

ldi r30,\$F0 ;立即数送 R30 中, $R16 \leqq R \leqq R31$

lpm ;装入程序存储器,请观察 Z 寄存器内容

Words: 1 (2 bytes)

Cycles: 1

4.6.2 间接数据传送指令

一、使用 X 寄存器间接传送数据

1. 使用变址 X 间接将 SRAM 中内容送入到寄存器

LD —使用变址 X 间接将 SRAM 中内容送入到寄存器

说明: 从 SRAM 中间送入一个字节到寄存器, SRAM 中的位置由寄存器区中的 X (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页中。为访问另外 SRAM 页, 则 I/O 范围内的寄存器 RAMPX 需改变。在指令执行中, X 指针寄存器值要么不改变, 要么就加 1 或减 1 操作。使用 X 指针寄存器的这些特性, 特别适合于访问矩阵、表和堆栈指针等。

操作: $Rd \leftarrow (X)$; 送数, X 指针寄存器值不改变
 $Rd \leftarrow (X)$ $X \leftarrow X + 1$; 先送数, 后 X 指针寄存器值加 1
 $X \leftarrow X - 1$ $Rd \leftarrow (X)$; 先 X 指针寄存器值减 1, 后送数

语法:

LD Rd, X

LD Rd, X+

LD Rd, -X

操作码:

$0 \leqq d \leqq 31$

$0 \leqq d \leqq 31$

$0 \leqq d \leqq 31$

操作流程

送数, X 指针不改变

先送数, 后 X 指针加 1

先 X 指针减 1, 后送数

程序计数器:

$PC \leftarrow PC + 1$

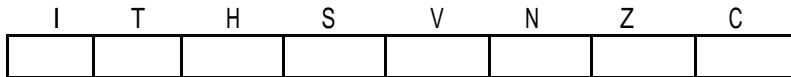
$PC \leftarrow PC + 1$

$PC \leftarrow PC + 1$

16 位操作码:

1.	1001	000d	dddd	1100
2.	1001	000d	dddd	1101
3.	1001	000d	dddd	1110

状态寄存器 (SREG) 和布尔格式:

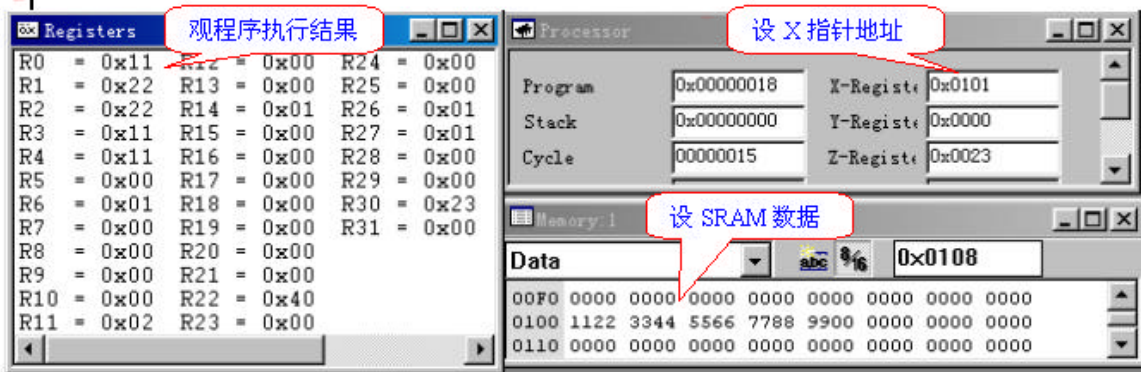


例子: (实践操作程序 4621.ASM)

LP:

```

clr r31 ;清零 Z 寄存器高位
ldi r30,$20 ;$20 送 Z 寄存器低位
ld r0,X+ ;执行 X 寄存器,X+1 为地址的 SRAM 内容送 R0,设:SRAM($0100)=$11
;($0101)=$22 ,($0102)=$33;再设 X 寄存器(X)=$0100
;这时(X)=$0100,先单步执行 (R0)=$11 ,然后(X)+1=$0101
; X 寄存器高位地址为 R27, 低位地址为 R26, SRAM 地址≥$60
    
```



```

ld r1,X ;这时(X)=$0101 单步执行后,(R1)=$22
ldi r30,$23 ; 单步执行后 (R30)=$23
ld r2,X ;这时(X)=$0101,单步执行后 (R2)=$22
ld r3,-X ;单步先执行(X)-1=$0100,然后执行(R3)=$11
ld r4,X+ ;先单步执行,这时(X)=$0100,(R4)=$11,然后执行(X)+1=$0101
NOP ;再从寄存器窗口看程序执行情况
RJMP LP
    
```

Words: 1 (2 bytes)
Cycles: 2

2. 使用变址 X 间接将寄存器内容传送到 SRAM

ST—使用变址 X 间接将寄存器内容传送到 SRAM

说明: 间接将寄存器的一个字节传送到 SRAM。SRAM 的位置由寄存器区中的 X (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页中。为访问另外 SRAM 页, 则 I/O 范围的寄存器 RAMPX 将被修改。在操作之后, X 指针寄存器要么不改变, 要么是加 1 或减 1。使用 X 指针寄存器的这些特性, 特别适合用作堆栈指针。

操作: (X)← Rr
X ← Rr X←X+1
X ← X-1 (X)←Rr

语法: ST X,Rr 操作码: 0≤ d≤31 操作流程: 送数,X 指针不改变 程序计数器: PC←PC+1

ST X+,Rr 0≤d≤31 先送数,后 X 指针加 1 PC←PC+1
 ST -X,Rr 0≤d≤31 先 X 指针减 1, 后送数 PC←PC+1

16 位操作码:

1.	1001	001r	rrrr	1100
2.	1001	001r	rrrr	1101
3.	1001	001r	rrrr	1110

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4622.ASM)

```

lp:
clr r27 ;X 寄存器高位清零, X 寄存器高位地址为 R27, 低位地址为 R26
ldi r26,$70 ;$70 送 X 寄存器低位, SRAM 地址≥$60
st X+,r0 ;设 (R0)=$11, 先单步执行送 SRAM(X)=$0070, 然后(X)+1=$0071
st X,r1 ;设 (R1)=$22, 这时(X)=$0071, 单步执行后 SRAM($0071)=$22
ldi r26,$80 ;$80 送 X 寄存器低位
st x,r2 ;设 (R2)=$33, 这时(X)=$0080, 单步执行后 SRAM($0080)=$33
st -x,r3 ;设 (R3)=$44, 这时先单步执行(X)-1=$007F, 然后 R3 内容送 SRAM($007F)
nop ;
rjmp lp ;反复测试
    
```

Words: 1 (2 bytes)
 Cycles: 2

二、使用 Y 寄存器间接传送数据

3. 使用变址 Y 间接将 SRAM 中的内容传送到寄存器

LD (LDD) 一使用变址 Y 间接将 SRAM 中的内容传送到寄存器

说明: 带或不带偏移间接从 SRAM 中传送一个字节到寄存器, SRAM 中的位置由寄存器区中的 Y (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页中。为访问另外 SRAM 页, 则 I/O 范围内的寄存器 RAMPY 需改变。在指令执行后, Y 指针寄存器值要么不改变, 要么就加 1 或减 1 操作。使用 Y 指针寄存器的这些特性, 特别适合于访问矩阵、表和堆栈指针等。

操作: Rd←(Y)
 Rd←(Y)
 Y←Y-1 Y←Y+ 1
 Rd←(Y+q) Rd←(Y)

语法:	操作码:	操作流程:	程序计数器:
LD Rd,Y	0≤d≤31	送数, Y 指针不改变	PC←PC+1
LD Rd Y+	0≤d≤31	先送数, 后 Y 指针加 1	PC←PC+1
LD Rd, -Y	0≤d≤31	先 Y 指针减 1, 后送数	PC←PC+1
LDD Rd Y+q	0≤d≤31, 0≤q≤63	先 Y 指针加 q, 后送数, 执行后 Y 指针(Y 不含 q)不变	PC←PC+1

16 位操作码:

1.	1000	000d	dddd	1000
2.	1001	000d	dddd	1001
3.	1001	000d	dddd	1010
4.	10q0	qq0d	dddd	1qqq

