第3章

MCS-51单片机的指令系统

第2章介绍了MCS-51系列单片机的内部构造和工作原理，程序存储器和数据存储器的配置，I/O口的结构和功能等；要在单片机提供的硬件平台上实现人们期望的功能，还必须有软件(程序)的支持。计算机之所以能够完成给定的任务，是因为事先已把编制好的程序存入程序存储器中，通过程序指挥CPU完成规定的运算和控制任务。编制程序是为了实现某种意图而把一系列指令按照一定规则组织起来。指令是程序的基本单元，它是控制、指挥CPU的命令，指令是由计算机的硬件特性决定的，不同类型的计算机的指令系统是不兼容的。一台计算机所有指令的集合称为指令系统。指令系统展示了计算机的操作功能，它是表征计算机性能的一个重要指标。MCS-51单片机指令系统共有111条指令，提供了多种快速灵活的寻址方式，指令代码短、功能强、执行快；另外，指令系统把位变量作为一种数据类型，提供了位操作指令，允许直接对位进行逻辑和传送操作，极大地方便了在控制系统中对逻辑变量的布尔处理。本章将介绍MCS-51单片机指令系统的格式、MCS-51单片机寻址方式以及指令分类、功能和使用方法。

3.1 指令格式

指令(Instruction)是人们给计算机的命令，是芯片制造厂家提供给用户使用的软件资源。一台计算机所有指令的集合称为指令系统。由于计算机只能识别二进制数和二进制编码，而对于用户来说，二进制编码可读性差，难以记忆和理解，因此，一条指令有两种表示方式：一种是计算机能够识别的机器码(二进制编码的指令代码)——机器语言(Machine Language)；另一种是采用人们容易理解和记忆的助记符形式——汇编语言(Assembly Language)。汇编语言便于用户编写、阅读和识别程序，但不能直接被计算机识别和理解，必须汇编成机器语言才能被计算机识别和执行。汇编语言可以汇编为机器语言，机器语言也可反汇编为汇编语言，它们之间一一对应。汇编和反汇编可以由编译系统自动完成，也可以由用户通过人工查表的方法手工完成。本章主要介绍指令的汇编语言形式。

MCS-51单片机的指令由标号、操作码、操作数和注释4个部分组成，格式如下：

[标号：]操作码助记符[操作数]；[注释]

(1) 标号：标号表示该指令代码的第一个字节所在单元地址。由用户自行定义，标号必须以英文字母开头。在汇编语言程序中，标号可有可无。程序由编译系统汇编或手工汇编时，把标号替换成该指令代码的第一个字节所在单元地址。

(2) 操作码助记符：规定指令所执行的操作，描述指令的功能。操作码助记符在指令中不可缺少。

(3) 操作数：参与操作的数据信息。

(4) 注释：用户对指令的操作说明，便于阅读和理解程序。注释部分可有可无。

编制程序时，一般标号后带冒号(:)，与操作码助记符之间应有若干个空格；助记符和操作数用空格隔开；如果指令中包含多个操作数，操作数之间用逗号(,)隔开；注释与指令之间采用分号(;)隔开，一般情况下，在程序中，分号(;)之后的一切信息均为说明注释部分，在编译系统汇编时不予处理。

下面为一段汇编语言程序：

```
【标号】    【助记符】【操作数】    【注释】
START :     MOV      A,#20H      ;把数20H送入累加器A中
            INC      A           ;(A)加一
```

MCS-51单片机汇编语言指令有以下几种形式：

(1) 没有操作数——RET、RETI、NOP。

(2) 有1个操作数——INC A、DEC 20H、CLR C、SJMP NEXT。

(3) 有2个操作数——MOV R7,#DATA、ADD A、R0、DJNZ R2、LOOP。

(4) 有3个操作数——CJNE A、#20H、NEQ。

从机器语言的指令代码长度来看，MCS-51单片机汇编语言指令有以下3种形式：

(1) 单字节指令：指令机器代码为1个字节，占用1个单元。如：

```
INC DPTR            (指令机器代码：A3)
ADD A,R7            (指令机器代码：2F)
```

