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 直接数据传进指令

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：	操作码：	程序计数器：
MOV Rd Rr	0≤ d≤ 31，0≤ r≤31	PC←PC＋ 1

16 位操作码：

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：	操作码：	程序计数器：
LDS Rd k	0≤ d≤ 31，0≤ k≤65535	PC←PC＋ 2

32 位操作码：

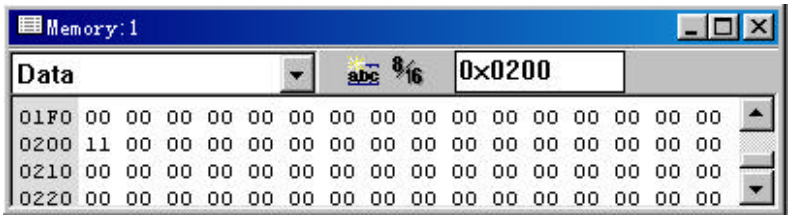
1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

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 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：	操作码：	程序计数器：
STS k，Rr	0≤r≤31，0≤k≤65535	PC←PC＋2

32 位操作码：

1001	001d	dddd	0000
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 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←K

语法：	操作码：	程序计数器：
LDI Rd K	16≤ d≤ 31， 0≤K≤ 255	PC←PC＋ 2

16 位操作码：

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≤ R ≤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 一使用变址 X 间接将 SRAM 中内容送入到寄存器

说明：从 SRAM 中间接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←（X）                    ；送数,X 指针寄存器值不改变
      Rd←（X）        X←X＋1      ;先送数,后 X 指针寄存器值加 1
      X←X－1          Rd←(X)      ;先 X 指针寄存器值减 1，后送数
```

语法：	操作码：	操作流程	程序计数器：
LD Rd, X	0≤ d≤31	送数,X 指针不改变	PC←PC＋1
LD Rd,X+	0≤d≤31	先送数,后 X 指针加 1	PC←PC＋1
LD Rd,-X	0≤d≤31	先 X 指针减 1，后送数	PC←PC＋l

16 位操作码：

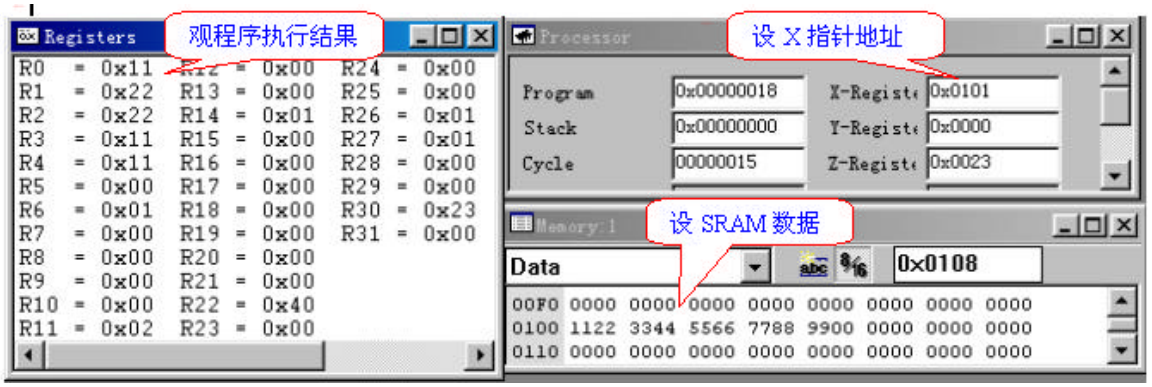
1.	1001	000d	dddd	1100
2.	1001	000d	dddd	1101
3.	1001	000d	dddd	1110

状态寄存器（SREG）和布尔格式：

I	T	H	S	V	N	Z	C
—	—	—	—	—	—	—	—

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

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）—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+l
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）← Rr

（Y）← Rr　　Y←Y＋1

Y ← Y—l　　（Y）←Rr

(Y＋q)← Rr

语法:	操作码:	操作流程:	程序计数器:
ST Y,Rr	0≤ d≤31	送数,Y 指针不改变	PC←PC＋1
ST Y＋,Rr	0≤d≤31	先送数,后 Y 指针加 1	PC←PC＋1
ST -Y，Rr	0≤d≤31	先 Y 指针减 1，后送数	PC←PC＋l
STD Y＋q,Rr	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	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 指计内容不变(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←（Z）
Rd←（Z）
Z ← Z—1
Rd← （Z+q）

语法:	操作码:	操作流程:	程序计数器:
LD Rd,Z	0≤ d≤31	送数,Z 指针不改变	PC←PC+1
LD Rd Z+	0≤d≤31	先送数,后 Z 指针加 1	PC←PC+1

LD Rd,-Z 0≤d≤31 先 Z 指针减 1，后送数 PC←PC＋l
LDD Rd Z＋q 0≤ d≤31, 先 Z 指针加 q,后送数, PC←PC＋1
0≤q≤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≥$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，特别适合用作堆栈指针。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）←Rr
(Z）←Rr Z←Z＋1
Z← Z-1 （Z）←Rr

（Z+q）←Rr

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

语法:	操作码:	程序计数器:
LPM	None	PC←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($000F0)=$00,以此类推至设($00FF)=$FF
  lPM             ;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←P

语法：	操作码：	程序计数器：
IN Rd P	0≤ d≤31， 0≤ P≤63	PC←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← Rr

语法:	操作码:	程序计数器:
OUT P, Rr	0≤ r≤31, 0≤ P≤63	PC←PC 十 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: l（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
```