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4623.ASM)

```

LP:
clr r29 ;Y 寄存器高位清零, Y 寄存器高位地址为 R29, 低位地址为 R28
ldi r28,$60 ;将$60 送 Y 寄存器低位 SRAM 地址≥$60
ld r0,y+ ;;执行 Y 寄存器, Y+1 为地址的 SRAM 内容送 R0, 设: SRAM($0060)=$11
;($0061)=$22, ($0062)=$33;这时(Y)=$0060, 先单步执行 (R0)=$11, 然后(Y)+1=$0061
ld r1,y ;这时(Y)=$0061, 单步执行后, (R1)=$22
ldi r28,$70 ;将$70 送 Y 寄存器低位
ld r2,y ;这时(Y)=$0070, 设($0070)=$44, ($006F)=$55, ($0071)=$66
;单步执行后(R2)=$44
ld r3,-y ;单步先执行(Y)-1=$006F, 然后单步执行(R3)=$55
ldd r4,y+2 ;单步执行, 先找增量地址(Y+Q)=(Y+2)=$006F+2=$071, 后送数(R4)=$66,
;但 Y 指计内容不变(Y 不含 q)(Y)=$006F
nop ;
rjmp lp ;反复测试

Words: 1 (2 bytes)
Cycles: 2
    
```

4. 使用变址 Y 间接将寄存器内容传送到 SRAM

ST (STD) —使用变址 Y 间接将寄存器内容传送到 SRAM

说明: 间接将带或不带偏移的寄存器的一个字节传送到 SRAM。SRAM 的位置由寄存器区中的 Y (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页。为访问另外 SRAM 页, 则 I/O 范围的寄存器 RAMPY 将被修改。在操作之后, Y 指针寄存器要么不改变, 要么是加 1 或减 1。使用 Y 指针寄存器的这些特性, 特别适合用作堆栈指针。

操作: $(Y) \leftarrow Rr$
 $(Y) \leftarrow Rr \quad Y \leftarrow Y + 1$
 $Y \leftarrow Y - 1 \quad (Y) \leftarrow Rr$
 $(Y + q) \leftarrow Rr$

语法:	操作码:	操作流程:	程序计数器:
ST Y, Rr	$0 \leq d \leq 31$	送数, Y 指针不改变	$PC \leftarrow PC + 1$
ST Y+, Rr	$0 \leq d \leq 31$	先送数, 后 Y 指针加 1	$PC \leftarrow PC + 1$
ST -Y, Rr	$0 \leq d \leq 31$	先 Y 指针减 1, 后送数	$PC \leftarrow PC + 1$
STD Y+q, Rr	$0 \leq d \leq 31,$ $0 \leq q \leq 63$	先 Y 指针加 q, 后送数, 执行后 Y 指针(Y 不含 q)不变	$PC \leftarrow PC + 1$

16 位操作码:

1.	1000	001r	rrrr	1000
2.	1001	001r	rrrr	1001
3.	1001	001r	rrrr	1010
4.	10q0	Qq1r	rrrr	1qqqq

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4624.ASM)

LP:

```

clr r29 ;Y 寄存器高位清零 Y 寄存器高位地址为 R29, 低位地址为 R28
ldi r28,$70 ;$70 送 Y 寄存器低位, SRAM 地址≥$60
st y+,r0 ;设(R0)=$11,先单步执行送 SRAM(Y)=$0070,然后(Y)+1=$0071
st y,r1 ;设(R1)=$22,这时(Y)=$0071,单步执行后 SRAM($0071)=$22
ldi r28,$80 ;$80 送 Y 寄存器低位
st y,r2 ;设(R2)=$33,这时(Y)=$0080,单步执行后 SRAM($0080)=$33
st -y,r3 ;设(R3)=$44,这时先单步执行(Y)-1=$007F,然后 R3 内容送 SRAM($007F)
std y+2,r4 ;设(R4)=$55,
;单步执行,先计算出增量地址(Y+q)=(y+2)($007F+2)=$0081,后传送数据 SRAM($0081)=$55,
;但这时 Y 指针内容不变(Y 不含 q)(Y)=$007F
nop ;
rjmp lp ;反复测试

Words: 1 ( 2 bytes)
Cycles: 2

```

三、使用 Z 寄存器间接传送数据

5. 使用变址 Z 间接将 SRAM 中的内容传送到寄存器

LD (LDD) —使用变址 Z 间接将 SRAM 中的内容传送到寄存器

说明: 带或不带偏移间接从 SRAM 中传送一个字节到寄存器, SRAM 中的位置由寄存器区中的 Z (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页中。为访问另外 SRAM 页, 则 I/O 范围内的寄存器 RAMPZ 需改变。在指令执行后, Z 指针寄存器值要么不改变, 要么就加 1 或减 1 操作。使用 Z 指针寄存器的这些特性, 特别适合于堆栈指针, 因为 Z 指针寄存器能用于直接子程序调用, 直接跳转和查表。Z 指针寄存器用作为专用堆栈指针要比 X、Y 指针方便。

用 Z 指针在程序存储器中查表, 可参见 LPM 指令。

操作: $Rd \leftarrow (Z)$
 $Rd \leftarrow (Z)$
 $Z \leftarrow Z-1$
 $Rd \leftarrow (Z+q)$

语法: LD Rd,Z 操作码: $0 \leq d \leq 31$ 操作流程: 送数,Z 指针不改变 程序计数器: $PC \leftarrow PC+1$

LD Rd Z+ $0 \leq d \leq 31$ 先送数,后 Z 指针加 1 $PC \leftarrow PC+1$
 LD Rd, -Z $0 \leq d \leq 31$ 先 Z 指针减 1, 后送数 $PC \leftarrow PC+1$
 LDD Rd Z+q $0 \leq d \leq 31,$ 先 Z 指针加 q, 后送数, $PC \leftarrow PC+1$
 $0 \leq q \leq 63$ 执行后 Z 指针(Z 不含 q)不变

16 位操作码:

1.	1000	000d	dddd	0000
2.	1001	000d	dddd	0001
3.	1001	000d	dddd	0010
4.	10q0	qq0d	dddd	0qqq

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4625.ASM)

```

LP:
clr r31 ;Z 寄存器高位清零,Z 寄存器高位地址为 R31, 低位地址为 R30
ldi r30,$60 ;将$60 送 Z 寄存器低位 SRAM 地址  $\geq$  $60
ld r0,Z+ ;;执行 Z 寄存器,Z+1 为地址的 SRAM 内容送 R0, 设:SRAM($0060)=$11
;($0061)=$22, ($0062)=$33;这时(Z)=$0060, 先单步执行 (R0)=$11, 然后(Z)+1=$0061
ld r1,Z ;这时(Z)=$0061, 单步执行后, (R1)=$22
ldi r30,$70 ;将$70 送 Y 寄存器低位
ld r2,Z ;这时(Z)=$0070, 设($0070)=$44, ($006F)=$55, ($0071)=$66
;单步执行后(R2)=$44
ld r3, -Z ;单步先执行(Z)-1=$006F, 然后单步执行(R3)=$55
ldd r4,Z+2 ;单步执行, 先找增量地址(Z+Q)=(Z+2)=$006F+2=$071, 后送数(R4)=$66,
;但 Z 指计内容不变(Z 不含 q)(Z)=$006F
nop ;
rjmp lp ;反复测试
    
```

Words: 1 (2 bytes)

Cycles: 2

6. 使用变址 Z 间接将寄存器内容传送到 SRAM

ST (STD) —使用变址 Z 间接将寄存器内容传送到 SRAM

说明: 间接将带或不带偏移的寄存器的一个字节传送到 SRAM。SRAM 的位置由寄存器区中的 Z (16 位) 指针寄存器指出。存储器访问被限制在当前 64K 字节的 SRAM 页。为访问另外 SRAM 页, 则 I/O 范围的寄存器 RAMPZ 将被修改。在操作之后, Z 指针寄存器要么不改变, 要么是加 1 或减 1。使用 Z 指针寄存器的这些特性, 特别适合作堆栈指针。因为 Z 指针寄存器能适用于间接子程序调用, 间接跳转和查表, 所以 Z 指针寄存器像一个专用堆栈指针, 用起来比 X 和 Y 指针更方便。

操作: $(Z) \leftarrow Rr$
 $(Z) \leftarrow Rr \quad Z \leftarrow Z + 1$
 $Z \leftarrow Z - 1 \quad (Z) \leftarrow Rr$

$(Z+q) \leftarrow Rr$

语法:	操作码:	操作流程:	程序计数器:
ST Z,Rr	$0 \leq d \leq 31$	送数,Z 指针不改变	$PC \leftarrow PC+1$
ST Z+,Rr	$0 \leq d \leq 31$	先送数,后 Z 指针加 1	$PC \leftarrow PC+1$
ST -Z, Rr	$0 \leq d \leq 31$	先 Z 指针减 1, 后送数	$PC \leftarrow PC+1$
STD Z+q,Rr	$0 \leq d \leq 31,$ $0 \leq q \leq 63$	先 Z 指针加 q, 后送数, 执行后 Z 指针(Z 不含 q)不变	$PC \leftarrow PC+1$