(2) 双字节指令：指令机器代码为2个字节，占用2个单元。如：

```
SUBB A,2BH          (指令机器代码：95 2B)
ORL C,/27H          (指令机器代码：A0 27)
```

(3) 3字节指令。如：

```
MOV 20H,#00H        (指令机器代码：75 20 00)
LJMP 2000H          (指令机器代码：02 20 00)
```

3.2　MCS-51单片机的寻址方式

所谓寻址方式(Addressing Mode)就是CPU执行指令时获取操作数的方式。寻址方式的多少是反映指令系统优劣的主要指标之一。寻址方式隐含在指令代码中，寻址方式越多，灵活性越大，指令系统越复杂。MSC-51单片机提供了7种不同的寻址方式：立即寻址、直接寻址、寄存器寻址、寄存器间接寻址、变址寻址、位寻址和相对寻址。

1. 立即寻址方式

立即寻址方式也称为立即数(Immediate Constants)寻址。立即寻址方式在指令中直接给出了参与运算的操作数,即由指令直接提供操作数,操作数紧跟在操作码之后。这种由指令直接提供的操作数叫“立即数”,是一个常数,操作数前面加有“#”号。

例如:

```
MOV A,#20H          ;把立即数 20H 送入累加器 A 中
```

2. 直接寻址方式

直接寻址方式(Direct Addressing)在指令中给出了参与运算的操作数所在单元的地址或所在位的位地址。操作数在指定的单元或位中。

例如:

```
MOV A,20H           ;把 20H 单元的内容(数)送到累加器 A 中
```

直接寻址方式可以访问 3 种地址空间。

(1) 内部 RAM:00H~7FH。

(2) 21 个特殊功能寄存器,对这些特殊功能寄存器只能采用直接寻址方式。

(3) 位寻址空间。

3. 寄存器寻址方式

寄存器寻址(Register Addressing)方式在指令中指出了参与运算的操作数所在的寄存器,操作数存储在寄存器中。

例如:

```
MOV A,R0            ;将工作寄存器 R0 中的数送到累加器 A 中
```

寄存器寻址方式中的寄存器为工作寄存器 R0~R7、DPTR、累加器 A、寄存器 B(仅在乘除法时)和布尔累加器 C。

4. 寄存器间接寻址方式

寄存器间接寻址(Indirect Addressing)方式在指令中用地址寄存器指出存放操作数的单元地址。地址寄存器(Address Register)的内容是操作数所在单元的地址。操作数是通过指令中给出的地址寄存器内容间接得到的。

作为地址寄存器的寄存器只有 R0、R1、DPTR,在指令中表示为@R0、@R1、@DPTR。

下面介绍寄存器间接寻址方式的寻址范围。

(1) 片内 RAM:00H~7FH,由 8 位地址寄存器@R0 和@R1 指出操作数所在单元的地址。

(2) 片外 RAM 和片外 I/O 口:0000H~FFFFH,由 16 位地址寄存器@DPTR 指出操作数所在单元或 I/O 口的地址,也可由 8 位地址寄存器@R0 和@R1 指出操作数所在单元的低 8 位地址,此时,高 8 位地址由 P2 口提供。

另外，还有一个隐含的8位地址寄存器SP，用于与堆栈操作相关的指令，由SP内容间接指出操作数所在单元的地址。

例如：已知PSW的内容是00H，寄存器R0的内容为20H，20H单元的内容为30H，则指令“MOV A，@R0”的功能是：把地址寄存器R0的内容20H作为操作数所在单元地址，该单元中的内容30H为操作数，把该数送到累加器A中。