16 位操作码:

1.	1000	001r	rrrr	0000
2.	1001	001r	rrrr	0001
3.	1001	001r	rrrr	0010
4.	10q0	qq1r	rrrr	0qqq

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4626.ASM)

```

LP:
clr r31 ;Z 寄存器高位清零 Z 寄存器高位地址为 R31, 低位地址为 R30
ldi r30,$70 ;$70 送 Z 寄存器低位, SRAM 地址  $\geq$  $60
st Z+,r0 ;设(R0)=$11, 先单步执行送 SRAM(Y)=$0070, 然后(Z)+1=$0071
st Z,r1 ;设(R1)=$22, 这时(Z)=$0071, 单步执行后 SRAM($0071)=$22
ldi r30,$80 ;$80 送 Z 寄存器低位
st Z,r2 ;设(R2)=$33, 这时(Z)=$0080, 单步执行后 SRAM($0080)=$33
st -Z,r3 ;设(R3)=$44, 这时先单步执行(Z)-1=$007F, 然后 R3 内容送 SRAM($007F)
std Z+2,r4 ;设(R4)=$55,
;单步执行, 先计算出增量地址(Z+q)=(Z+2)($007F+2)=$0081, 后传送数据 SRAM($0081)=$55,
;但这时 Z 指计内容不变(Z 不含 q)(Z)=$007F
nop ;
rjmp lp ;反复测试

Words: 1 ( 2 bytes)
Cycles: 2
    
```

3.6.3 从程序存储器直接取数据指令

1. LPM—装入程序存储器

说明: 将 Z 寄存器指向的一个字节传送到寄存器 0 (R0)。该指令使 100 %空间有效, 常量初始化或常数取数特别有用。程序存储器被编为 16 位字, Z (16 位) 指针的最低位 (LSB) 选择为 0 是低字节, 选择为 1 是高字节。该指令能寻址程序存储器第一个 64K 字节 (32 字)。

操作: $R0 \leftarrow (Z)$

语法:	操作码:	程序计数器:
LPM	None	$PC \leftarrow PC+1$

16 位操作码:

1001	0101	110X	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4631.ASM,更详细资料阅“按钮猜数.ASM”)

```

.org $0010      ;主程序实际地址为$0020
  clr  r31      ;Z 寄存器高位,存放程序存储器高位地址
  ldi  r30,$F0  ;Z 寄存器低位,存放程序存储器低位地址
  CLR  R29      ;清零 Y 寄存器高位
  LDI  R28,$F0  ;将$F0 装入 Y 寄存器低位
LP: NOP        ;也可设程序存储器($00F0)=$00, 以此类推至设($00FF)=$FF
  IPM          ;将 Z 寄存器低位数送 R0
  ST  Y+,R0    ;Y 变址先将 R0 送 SRAM($00F0),后 Y 地址 Y=(Y+1)
  INC  R30     ;Y 变址将 R0 送 SRAM($00F1),这时 Y=(Y+1)
  CPI  R30,$00 ;R30 内容(Z 寄存器低位)与立即数$00 比较
  BRNE LP     ;R30 内容不为 0 转,为 0 顺执
  INC  R31     ;Z 寄存器高位加 1 ,(Z)=$0100
  RJMP LP     ;反复取数送数,怎样修改程序,使数据存储器与
              ;程序存储器数据大小排列相同,

.ORG $0078    ;实际存放数据地址为$00F0
.DW 0X1122,0X2233,0X4455,0X6677,0X8899,0XAABB,0XCCDD,0XEEFF
.DW $0208,$5510,$1910,$8750,$5012,$8757,$8872,$8757,$8852
    
```

Words: 1 (2 bytes)
Cycles: 3

4.6.4 I/O 口数据传送

1. I/O 口数据传送到寄存器

IN—I/O 口数据传送到寄存器

说明: 将 I/O 空间 (口, 定时器, 配置寄存器等) 的数据传送到寄存器区中的寄存器 Rd 中。

操作: $Rd \leftarrow P$

语法:

操作码:

程序计数器:

$IN\ Rd\ P$ $0 \leq d \leq 31, \quad 0 \leq P \leq 63$

$PC \leftarrow PC + 1$

16 位操作码:

1011	0PPd	dddd	PPPP
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4641.ASM)

LP:

```
IN R25 , $1B    ;A 口的内容送入 R25 中
LDI R23 , $26   ;将 26 送入 R23 中
ADD R23 ,R25    ;R23 与 R25 相加送入 R23
RJMP LP        ;循环
Words:         1 ( 2 bytes)
Cycles:        1
```

2. 寄存器数据送 I / O 口

OUT 寄存器数据送 I / O 口

说明: 将寄存器区中寄存器 Rr 的数据传送到 I / O 空间 (口、定时器、配置寄存器等)。

操作: $P \leftarrow Rr$

语法:

操作码:

程序计数器:

OUT P, Rr $0 \leq r \leq 31, 0 \leq P \leq 63$

$PC \leftarrow PC + 1$

16 位操作码:

1011	1PPr	rrrr	PPPP
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4642.ASM)

;AT90S1200 的 PB 口、PD 口 接 LED 发光二极管,用单步模拟测试 I/O 口工作情况

```
.DEVICE AT90S1200    ;定义被汇编的器件为 AT90S1200
.EQU  PORTB    =0X18
.EQU  DDRB     =0X17    ; B 口数据方向寄存器
.EQU  PORTD    =0X12
.EQU  DDRD     =0X11    ;D 口数据方向寄存器
.ORG  0X0000
LOP:  RJMP  LOP1        ;跳转
.ORG  0X0010
LOP1:  LDI   R20,0XFF    ;硬件设定低电平 LED 灯亮,高电平 LED 灭
      OUT  0X17,R20      ;送 B 口数据方向寄存器
      OUT  0X11,R20      ;送 D 口数据方向寄存器
      CLR  R20           ;清零 LED 灯亮
      OUT  0X18,R20      ;送 B 口数据寄存器
      OUT  0X12,R20      ;送 D 口数据寄存器
      LDI  R20,0X64

      LDI  R20,0XFF      ;关 LED 灯
      OUT  0X18,R20      ;送 B 口 PORTB
      OUT  0X12,R20      ;送 D 口 PORTD
      LDI  R20,0X64
```


RJMP LOP1 ;循环

Words: 1 (2 bytes)
Cycles: 1

4.6.5 堆栈操作指令

AVR 单片机的特殊功能寄存器中有一个堆栈指针 SP。它指出栈顶的位置，在指令系统中有两条用于数据传送的栈操作指令。

1. 进栈指令

PUSH—压寄存器到堆栈

说明：该指令存储寄存器 Rr 的内容到堆栈。

操作：STACK←Rr

语法： 操作码： 程序计数器：
PUSH Rr 0 ≤ d ≤ 31 PC←PC+1 SP←SP- 1
16 位操作码：

1001	001d	dddd	1111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子：(实践操作程序 4651.ASM)

```
LP:
    rcall routine          ;调用 ROUTINE 子程序
routine:   push r13       ;把 R13 压入堆栈
           push r14       ;把 R14 压入堆栈
           push r15       ;把 R15 压入堆栈
           pop  r15        ;从堆栈中取出数据放入 R15
           pop  r14        ;从堆栈中取出数据放入 R14
           pop  r13        ;从堆栈中取出数据放入 R13
           ret             ;返回 R13, R14, R15 的数据
           rjmp lp        ;循环继续做
```

Words: 1 (2 bytes)
Cycles: 2

2. 出栈指令

POP—堆栈弹出到寄存器

说明：该指令将堆栈中的字节装入到寄存器 Rd 中。

操作：Rd←STACK

语法： 操作码： 程序计数器：
POP Rd 0 ≤ d ≤ 31 PC←PC+1 SP←SP+ 1
16 位操作码：

1001	000d	dddd	1111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4652.ASM)

LP:

```

rcall routine          ; 调用子程序 ROUTINE
  routine:
    ldi r16,$01        ; 把立即数 01 送入到 R16 中
    ldi r17,$02        ; 把立即数 02 送入到 R17 中
    push r16           ; 把 R16 压入堆栈内
    push r17           ; 把 R17 压入堆栈内
    pop r17            ; 出栈
    pop r16            ; 出栈
    ret                ; 返回

```

Words: 1 (2 bytes)