例如：已知DPTR的内容为2020H，外部RAM2020H单元的内容为4AH。

则指令“MOVX A，@DPTR”的功能是：地址寄存器DPTR的内容2020H作为操作数所在单元地址，该单元中的内容4AH为操作数，把它到累加器A中。

5. 变址寻址方式

变址寻址(Indexed Addressing)方式也称为基址寄存器加变址寄存器间接寻址方式。存放操作数单元的地址为基址寄存器和变址寄存器二者内容之和。变址寻址方式中，操作数所在单元的地址以基址寄存器与变址寄存器内容之和的形式在指令中指出。这种寻址方式只适用于程序存储器，用于读取存储在程序存储器(ROM)中的操作数。在MCS-51单片机中，2个16位寄存器DPTR和PC可以作为基址寄存器，可作为变址寄存器的只有累加器A。从程序存储器中读取操作数的指令有2种：

```
MOVC A,@A + DPTR
MOVC A,@A + PC
```

基址寄存器PC或DPTR与累加器A两者内容相加得到的16位地址作为操作数所在单元的地址，把该地址对应单元的内容取出送入累加器A。

另外一种变址寻址方式的指令用于程序散转指令，在这种情况下，程序转移的目标地址(Destination Address)由基址寄存器DPTR与累加器A内容之和确定，二者内容之和传送给程序计数器PC，程序执行的顺序发生改变，从而实现程序转移。指令如下：

```
JMP @A + DPTR
```

6. 位寻址方式

位寻址(Bit Addressing)方式是在指令中指出了参与运算的操作数(1位)所在的位的位地址或位寄存器(仅有位累加器C)。位寻址方式是MCS-51单片机特有的一种寻址方式。位寻址方式的适用范围为MCS-51单片机的整个位寻址空间，即内部RAM的位寻址区(20H～2FH的16个单元的128位)，位地址范围为00H～7FH，特殊功能寄存器(SFR)区的某些寄存器中的位，位地址范围为80H～FFH。

在指令中位地址通常以下列几种形式之一表示：

(1) 直接位地址方式。如：

```
CLR  07H
MOV  22H,C
```

(2) 被操作位在单元或特殊功能寄存器中相对位置，常用点操作符方式。如：

```
SETB  ACC.6
```

```
ANL   C,/25H.5
```

(3) 特殊功能寄存器中规定的位名称。如：

```
CPL  RS0
JBC  TF0,OVER
```

实际上，位寻址方式在指令中指出参与操作的位的位地址时，可以理解为直接寻址方式；而在指令中指出参与操作的位所在的位寄存器(位累加器 C)时，可以认为是寄存器寻址方式。

7. 相对寻址方式

相对寻址(Relative Addressing)方式是在指令中给出了程序转移的目标地址与当前地址之间的相对偏移量。它是为解决程序转移而专门设置的，用于控制转移类指令。相对偏移量为一个字节的补码，在指令代码中用 rel 表示，取值为－128～＋127，若相对偏移量 rel 大于 0；则程序向下转移；若偏移量 rel 小于 0，则向上反向转移。程序转移的目标地址为当前地址与偏移量 rel 之和，该值送给程序计数器 PC，则程序转移到目标地址处执行。

下面介绍相对寻址方式时目标地址计算的过程和相关概念，以便说明程序转移的原理。程序计数器 PC 的作用是指出 CPU 将要执行的指令代码所在的单元，PC 的内容是该单元的地址，PC 的内容确定了 CPU 将从何处取指令以及程序的执行方向。

为了解释相对寻址方式，我们把一条指令的第 1 个指令代码所在的单元地址称为源地址。当 PC 指向该单元时，CPU 开始执行该指令，每取一次指令代码(1 个字节)，PC 内容自动加 1；指令被 CPU 执行的标志是它的所有指令代码全部被读到 CPU 中，即 PC 内容变为源地址的基础上加上了这条指令的指令代码长度，这个地址称为当前地址，即：

```
当前地址 = 源地址 + 指令代码长度
```

如果这条指令为相对转移指令，那么，执行这条指令的结果将改变程序执行的顺序，即修改 PC 内容，产生下一步 CPU 读取指令的单元地址，即目标地址。因为在指令代码中已经告知 CPU 需要转移的相对偏移量 rel，则 CPU 根据转移指令提供的 rel 进行如下运算：

```
目标地址 = 当前地址 + rel
```

则程序转移到目标地址处执行。

下面一段程序存储在程序存储器 0100H 开始的区域。存储在程序存储器的指令代码映像如图 3.1 所示。

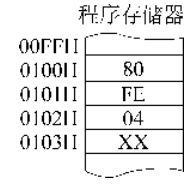

图 3.1　程序存储器的指令代码

【汇编语言】	【注释】	【单元地址】	【指令代码】
HERE:　SJMP　HERE	；程序在 HERE 处循环	0100H	80　FE
INC　A		0102H	04

- 源地址：0100H。
- 指令代码给出的偏移量：FEH，它是－2 的补码。
- SJMP 指令代码为 2 字节，当前地址：0100＋2＝0102H。

• 目标地址：0102H＋(－2)＝0100H。

这样，程序会转移到0100单元执行，即在HERE处循环，而不会执行“INC A”。

下面这一段程序是带有条件判别的程序，存储在程序存储器0200H开始的区域。

【汇编语言】	【注释】	【单元地址】	【指令代码】
MYPROG: MOV A,20H	;	0200H	E5　20
ADD A,30H	; 2个单元内容相加	0202H	25　30
JC　CARRY	; 有进位时，转移	0204H	40　03
MOV 20H,A	; 存结果	0206H	F5　20
RET		0208H	22
CARRY: SETB 28H.0	; 标志位置1	0209H	D2　40
…		020BH	…

在本程序中，如果进位(Cy)＝1，程序应转移到CARRY处执行。对于转移指令“JC CARRY”来说，源地址为0204H，指令代码给出的偏移量是03H，SJMP指令代码为2字节，则当前地址：0204＋02＝0206H；目标地址：0206H＋03＝0209H。这样，程序会转移到0209H单元执行，即转移到CARRY处执行。

3.3　指令系统分析

3.3.1　指令的分类

MCS-51指令系统中共有111条指令。

按指令代码的字节数划分，可分为3类：

(1) 单字节指令(49条)。

(2) 双字节指令(45条)。

(3) 三字节指令(17条)。

按指令执行的时间划分，可分为3类：

(1) 单机器周期指令(64条)。

(2) 双机器周期指令(45条)。

(3) 四机器周期指令(2条)。

按指令功能进行划分，可分5大类：

(1) 数据传送类指令(29条)。

(2) 算术运算类指令(24条)。

(3) 逻辑运算类指令(24条)。

(4) 控制转移类指令(17条)。

(5) 位操作类指令(17条)。

本章后面将按指令功能详细介绍MCS-51的指令系统，主要以汇编语言指令为主，指令的机器码(指令代码)请查阅附录的MCS-51单片机指令及其指令代码表。为了说明指令的功能，下面定义一些在汇编指令中用到的符号和字段。

- Rn：n=0～7，表示当前工作寄存器的 8 个工作寄存器 R0～R7。
- @Ri：i=0 或 1，表示作为地址寄存器的工作寄存器 R0 和 R1。
- direct：表示一个单元地址，8 位二进制数，取值范围为 00H～FFH。因此，可以是内部 RAM 的单元地址(00H～7FH)，也可以是特殊功能寄存器(SFR)的地址(80H～FFH)。
- ＃data：表示一个 8 位二进制常数，取值范围为 00H～FFH。
- ＃data16：表示一个 16 位二进制常数，取值范围为 0000H～FFFFH。
- addr16：表示一个 16 位单元地址，16 位二进制数，取值范围为 0000H～FFFFH。
- addr11：表示一个 11 位单元地址，11 位二进制数，取值范围为 000 0000 0000B～111 1111 1111B。
- bit：表示一个位地址，8 位二进制数，取值范围为 00H～FFH。bit 可以是内部 RAM(20H～2FH)或是特殊功能寄存器(SFR)中的可寻址位的位地址。
- rel：表示指令机器代码中的相对偏移量，8 位带符号二进制数，补码，取值范围在－128～＋127 之间。
- (direct)：表示由地址 direct 指定的寄存器或单元的内容。
- [(AddReg)]：表示由地址寄存器 AddReg 内容所指定的存储单元的内容。
- LABEL：在汇编语言程序中指定的标号。