Cycles: 2

4.7 位指令和位测试指令

AVR 单片机指令系统中有四分之一的指令为位和位测试指令。位指令的灵活应用，极大地提高了系统的逻辑控制和处理能力。

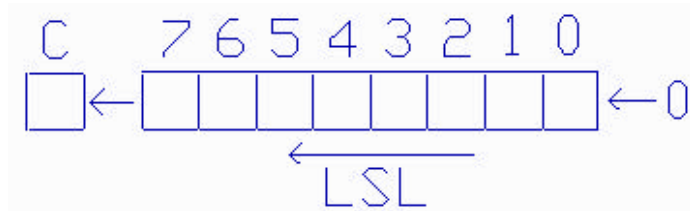
4.7.1 带进位逻辑操作指令

1. 逻辑左移

LSL—逻辑左移

说明：寄存器 Rd 中所有位左移 1 位。第 0 位被清零，第 7 位移到 SREG 中的 C 标志。该指令完成一个无符号数乘 2 的操作。

操作：



语法：

LSL Rd

操作码：

$0 \leq d \leq 31$

程序计数器：

$PC \leftarrow PC + 1$

16 位操作码：

0000	11dd	dddd	dddd
------	------	------	------

P125

状态寄存器 (SREG) 和布尔格式：

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: Rd3

N: R7

S: NV

Z: /R7·/R6·/RS·/R4·/R3·/R2·/R1·/R0

V: NC

C: Rd7

例子：(实践操作程序 4711.ASM)

LP:

ldi r16,\$01 ;将 01 送入到 R16 中

ldi r17,\$02 ;将 02 送入到 R17 中

add r16,r17 ;R16 与 R17 相加

lsl r16 ;加后再左移一位 (01+02=03, 左一位, 即乘以 2。所以 r=06)

rjmp lp ;继续实验

Words: 1 (2 bytes)

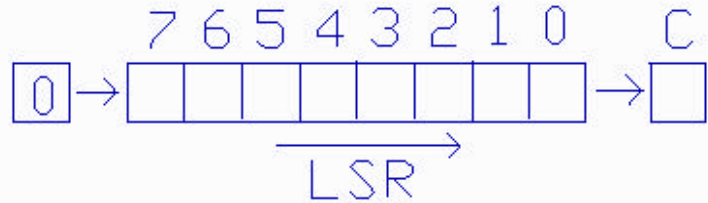
Cycles: 1

2. 逻辑右移

LSR—逻辑右移

说明：寄存器 Rd 中所有位右移 1 位。第 7 位被清零，第 0 位移到 SREG 中的 C 标志。该指令完成一个无符号数除 2 的操作。C 标志被用于结果舍入。

操作：



语法：

LSR Rd

16 位操作码：

操作码：

$0 \leq d \leq 31$

程序计数器：

$PC \leftarrow PC + 1$

1001	010d	dddd	0110
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

I	T	H	S	V	N	Z	C
-	-		↔	↔	↔	↔	↔

S: N V

V: NC

N: 0

例子：(实践操作程序 4712.ASM)

LP:

```
add r0,r1 ;将 R0 与 R1 的内容相加 (r0)= , (r1)=
lsr r0    ;将 R0 的内容右转移一位
rjmp lp   ;循环继续做
```

Words: 1 (2 bytes)

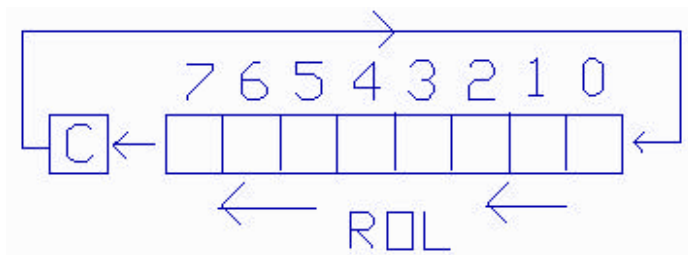
Cycles: 1

3. 通过进位左循环

ROL—通过进位左循环

说明：寄存器 Rd 的所有位左移 1 位，C 标志被移到 Rd 的第 0 位，Rd 的第 7 位移到 C 标志。

操作：



语法: ROL Rd
16 位操作码:
操作码: $0 \leq d \leq 31$
程序计数器: $PC \leftarrow PC + 1$

0001	11dd	dddd	dddd
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: Rd3
S: N V
V: NC
N: R7
Z: /R7·/R6·.R·/R4·/R3·/R2·/R1·R0
C: Rd7

例子: (实践操作程序 4713.ASM)

```
LP:
    rol r15      ;将 R15 中的内容通过进位位循环左移一位, 设: (r15)=
    rol r15      ;通过进位位循环左移一位
    nop         ;
    rjmp lp      ;
```

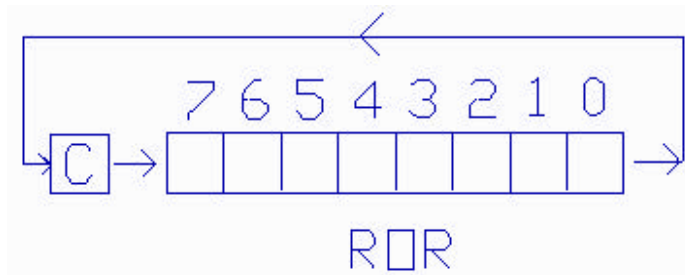
Words: 1 (2 bytes)
Cycles: 1

4. 通过进位右循环

ROR—通过进位右循环

说明: 寄存器 Rd 的所有位右移 1 位, C 标志被移到 Rd 的第 7 位, Rd 的第 0 位移到 C 标志。

操作:



语法: ROR Rd
16 位操作码:
操作码: $0 \leq d \leq 31$
程序计数器: $PC \leftarrow PC + 1$

1001	010d	dddd	0111
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

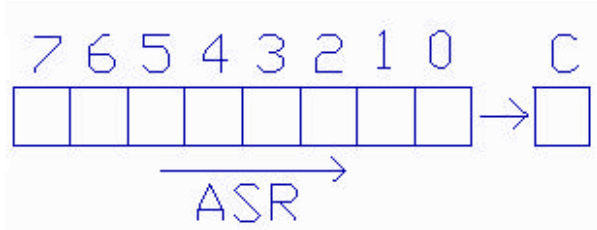
I	T	H	S	V	N	Z	C
-	-		↔	↔	↔	↔	↔

S: N V Z: /R7·/R6·.R·/R4·/R3·/R2·/R1·R0
 V: N C C: Rd0
 N: R7

例子: (实践操作程序 4714.ASM)

```
LP:
    ror r15      ;将 R15 中的内容通过进位循环右移,设:(r15)=
    ror r15
    nop         ;
    rjmp lp     ;
```

Words: 1 (2 bytes)
 Cycles: 1



5. 算术右移

ASR—算术右移

说明: 寄存器 Rd 中的所有位右移 1 位, 而位 7 保持常量, 位 0 被装入 SREG 的 C 标志位。这个操作实现 2 的补码值除 2, 而不改变符号, 进位标志用于结果的舍入。

操作:

语法: 操作码: 程序计数器:
 ASR Rd 0 ≤ d ≤ 31 PC←PC+ 1
 16 位操作码:

1001	010d	dddd	0101
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-		↔	↔	↔	↔	↔

S: N V Z: R7·R6·R5·R4·R3·R2·R1·R0
 V: N C C: Rd0
 N: R7

例子: (实践操作程序 4715.ASM)

```
LP:
    ldi r16,$10 ;
    ldi r17,$fc ;
    asr r16     ;
    asr r17     ;
    rjmp lp     ;
```

words: 1 (2 bytes)
 Cycles: 1

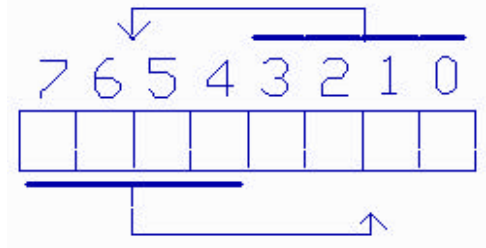
6. 半字节交换

SWAP—半字节交换

说明: 寄存器中的高半字节和低半字节交换。

操作: $R(7\sim4) \leftarrow R_d(3\sim0) \leftarrow R(3\sim0) \leftarrow R_d(7\sim4)$
 语法: 操作码: 程序计数器:
 SWAP Rd $0 \leq d \leq 31$ $PC \leftarrow PC + 1$
 16 位操作码:

1001	010d	dddd	0010
------	------	------	------



状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4716.ASM)

```

LP:
    inc r1    ;
    inc r2    ;
    swap r1   ;
    swap r2   ;
    rjmp lp   ;
    
```

Words: 1 (2 bytes)
 Cycles: 1

4.7.2 位变量传送指令

1. 寄存器中的位存储到 SREG 中的 T 标志

BST—寄存器中的位存储到 SREG 中的 T 标志

说明: 把寄存器中的位 b 存储到 SREG (状态寄存器) 中的 T 标志。

操作: $T \leftarrow R_d(b)$

语法: 操作码: 程序计数器:
 BST Rd b $0 \leq d \leq 31, 0 \leq b \leq 7$ $PC \leftarrow PC + 1$

16 位操作码:

1111	101d	dddd	Xbbb
------	------	------	------

状态寄存器 (SRE) 和布尔格式:

I	T	H	S	V	N	Z	C

T: 0 if bit b in Rd is cleared. Set to 1 otherwise.

例子:

```

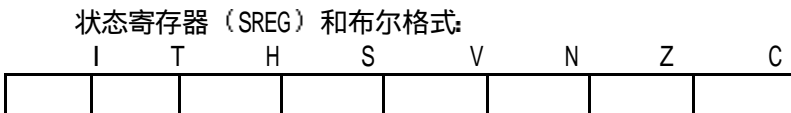
    bst  r1,2 ; T←R1(2)
    bld  r0,4 ; R0(4)←T
    
```

Words: 1 (2 bytes)
 Cycles: 1

2. SREG 中的 T 标志装入寄存器中的某一位

BLD 一位装入：将 SREG 中的 T 标志装入到寄存器中的某一位。
 说明：拷贝 SREG（状态寄存器）的 T 标志到寄存器 Rd 中的位 b。
 操作：Rd (b) ← T
 语法：操作码：程序计数器：
 BLD Rd,d 0 ≤ d ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1
 16 位操作码：

1111	100d	dddd	0bbb
------	------	------	------



例子：(实践操作程序 4722.ASM)
 bst r1,2 ; T ← R1 (2)
 bld r0, 4 ; R0 (4) ← T
 Words: 1 (2 bytes)
 Cycles: 1

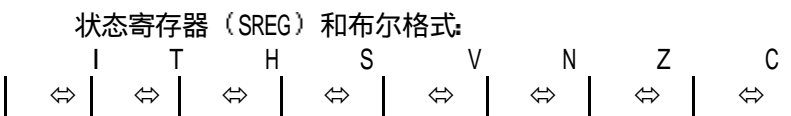
4.7.3 位变量修改指令

1. 置状态寄存器的位

BSET—置状态寄存器的位
 说明：置状态寄存器 (SREG) 的某一标志或某一位。

操作：SREG (S) ← 1
 语法：操作码：程序计数器：
 BSET s 0 ≤ s ≤ 7 PC ← PC + 1
 16 位操作码：

1001	0100	0sss	1000
------	------	------	------



I: 1 if s=7 V: 1 if s=3
 T: 1 if s=6 N: 1 if s=3
 H: 1 if s=5 Z: 1 if s=1
 S: 1 if s=4 C: 1 if s=0

例子：
 bset 6 ; SREG (6) ← 1
 bset 7 ; SREG (7) ← 1
 Words: 1 (2 bytes)
 Cycles: 1

2. 清状态寄存器的位

BCLR——SREG 中的位清零

说明：清零 SREG 状态寄存器中的一个标志位。

操作：SREG (S) ← 0

语法：操作码：程序计数器：
 BCLR s 0 ≤ s ≤ 7 PC ← PC + 1
 16 位操作码：

1001	0100	1sss	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
↔	↔	↔	↔	↔	↔	↔	↔

I: 0 if s=7 V: 0 if s=3
 T: 0 if s=6 N: 0 if s=2
 H: 0 if s=5 Z: 0 if s=1
 S: 0 if s=4 C: 0 if s=0

例子:

```
bclr 0 ; SREG (0) ← 0
bclr 7 ; SREG (7) ← 0
```

Words: 1 (2 bytes)

Cycles: 1

3. 置 I/O 寄存器的位

SBI——置 I/O 寄存器的位

说明：对 I/O 寄存器指定的位置位，该指令在低 32 个 I/O 寄存器内操作，I/O 寄存器地址为 0~31。

操作：I/O (P b) ← 1
 语法：操作码：程序计数器：
 SBIP, b 0 ≤ P ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1
 16 位操作码：

1001	1010	PPPP	Pbbb
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子:

```
out $1e,r0 ; (EEARL 寄存器) ← (R0)
sbi $1c,0 ; (EECR 寄存器 0 位) ← 1
in r1,$1d ; (R0) ← (EEDR 寄存器)
```

Words: 1 (2 bytes)

Cycles: 2

4. 清 I/O 寄存器的位

CBI—清 I/O 寄存器的位

说明：清零 I/O 寄存器中的指定位，该指令用在寄存器最低的 32 个 I/O 寄存器上，I/O 寄存器地址为 0~31。

操作：I/O (P, b) ← 0

语法：

CBI P, b

操作码：

$0 \leq P \leq 31, 0 \leq b \leq 7$

程序计数器：

PC ← PC + 1

16 位操作码：

1001	1000	PPPP	Pbbb
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

I	T	H	S	V	N	Z	C

C: 1

例子：

cbi \$18,7 ; I/O (PORTB 寄存器的 7 位) ← 0

Words: 1 (2 bytes)

Cycles: 2

5. 置进位位

SEC —置位进位标志

说明：置位 SREG (状态寄存器) 中的进位标志 (C)。

操作：C ← 1

语法：

SEC

操作码：

None

程序计数器：

PC ← PC + 1

16 位操作码：

1001	0100	0000	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式：

I	T	H	S	V	N	Z	C
							1

C: 1

例子：

sec ; C ← 1

adc r0, r1 ;带进位位加

words: 1 (2 bytes)

Cycles: 1

6. 清进位位

CLC —清零进位标志

说明：清零 SREG (状态寄存器) 中的进位标志 (C)。

操作：C ← 0

语法：

CLC

操作码：

None

程序计数器：

PC ← PC + 1

16 位操作码:

1001	0100	1000	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
							0

C: 0

例子:

```

add r0, r0 ;加
clc      ; C←0
Words:  1 ( 2 bytes)
Cycles:  1

```

7. 置位负标志位

SEN—置位负数标志

说明: 置位 SREG (状态寄存器) 中的负数标志 (N)。

操作: $N \leftarrow 1$

语法	操作码:	程序计数器:
SEN	None	$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	0010	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
					1		

N: 1

例子:

```

add r2, r19 ;加
sen      ; N←1
Words:  1 ( 2 bytes)
Cycles:  1

```

8. 清负标志位

CLN—清零负数标志

说明: 清零 SREG (状态寄存器) 中的负数标志 (N)。

操作: $N \leftarrow 0$

语法:	操作码:	程序计数器:
CLN	None	$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	1010	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
					0		

N: 0

例子:

```

    add r2, r3 ;加
    cln      ; N←0
Words:    1 ( 2 bytes);
Cycles:    1

```

9. 置零标志位

SEZ—置位零标志

说明: 置位 SREG (状态寄存器) 中的零标志 (Z)。

操作: $Z \leftarrow 1$

语法:

SEZ

操作码:

None

程序计数器:

$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	0001	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
						1	

Z: 1

例子:

```

    add r2, r19 ;加
    sez      ; Z←1
Words:    1 ← (2 bytes)
Cycles:    1

```

10. 清零标志位

CLZ—清零零标志

说明: 清零 SREG (状态寄存器) 中的零标志 (Z)。

操作: $Z \leftarrow 0$

语法:

CLZ

操作码:

None

程序计数器:

$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	1001	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
						0	

Z: 0

例子:

```
add r2, r3 ;加
clz      ; Z←0
```

Words: 1 (2 bytes)

Cycles: 1

11. 触发全局中断位

SEI—置位全局中断标志

说明: 置位 SREG (状态寄存器) 中的全局中断标志 (I)。

操作: $I \leftarrow 1$

语法:

操作码:

程序计数器:

SEI

None

$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	0111	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
1							

I: 1

例子:

```
cli      ; I←0
in r13,$16 ; (r13)←(PINB 寄存器数据)
set     ; I←1
```

Words: 1 (2 bytes)

Cycles: 1

12. 禁止全局中断位

CLI—清零全局中断标志

说明: 清除 SREG (状态寄存器) 中的全局中断标志 (I)。

操作: $I \leftarrow 0$

语法:

操作码:

程序计数器:

CLI

None

$PC \leftarrow PC + 1$

16 位操作码:

1001	0100	1111	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
0							

I: 0

例子: (实践操作程序 47312.ASM)

```
cli          ; I←0
in r13,$16  ;(r13)←(PINB 寄存器数据)
set         ; I←1
```