另外，在 MCS-51 单片机系统中，除了 16 位寄存器(PC 和 DPTR)和布尔累加器(位处理器)C，我们提到的单元或寄存器内容为 8 位二进制数，数值范围为 00H～FFH，不管它是属于内部 RAM、外部 RAM、特殊功能寄存器、程序存储器。存储在位寻址空间上的数据信息称为状态，1 位的状态有 2 种：0 和 1。

3.3.2 数据传送类指令

数据传送(Data Transfers)类指令共有 29 条，是 MCS-51 单片机指令系统中种类最多、程序中使用最频繁的一类指令。数据传送类指令分为以下 5 种类型：

(1) 通用传送指令。

(2) 堆栈操作指令。

(3) 交换指令。

(4) 访问程序存储器的指令。

(5) 访问外部 RAM 的指令。

1. 通用传送指令

通用传送指令的助记符为 MOV，指令的一般形式为：

```
MOV  目的操作数,源操作数
```

该类指令的功能是：把源操作数送给目的操作数，源操作数保持不变。除非是 PSW 作为目的操作数，指令执行时，一般不影响标志位 Cy、AC、OV，但累加器 A 作为目的操作数时，将会对奇偶校验位 P 产生影响。这类指令操作均在单片机内部 RAM 或特殊功能寄存器区。

我们分别以累加器 A、工作寄存器 R0～R7、某一个单元为目的操作数，介绍通用传送指令的功能。

1）以 A 为目的操作数的传送指令

```
MOV  A,源操作数
```

源操作数复制给累加器 A。有以下 4 种形式：

```
MOV    A,Rn              ;(Rn)→(A),n=0～7
MOV    A,direct          ;(direct) →(A)
MOV    A,@Ri             ;[(Ri)] →(A),i=0,1
MOV    A,#data           ;data →(A)
```

如：

```
MOV    A,R2              ;(R2)→(A),把寄存器 R2 的内容复制给累加器 A
MOV    A,30H             ;(30H) →(A),把内部 RAM 地址为 30H 单元的内容复制给
                         ;累加器 A
MOV    A,@R0             ;[(R0)] →(A),把地址寄存器 R0 内容指定的内部 RAM 单元内容复制给累
                         ;加器 A。也就是把内部 RAM 一个单元的内容送给累加器 A,该单元的地址
                         ;是由地址寄存器 R0 的内容指定的
MOV    A,#36H            ;把常数 36H 存放在累加器 A 中
```

2）以工作寄存器 Rn 为目的操作数的传送指令

```
MOV  Rn,源操作数
```

源操作数送给工作寄存器 Rn，有 3 种形式：

```
MOV    Rn,A              ;(A) →(Rn),n=0～7
MOV    Rn,direct         ;(direct)→(Rn),n=0～7
MOV    Rn,#data          ;data →(Rn),n=0～7
```

如：

```
MOV    R0,A              ;(A) →(R0),把累加器 A 的内容送到寄存器 R0 中
MOV    R3,30H            ;(30H) →(R3),把内部 RAM 地址为 30H 单元的内容复制给寄存器 R3
MOV    R7,#36H           ;36H →(R7),把常数 36H 存放在寄存器 R7 中
MOV    R1,#30            ;30 →(R1),把十进制数 30 送到 R1 中,(R1)=1EH
MOV    R6,#01101100B     ;把二进制数 01101100B 送到 R1 中,(R1)=6CH
```

3）以直接地址为目的操作数的传送指令

```
MOV  direct,源操作数
```

把源操作数送到指定单元 direct 中，这是通用传送指令中操作形式最为丰富的一组指令，支持任意两个单元之间的数据传送，不管该单元的地址是直接给定的还是由地址寄存器间接指出来的。这组指令有 5 种形式：

```
MOV    direct,A          ;(A)→(direct)
MOV    direct,Rn         ;(Rn)→(direct),n=0～7
MOV    direct,direct1    ;(direct1) →(direct)
```

```
MOV    direct,@Ri      ;[(Ri)]→(direct),i=0,1
MOV    direct,#data    ;data→(direct)
```