Words: 1 (2 bytes)

Cycles: 1

13. 置 S 标志位

SES—置位符号标志

说明: 置位 SREG (状态寄存器) 中的符号标志 (S)。

操作: $s \leftarrow 1$

语法: 操作码: 程序计数器:
SES None $PC \leftarrow PC + 1$

16 位操作码:

1001	0100	0100	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
			1				

S: 1

例子: (实践操作程序 47313.ASM)

```
add r2, r19 ;加
ses         ; s←1
```

Words: 1 (2 bytes)

Cycles: 1

14. 清 S 标志位

CLS—清零符号标志

说明: 清零 SREG (状态寄存器) 中的符号标志 (S)。

操作: $S \leftarrow 0$

语法: 操作码: 程序计数器:
CLS None $PC \leftarrow PC + 1$

16 位操作码:

1001	0100	1100	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
			0				

S: 0

例子:

```
add r2, r3 ;加
```

Words: 1 (2 bytes)
 Cycles: 1

15. 置溢出标志位

SEV——置位溢出标志

说明: 置位 SREG (状态寄存器) 中的溢出标志 (V)。

操作: $V \leftarrow 1$

语法:

SEV

16 位操作码:

操作码:

None

程序计数器:

$PC \leftarrow PC + 1$

1001	0100	0011	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
				1			

V: 1

例子: (实践操作程序 47315.ASM)

add r2, r19 ;加

sev ; $V \leftarrow 1$

Words: 1 (2 bytes)

Cycles: 1

16. 清溢出标志位

CLV——清零溢出标志

说明: 清零 SREG (状态寄存器) 中的溢出标志 (V)。

操作: $V \leftarrow 0$

语法:

CLV

16 位操作码:

操作码:

None

程序计数器:

$PC \leftarrow PC + 1$

1001	0100	1011	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
				0			

V: 0

例子:

add r2, r3 ;加

clv ; $V \leftarrow 0$

Words: 1 (2 bytes)

Cycles: 1

17. 置 T 标志位

SET 置位 T 标志

说明: 置位 SREG (状态寄存器) 中的 T 标志。

操作: $T \leftarrow 1$

语法:

操作码:

程序计数器:

SET

None

 $PC \leftarrow PC + 1$

16 位操作码:

1001	0100	0110	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
	1						

T: 1

例子:

set ; $T \leftarrow 1$

Words: 1 (2 bytes)

Cycles: 1

18. 清 T 标志位

CLT—清零 T 标志

说明: 清零 SREG (状态寄存器) 中的 T 标志。

操作: $T \leftarrow 0$

语法:

操作码:

程序计数器:

CLT

None

 $PC \leftarrow PC + 1$

16 位操作码:

1001	0100	1110	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
	0						

T: 0

例子:

clt ; $T \leftarrow 0$

Words: 1 (2 bytes)

Cycles: 1

19. 置半进位标志

SEH—置位半进位标志

说明: 置位 SREG (状态寄存器) 中的半进位标志 (H)。

操作: $H \leftarrow 1$

语法:

操作码:

程序计数器:

SEH None PC←PC+ 1
16 位操作码:

1001	0100	0101	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
		1					

H: 1

例子:

seh ; H←1

Words: 1 (2 bytes)

Cycles: 1

20. 清半进位标志

CLH—清零半进位标志

说明: 清零 SREG (状态寄存器) 中的半进位标志 (H)。

操作: H←0

语法: 操作码: 程序计数器:
CLH None PC←PC+ 1

16 位操作码:

1001	0100	1101	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
		0					

H: 0

例子:

clh ; H←0

Words: 1 (2 bytes)

Cycles: 1

4.7.4 其它指令

1. 空指令

NOP—空操作

说明: 该指令完成一个单周期空操作。

应用: 延时等待;产生方波;抗干扰,在无程序单元写上空操作,空操作指令最后转到\$000H

操作: No

语法: 操作码: 程序计数器:
NOP None PC←PC+ 1

16 位操作码:

0000	0000	0000	0000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4741.ASM)

LP:

```

nop          ;空操作
add r0 ,r1  ;R1 与 R0 相加
nop          ;空操作
add r0, r1  ;R1 与 R0 相加
rjmp lp     ;循环 P138
Words:      1 (2 bytes)
Cycles:     1

```

2. 休眠指令

SLEEP—休眠

说明: 该指令设置电路休眠模式, 由 MCU 控制寄存器定义。当在休眠状态由一个中断唤醒时, 在中断程序执行后, 紧跟在休眠指令后的指令被执行。

应用: 省电, 尤其对便携式仪器特别有用

操作:

语法:	操作码:	程序计数器:
SLEEP	None	PC+PC+ 1
16 位操作码:		

1001	0101	100x	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4742.ASM)

```

mov ro, rll ;拷贝
sleep      ;休眠
words:     1 (2 bytes)
Cycles:    1

```

3. 看门狗复位

WDR—看门狗复位

说明: 该指令复位看门狗定时器, 在 WD 预定比例器给出限定时间内必须执行。参见看门狗定时器硬件部分。

应用: 抗干扰; 延时

操作: WD timer restart.

语法:

WDR

16 位操作码:

操作码:

None

程序计数器:

PC~PC+ 1

1001	0101	101x	1000
------	------	------	------

状态寄存器 (SREG) 和布尔格式:

I	T	H	S	V	N	Z	C

例子: (实践操作程序 4743.ASM)

PLYDEL: LDI TEMP,185 ;用 WDR 做延时子程序

DT3: LDI TEMP1,04 ;

DT2: LDI TEMP2,250

DT1: WDR ;1T

WDR ;2T

WDR ;3T

WDR ;4T

WDR ;5T

DEC TEMP2 ;

BRNE DT1 ;

DEC TEMP1 ;

BRNE DT2 ;

DEC TEMP ;

BRNE DT3 ;

RET

Words: 1 (2 bytes)

Cycles: 1

4.8 新增指令

到目前为止,只有 ATmega161 单片机有 130 条指令。以下指令仅适用 ATmega161 单片机。

3.8.1 EICALL - 延长间接调用子程序

说明:间接调用由寄存器文件中的 Z(16位) 指针和 I/O 端口中的 EIND 寄存器指向的子程序。该指令允许调用整个程序存储器空间内的子程序。该指令在双字节 PC 的设备中是无效的,见 ICALL。在 EICALL 指令执行期间堆栈指针使用一种后进先出的设置。

操作:

(i) $PC(15:0) \leftarrow Z(15:0)$
 $PC(21:16) \leftarrow EIND$

语法:	操作码:	程序计数器:	堆栈:
(i) EICALL	None	See Operation	STACK \leftarrow PC + 1 SP \leftarrow SP - 3 (3 bytes, 22 bits)

16位操作码:

1001	0101	0001	1001
------	------	------	------

状态寄存器(SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

例子: (实践操作程序481.ASM)

```
ldi r16,$05          ; 设置EIND和z指针
out EIND,r16
ldi r30,$00
ldi r31,$10
eicall               ; 调用$051000
```

Words: 1 (2 bytes)

Cycles: 4 (仅在带22位PC的器件上执行)

4.8.2 EIJMP - 扩展间接跳转

说明:间接跳转到由寄存器文件中的 Z (16 位) 指针和 I/O 端口中的 EIND 寄存器指向的地址。该指令允许间接跳转到整个程序存储器空间。

操作:

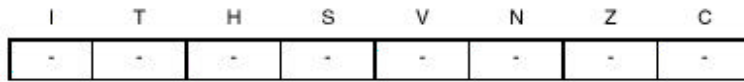
(i) $PC(15:0) \leftarrow Z(15:0)$
 $PC(21:16) \leftarrow EIND$

语法:	操作码:	程序计数器:	堆栈:
(i) EIJMP	None	See Operation	不影响

16位操作码:

1001	0100	0001	1001
------	------	------	------

状态寄存器(SREG) 和布尔格式:



例子: (实践操作程序482.ASM)

```

ldi r16,$05          ; 设置EIND和Z指针
out EIND,r16
ldi r30,$00
ldi r31,$10
eijmp                ; 跳转到$051000
    
```

Words: 1 (2 bytes)
Cycles: 2

4.8.3 ELPM -扩展装载程序存储器

说明:装入由Z寄存器和I/O端口中的RAMPZ寄存器指向的一个字节,将这一字节装入目的寄存器Rd。该指令描述一个百分之百空间有效的常量初始化或常量数据引出。程序存储器是按16位字组织起来,Z寄存器最重要的位选择低字节(0)或高字节(1)。该指令能寻址整个程序存储器空间。该操作不改变Z指针寄存器,或使之增加。增加适用于RAMPZ和Z指针寄存器的整个24位串联。这些合并的结果是不确定的:

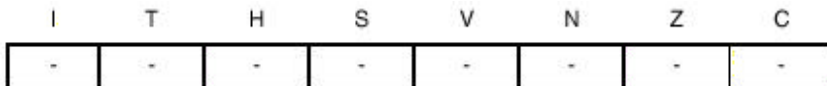
	ELPM r30, Z+ ELPM r31, Z+	
操作:		注释:
(i)	R0 ← (RAMPZ:Z)	RAMPZ:Z: Unchanged, R0 implied destination register
(ii)	Rd ← (RAMPZ:Z)	RAMPZ:Z: Unchanged
(iii)	Rd ← (RAMPZ:Z) (RAMPZ:Z) ← (RAMPZ:Z) + 1	RAMPZ:Z: Post incremented

	语法:	操作码:	程序计数器:
(i)	ELPM	None, R0 implied	PC ← PC + 1
(ii)	ELPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	ELPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

16位操作码:

(i)	1001	0101	1101	1000
(ii)	1001	000d	d d d d	0110
(iii)	1001	000d	d d d d	0111

状态寄存器(SREG) 和布尔格式:



例子: (实践操作程序483.ASM)

```

clr r16              ; RAMPZ寄存器清零
out RAMPZ, r16
clr r31              ; 清除Z指针寄存器的高字节
    
```

```
ldi r30,$F0      ; z指针寄存器的低字节置1
elpm r16, Z+     ; 从程序装入常数
                  ; RAMPZ:Z (r31:r30)指向的存储器
```

Words: 1 (2 bytes)

Cycles: 3

4.8.4 ESPM - 扩展存储程序存储器

说明:ESPM 指令用来擦除程序存储器中的一页, 写程序存储器中的一页(那是已经被擦除的), 设置引导装载器锁位。在一些器件中, 程序存储器是一次写一字节, 另一些器件中在先填充一个暂时页缓冲器后能同时编程一整页。就一切情况而论, 程序存储器必须一次被擦除一页。擦除程序存储器时, RAMPZ和Z寄存器被用来页寻址。写程序存储器时, RAMPZ和Z寄存器被用作页或字寻址, R1:R0 寄存器组被当作数据。设置引导装载锁位时, R1:R0 寄存器组被当作数据。参考器件文件以获得ESPM 用法的详细说明。该指令可寻址整个程序存储器。

操作:		注释:
(i)	(RAMPZ:Z) ← \$ffff	擦除程序存储器页
(ii)	(RAMPZ:Z) ← R1:R0	写程序存储器字
(iii)	(RAMPZ:Z) ← R1:R0	写暂时页缓冲器
(iv)	(RAMPZ:Z) ← TEMP	写暂时页缓冲器到程序存储器
(v)	BLBITS ← R1:R0	设置引导装载器锁位
	语法: 操作码:	程序计数器:
(i)-(v)	ESPM None	PC ← PC + 1
	16位操作码:	

1001	0101	1111	1000
------	------	------	------

状态寄存器(SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example: (实践操作程序484.ASM)

```
clr r31          ; 此例说明了对于带页写的器件的字ESPM写
                 ; z寄存器的高字节清零
clr r30          ; z寄存器的低字节清零
ldi r16,$F0     ; 装入RAMPZ寄存器
out RAMPZ, r16 ;
ldi r16,$CF     ; 存入数据
mov r1, r16
ldi r16,$FF
mov r0, r16
ldi r16,$03     ; ESPM使能, 擦除页
out SPMCR, r16 ;
espm           ; 从$F00000开始擦除页
ldi r16,$01     ; ESPM使能, 将R1:R0存入暂时缓冲器
out SPMCR, r16 ;
espm           ; 执行ESPM, 将R1:R0存入暂时缓冲器的$F00000处
ldi r16,$05     ; ESPM使能, 写页
out SPMCR, r16 ;
```

espm ; 执行ESPM,将暂时缓冲器存入从程序存储器地址\$F00000开始的页

Words: 1 (2 bytes)
Cycles: 依操作而定

4.8.5 FMUL - 小数乘法

说明: 该指令完成8位×8位→16位的无符号数乘法操作并把结果左移一位。



(N.Q) 表示一个小数点左边有N个二进制数位、小数点右边有Q个二进制数位的小数。以(N1.Q1)和(N2.Q2)为格式的两个数相乘产生格式为((N1+N2).(Q1+Q2))的结果。为处理符号,以(1.7)格式作输入,产生(2.14)格式的结果。结果的高字节要左移一位以使结果的格式与输入的一致。FMUL指令在与MUL指令相同的周期内合并了左移操作。

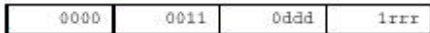
被乘数Rd和乘数Rr是两个包含无符号小数的寄存器,固定的小数位在第6位和第7位之间。16位无符号小数结果的固定小数位在第14位和第15位之间,即存放在R1(高字节)和R0(低字节)。

操作:
(i) $R1:R0 \leftarrow Rd \times Rr$ (unsigned (1.15) \leftarrow unsigned (1.7) \times unsigned (1.7))

语法: 操作码: 程序计数器:

(ii) FMUL Rd,Rr $16 \leq d \leq 23, 16 \leq r \leq 23$ $PC \leftarrow PC + 1$

16位操作码:



状态寄存器(SREG)和布尔格式:



- C: R16
如果结果的第15位在左移前被置1则置1,否则清除。
- Z: $R15 \bullet R14 \bullet R13 \bullet R12 \bullet R11 \bullet R10 \bullet R9 \bullet R8 \bullet R7 \bullet R6 \bullet R5 \bullet R4 \bullet R3 \bullet R2 \bullet R1 \bullet R0$
如果结果是\$0000则置1,否则清除。
- R (结果)操作后等于R1,R0。

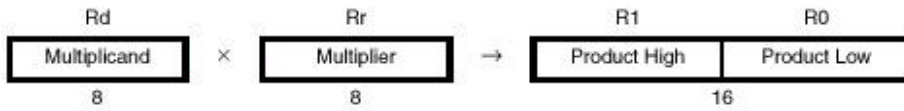
例子: (实践操作程序485.ASM)

```
fmul r23,r22 ; 无符号数r23和r22以(1.7)格式相乘,产生(1.15)格式的结果
movw r22,r0 ; 复制结果回r23:r22
```

Words: 1 (2 bytes)
Cycles: 2

4.8.6 FMULS -有符号数乘法

说明: 该指令完成8位×8位→16位的有符号数乘法操作并把结果左移一位。



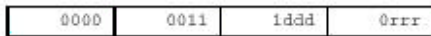
(N.Q) 表示一个小数点左边有N个二进制数位、小数点右边有Q个二进制数位的小数。以(N1.Q1)和(N2.Q2)为格式的两个数相乘产生格式为((N1+N2).(Q1+Q2))的结果。为处理符号, 以(1.7) 格式作输入, 产生(2.14) 格式的结果。结果的高字节要左移一位以使结果的格式与输入的一致。FMULS 指令在与MULS 指令相同的周期内合并了左移操作。

被乘数Rd和乘数Rr是两个包含有符号小数的寄存器, 固定的小数位在第6位和第7位之间。16位有符号小数结果的固定小数位在第14位和第15位之间, 即存放在R1(高字节)和R0(低字节)。

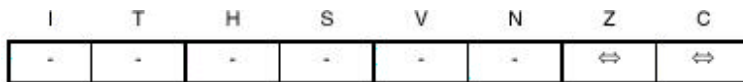
操作:

- (i) R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × signed (1.7))
 语法: 操作码: 程序计数器:
 (ii) FMUL Rd,Rr 16 ≤ d ≤ 23, 16 ≤ r ≤ 23 PC ← PC + 1

16位操作码:



状态寄存器(SREG) 和布尔格式:



- C: R16
如果结果的第15 位在左移前被置1则置1, 否则清除。
- Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0
如果结果是\$0000则置1, 否则清除。
- R(结果)操作后等于R1,R0。

例子: (实践操作程序486.ASM)

fmuls r23,r22 ; 有符号数r23和r22以(1.7)格式相乘,产生(1.15)格式的结果

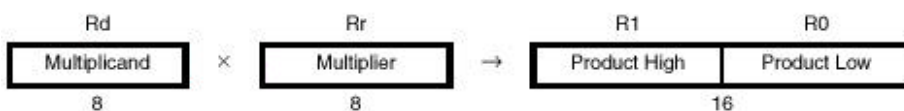
movw r22,r0碍 ; 复制结果回r23:r22

Words: 1 (2 bytes)