如：

```
MOV    30H,A           ;(A)→(30H),把累加器A的内容复制给内部RAM为30H单元
MOV    P1,R2           ;(R2)→(P1),把R2的内容从P1口输出
MOV    38H,60H         ;(60H)→(38H),把60H单元的内容复制给38H单元
MOV    TL0,@R1         ;[(R1)]→(TL0),把地址寄存器R1内容指定单元的内容送到特殊功能寄
                       ;存器TL0中
MOV    58H,#36H        ;36H→(58H),把常数36H写入内部RAM的58H单元
```

4）以间接地址为目的操作数的传送指令

```
MOV  @Ri,源操作数
```

这是一组以某一个单元为目的操作数传送指令，与上一组指令不同的是，单元地址不是直接给出的，而是由地址寄存器R0或R1的内容间接给出的，有3种形式：

```
MOV    @Ri,A           ;(A)→[(Ri)],i=0,1
MOV    @Ri,direct      ;(direct)→[(Ri)],i=0,1
MOV    @Ri,#data       ;data→[(Ri)],i=0,1
```

如：

```
MOV    @R0,A           ;(A)→[(R0)],把累加器A的内容送给地址寄存器R0内容指定的单元
MOV    @R1,36H         ;(36H)→[(R1)],把36H单元的内容送给另一个单元,该单元的地址由地
                       ;址寄存器R1内容指定
MOV    @R0,SBUF        ;(SBUF)→[(R0)],把串行口接收缓冲器SBUF的内容送给地址寄存器R0
                       ;内容指定的单元
MOV    @R0,#0D6H       ;给地址寄存器R0内容指定的单元设置常数D6H
```

例 3.1 已知(PSW)=00H,(A)=11H,(20H)=22H,分析下列程序的执行结果。

```
MOV    R0,A
MOV    R1,20H
MOV    R2,#33H
```

分析：(PSW)=00H意味着当前工作寄存器组为BANK0,R0～R7对应00H～07H单元。程序分析如下：

```
MOV    R0,A            ;(A)→(R0),则(R0)=11H
MOV    R1,20H          ;(20H)→(R1),则(R1)=22H
MOV    R2,#33H         ;33H→(R2),则(R2)=33H
```

执行结果为：(R0)=(00H)=11H,(R1)=(01H)=22H,(R2)=(02H)=33H。

例 3.2 已知(PSW)=00H,(A)=11H,(00H)=22H,(01H)=36H,(36H)=33H,(33H)=44H,分析下列程序的执行结果。

```
MOV    30H,A
MOV    31H,R0
MOV    32H,33H
```

```
MOV    34H,@R1
MOV    35H,#55H
```

分析：因为(PSW)＝00H，则当前工作寄存器组为 BANK0，R0～R7 对应 00H～07H 单元，所以(R0)＝22H，(R1)＝36H，程序分析如下：

```
MOV    30H,A          ;(A)→(30H),(30H)=11H
MOV    31H,R0         ;(R0)→(31H),(31H)=22H
MOV    32H,33H        ;(33H)→(32H),(32H)=44H
MOV    34H,@R1        ;[(R1)]→(34H),即[36H]→(34H),则(34H)=33H
MOV    35H,#55H       ;(35H)=55H
```

执行结果为：(30H)＝11H，(31H)＝22H，(32H)＝44H，(34H)＝33H，(35H)＝55H。

例 3.3　内部 RAM30H 单元的内容为 40H，40H 单元内容为 10H，当前从 P1 口输入数据 11001010B，分析下列程序的执行结果。

```
MOV    R0,#30H        ;30H→(R0),(R0)=30H
MOV    A,@R0          ;[(R0)]→(A),即[30H]→(A),(A)=40H
MOV    R1,A           ;(A)→(R1),(R1)=40H
MOV    B,@R1          ;[(R1)]→(B),即[40H]→(B),(B)=10H
MOV    @R1,P1         ;(P1)→[(R1)],即(P1)→[40H],(40H)=11001010B=0CAH
MOV    P2,P1          ;(P2)=0CAH
```