Cycles: 2

4.8.7 FMULSU - 有符号小数和无符号小数乘法

说明: 该指令完成8位×8位→16位的有符号数乘法操作并把结果左移一位。



(N.Q) 表示一个小数点左边有N个二进制数位、小数点右边有Q个二进制数位的小数。以(N1.Q1)和(N2.Q2)为格式的两个数相乘产生格式为((N1+N2).(Q1+Q2))的结果。为处理符号, 以(1.7) 格式作输入, 产生(2.14) 格式的结果。结果的高字节要左移一位以使结果的格式与输入的一致。

FMULSU指令在与MULSU指令相同的周期内合并了左移操作。被乘数Rd和乘数Rr是两个包含小数的寄存器，暗含的小数位在第6位和第7位之间。被乘数Rd是一个有符号小数，乘数Rr是一个无符号小数。16位有符号小数结果暗含的小数位在第14位和第15位之间，即存放在R1(高字节)和R0(低字节)。

操作:

(i) $R1:R0 \leftarrow Rd \times Rr$ (signed (1.15) \leftarrow signed (1.7) \times unsigned (1.7))

语法: 操作码: 程序计数器:

(ii) FMULSU Rd,Rr $16 \leq d \leq 23, 16 \leq r \leq 23$ $PC \leftarrow PC + 1$

16位操作码:

0000	0011	1ddd	1rrr
------	------	------	------

状态寄存器(SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	↔	↔

C: R16

如果结果的第15位在左移前被置1则置1，否则清除。

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0

如果结果是\$0000则置1，否则清除。

R (结果)操作后等于R1,R0。

例子: (实践操作程序487.ASM)

fmulSU r23,r22 ; 有符号数r23和无符号数r22以(1.7)格式相乘,产生(1.15)格式的结果

结果

movw r22,r0 ; 复制结果回r23:r22

Words: 1 (2 bytes)

Cycles: 2

4.8.8 MOVW -拷贝寄存器字

说明: 该指令完成将一个寄存器组拷贝到另一个寄存器组的操作。源寄存器组Rr+1:Rr 不改变,目的寄存器组Rd+1:Rd则是 Rr + 1:Rr所含内容的拷贝。

操作:

(i) $Rd+1:Rd \leftarrow Rr+1:Rr$

语法: 操作码: 程序计数器:

16位操作码:

(i) MOVW Rd,Rr $d \in \{0,2,\dots,30\}, r \in \{0,2,\dots,30\}$ $PC \leftarrow PC + 1$

0000	0001	ddd	rrr
------	------	-----	-----

状态寄存器(SREG) 和布尔格式:

例子: (实践操作程序488.ASM)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

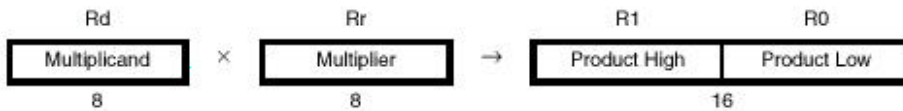
```

        movw r16,r0      ; 拷贝 r1:r0 到 r17:r16
        call check      ; 调用子程序
        ...
check:  cpi r16,$11      ; r16 - $11
        ...
        cpi r17,$32      ; r17 - $32
        ...
        ret              ; 子程序返回
    
```

Words: 1 (2 bytes)
Cycles: 1

4.8.9 MULS -有符号数乘法

说明: 该指令完成8位×8位→16位有符号数乘法的操作。



被乘数Rd和乘数Rr是两个包含有符号数的寄存器。16位有符号结果存放在R1 (高字节) 和 R0 (低字节)中。

操作:

(i) $R1:R0 \leftarrow Rd \times Rr$ (signed \leftarrow signed \times signed)

语法: 操作码: 程序计数器:

(ii) $MULS\ Rd,Rr$ $16 \leq d \leq 31, 16 \leq r \leq 31$ $PC \leftarrow PC + 1$

16位操作码:

0000	0010	dddd	rrrr
------	------	------	------

状态寄存器(SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	↔	↔

- C: R15
如果结果的第15 位被置1则置1, 否则清除。
- Z: $R15 \bullet R14 \bullet R13 \bullet R12 \bullet R11 \bullet R10 \bullet R9 \bullet R8 \bullet R7 \bullet R6 \bullet R5 \bullet R4 \bullet R3 \bullet R2 \bullet R1 \bullet R0$
如果结果是\$0000则置1, 否则清除。
- R (结果)操作后等于R1,R0。

例子: (实践操作程序489.ASM)

```

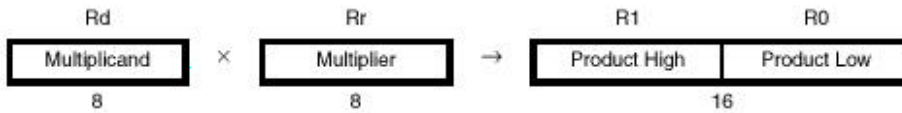
        muls r21,r20      ; 有符号数r21和r20相乘
    
```

`movw r20,r0 ; 拷贝结果回 r21:r20`

Words: 1 (2 bytes)
Cycles: 2

4.8.10 MULSU - 有符号数与无符号数乘法

说明: 该指令完成8位×8位→16位的一个有符号数和一个无符号数乘法的操作。



被乘数Rd和乘数Rr是两个寄存器。被乘数Rd是有符号数，乘数Rr是无符号数。16位有符号结果存放在R1 (高字节) 和 R0 (低字节)中。

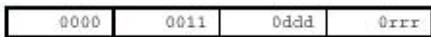
操作:

- (i) $R1:R0 \leftarrow Rd \times Rr$ (signed \leftarrow signed \times signed)

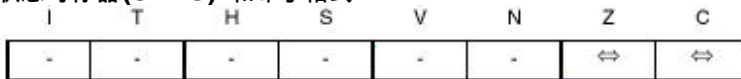
语法: 操作码: 程序计数器:

- (i) `MULSU Rd,Rr` $16 \leq d \leq 23, 16 \leq r \leq 23$ $PC \leftarrow PC + 1$

16位操作码:



状态寄存器(SREG) 和布尔格式:



C: R15

如果结果的第15 位被置1则置1，否则清除。

Z: R15 •R14 •R13 •R12 •R11 •R10 •R9 •R8 •R7 R6 R5 R4 R3 R2 •R1 R0

如果结果是\$0000则置1，否则清除。

R (结果)操作后等于R1,R0。

例子: (实践操作程序4810.ASM)

`mulsu r21,r20 ; 有符号数r21和无符号数r20相乘得有符号数结果。`
`movw r20,r0 ; 拷贝结果回 r21:r20`

Words: 1 (2 bytes)
Cycles: 2

4.8.11 SPM -存储程序存储器

说明:SPM 指令可用来擦除程序存储器的一页，写程序存储器中的一页（那是已经被擦除的），设置引导装载机锁位。在一些器件中，程序存储器是一次写一字节，另一些器件中在先填充一个暂时页缓冲器后能同时编程一整页。就一切情况而论，程序存储器必须一次被擦除一页。擦除程序存储器时，Z寄存器被用来页寻址。写程序存储器时，Z寄存器被用作页或字寻址，R1:R0 寄存器

组被当作数据。设置引导装载器锁位时，R1:R0 寄存器组被当作数据。参考器件文件以获得SPM用法的详细说明。该指令可寻址程序存储器的前64K字节（32K字）。

操作:	注释:
(i) (Z)←\$ffff	擦除程序存储器页
(ii) (Z)←R1:R0	写程序存储器字
(iii) (Z)←R1:R0	写暂时页缓冲器
(iv) (Z)←TEMP	写暂时页缓冲器到程序存储器
(v) BLBITS←R1:R0	设置引导装载器锁位

语法: 操作码: 程序计数器:

16位操作码:

1001	0101	1110	1000
------	------	------	------

状态寄存器(SREG) 和布尔格式:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

例子: (实践操作程序4811.ASM)

```

; 此例显示对带字写的器件的SPM字写
ldi r31, $F0 ; 装入 z 的高字节
clr r30 ; 清除 z 的低字节
ldi r16, $CF ; 装入数据以便存储
mov r1, r16
ldi r16, $FF
mov r0, r16
ldi r16,$03 ; SPM使能, 擦除页
out SPMCR, r16 ;
spm ; 从$F000开始擦除页
ldi r16,$01 ; SPM使能, 存储到存储器
out SPMCR, r16 ;
spm ; 执行SPM操作, 存储R1:R0 到存储器的$F000单元。

```

Words: 1 (2 bytes)

Cycles: 依据操作而定