执行结果为：(30H)＝40H，(R0)＝30H，(R1)＝40H，(A)＝40H，(B)＝10H，(40H)＝0CAH，(P2)＝0CAH。

5）十六位数据传送指令

```
MOV    DPTR,#data16   ;data8～15→(DPH),data0～7→(DPL)
```

这是 MCS-51 单片机指令系统中唯一一条设置 16 位二进制常数的指令。该指令操作等同于分别对 DPH 和 DPL 的操作：

```
MOV    DPH,#data8～15
MOV    DPL,#data0～7
```

两种实现方法的区别在于：前者为 3 字节指令，指令周期为 2 个机器周期，而后者为 2 条指令共 6 个字节，需要 4 个机器周期才能执行完毕。如：

```
MOV    DPTR,#2368H    ;(DPTR)=2368H;(DPH)=23H,(DPL)=68H
MOV    DPTR,#35326    ;(DPTR)=35326(89FEH)
```

在使用通用数据传送指令时，应注意以下几点：

(1) MCS-51 单片机不支持工作寄存器 R0～R7 内容直接传送给由地址寄存器内容指定的单元，或由地址寄存器内容指定单元的内容送给工作寄存器 R0～R7，如果在程序中需要这样的数据传送，可以采用其他方式间接实现。

例如：希望把地址寄存器 R1 内容指定的单元内容传送给工作寄存器 R5，可以采用

```
MOV    A,@R1
MOV    R5,A
```

把 R7 内容送给地址寄存器 R0 内容指定的单元，可以用下面的方法：

```
MOV    30H,R7
MOV    @R0,30H
```

(2) 虽然可以将一个指定的特殊功能寄存器的内容复制给累加器 A，但下面这条指令是无效的：

```
MOV    A,ACC
```

(3) 在通用数据传送指令中，地址寄存器只能由工作寄存器 R0 和 R1 担当，其他工作寄存器没有这个功能。

(4) 虽然 MCS-51 单片机由 2 个 16 位的寄存器：PC 和 DPTR，但只有 DPTR 用户可以用指令方式直接设置其内容。

2. 堆栈操作指令

堆栈是在内部 RAM 中开辟的一个先进后出(后进先出)的区域，用来保护 CPU 执行程序的现场，如 CPU 响应中断和子程序调用时的返回地址、重要单元和寄存器的内容等。其中重要单元和寄存器的内容采用堆栈操作指令完成。在保护时，先把它们传送到堆栈区暂时保存，待需要时再从堆栈区取出送回原来的单元和寄存器。堆栈的操作有 2 种：入栈和出栈。

1) 入栈指令

```
PUSH direct
```

CPU 操作：(SP)+1 →(SP)，修改堆栈指针；(direct) →[(SP)]，入栈，把指定单元 direct 的内容送到由堆栈指针 SP 内容所指的单元中。

入栈指令的功能是把指定单元 direct 的内容压入堆栈，指令执行时不影响标志位 Cy、AC、OV、P。

例如：

```
MOV    SP,#70H        ;(SP) = 70H,栈区开辟在 71H 开始的内部 RAM 区域
PUSH   60H            ;(SP) + 1→(SP),则(SP) = 71H
                      ;(60H)→[(SP)],则(60H)→(71H),60H 单元的内容被保存到栈区的 71H
                      ;单元中,60H 单元的内容没有改变
```

2) 出栈指令

```
POP    direct
```

CPU 操作：[(SP)] →(direct)，出栈，把堆栈中由(SP)所指单元的内容传送到指定单元 direct；(SP)−1 →(SP)，修改堆栈指针。

出栈指令功能是把堆栈中由(SP)所指单元的内容弹出到指定的 direct 单元。指令执行时不影响标志位 Cy、AC、OV、P。

例如：

```
MOV    SP,#71H        ;(SP) = 71H,目前栈顶为 71H 单元
```