第 3 章　MCS-51 单片机的指令系统

人们是通过单片机的指令系统控制单片机工作的。人机之间进行交流信息、驱动单片机实现各种功能操作的最直接、最基本的命令是一种直接面向机器的语言。理解并熟练掌握单片机的指令系统是应用单片机的前提。本章将详细介绍 MCS-51 系列单片机指令系统的格式、寻址方式、各类指令及其功能特点。

3.1　指令系统概述

指令是直接控制单片机相关硬件完成基本操作的命令。一个单片机能够执行的所有指令的集合，称为该单片机的指令系统。指令系统与机器硬件密切相关，不同系列型号的单片机有不同的指令系统。指令系统是由单片机生产厂商定义并集成在单片机中的，体现单片机的主要功能，也是表征单片机性能的重要指标之一。专用于 MCS-51 系列单片机的指令系统（即 MCS-51 单片机指令系统）共有 111 条，是本章学习的主要内容。

指令系统作为一种直接面向机器的计算机语言，发展出了两种语言形式：机器语言指令和汇编语言指令，二者本质上都是面向机器的语言，即机器语言，它们之间存在一一对应的关系，下面将详述两种指令的特点。

3.1.1　机器语言指令与汇编语言指令

单片机作为现代计算机发展的一个分支，与现代计算机同属于冯·诺依曼体系计算机。因此，单片机的指令和所处理的数据均用二进制码表示。

机器语言指令用二进制码表示，又称为机器码指令或机器指令，能够直接被计算机硬件识别和执行，是唯一一种可以被计算机硬件直接识别和执行的计算机语言。

例如，执行累加器 A 内容加 1 操作的 MCS-51 单片机指令对应的二进制代码为

```
0000 0100B
```

为了书写和阅读方便，常把它写成十六进制形式，十六进制表示的机器码为 04H。

机器语言是计算机自身固有的二进制形式的语言，可以被计算机直接识别并加以分析和执行，是计算机发展初期人们使用的编程语言。由于数字形式的指令很难与其功能联系起来，因此二进制形式的机器语言指令即便用十六进制去书写与记忆也是极其不方便且困难的。

为了克服机器语言指令难记、不易书写、难以阅读和调试、容易出错且不易查找错误、程序可维护性差等缺点，计算机制造厂家对指令系统的每一条机器语言指令都给出了助记符。助记符是用英文缩写来描述相应机器语言指令的功能，它不但便于记忆，也便于理解和分类。这种用指令助记符来表示的机器语言指令称为汇编语言指令。显然，汇编语言指令实际上是机器语言指令的符号化表示，故又称为符号语言。

例如上述指令：执行累加器 A 内容加 1 操作，机器码为 04H，汇编语言指令为

```
INC A
```

其中，助记符 INC 是英文单词增加的缩写，在此表示加 1 的意思。

由于汇编语言指令是机器语言指令的符号化表示，所以它与机器语言指令有着一一对应关系。与机器语言指令相比，汇编语言的特点是直观、易学习、易于记忆理解，编程方便；但用汇编语言编写的程序已不能被计算机识别和理解，须将其转换为对应的计算机唯一可以识别和执行的机器语言指令程序。将汇编语言源程序转换为机器语言程序的过程称为汇编。

可见，一条指令可以用机器语言和汇编语言两种语言形式表示。为了对指令有进一步理解，接下来详细分析指令的结构与格式。

3.1.2 指令格式

1. 机器语言指令格式

一条指令通常是由操作码和操作数两部分组成。操作码用来指明该指令所完成的操作，如传送、加法、减法、移位或转移等；通常，其位数反映了机器的操作种类；操作码的长度可以是固定的也可以是变化的。操作数用来指明操作码操作的对象；操作数可以是一个具体的数据，也可以是存储数据的地址或寄存器。指令的基本格式如图 3-1 所示。

操作码	操作数

图 3-1 指令的基本格式

MCS-51 单片机作为一种典型的 8 位微控制器，指令的操作码长度为 8 位。8 位长度的操作码在理论上最多能提供 $2^8=256$ 种具体操作。在 MCS-51 单片机中，设置了 255 种具体操作，其中 A5H 编码未定义。在 MCS-51 单片机指令中，操作码可以带有 0～3B 的操作数。带 0B 的操作数即没有操作数的指令通常是没有意义的。在 MCS-51 单片机指令中，没有操作数的指令只有一条空操作指令，绝大多数指令的操作码通常带有 1～3B 的操作数。需要说明的是，有些特殊形式的操作数（如数据位于寄存器时）是可以隐含在操作码中的，即不单独占用存储空间。指令的长度（字节数）等于操作码长度（字节数）与操作数长度（字节数）之和。MCS-51 单片机指令长度最长为 3B。

MCS-51 单片机指令按指令字节长度分，可分为单字节指令、双字节指令和三字节指令 3 种。MCS-51 单片机作为一种 8 位机，其操作数绝大多数为单字节操作数。

（1）单字节指令。指令长度只有一字节，除空操作指令外，操作数可以有 1 或 2 个，但都隐含在操作码中。例如指令

```
INC  A
MOV  @R0,A
```

（2）双字节指令。一字节的操作码，可以有 1 或 2 个操作数。例如指令

```
ADD  A,#22H
```

其中，操作码为 24H，操作数为 22H，目的操作数 A 隐含在操作码中，这种指令在内存中占 2B。

(3) 三字节指令。一字节操作码，可以有 2 或 3 个操作数。例如指令

```
MOV  5EH,4FH
```

该指令执行时把 4FH 地址单元的内容送到 5EH 地址单元中，操作码为 85H，操作数分别为源操作数 4FH 和目的操作数 5EH，它在内存中占 3B。

图 3-2 给出了机器码指令的 3 种形式在内存中的数据安排。

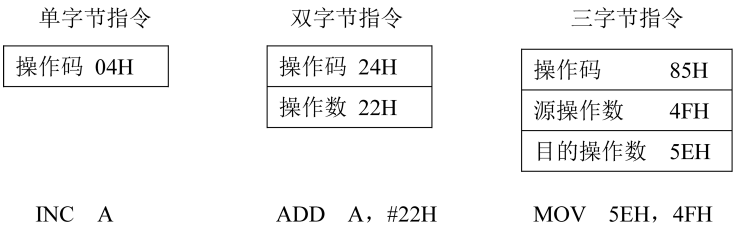

图 3-2 机器码指令格式

2. 汇编语言指令格式

通常，MCS-51 单片机典型汇编语言指令的格式如下：

```
[标号:] 操作码助记符 [目的操作数] [,源操作数] [;注释]
```

每一部分构成汇编指令的一个字段，各字段之间用空格或规定的标点符号隔开。方括号内的字段可以省略。

各字段的含义如下。

(1) 标号：指令的符号地址。它通常代表一条程序语句(助记符指令)的机器代码所在存储单元的地址。也就是说，在将汇编语言指令转换(即汇编)为机器代码时，它将被赋予该语句指令代码(首字节)存储单元的地址值。一条语句之前是否要冠以标号，根据程序的需要而定，当某条语句可能被调用或作为转移的目的地址时，通常要给该语句赋予标号，一旦某条语句被赋予标号，该标号就可作为其他语句的操作数使用。

标号书写时后面需要加冒号；标号可以单独占用一行；同一标号在同一程序中只能定义一次，不能重复定义；标号通常由 1～8 个 ASCII 码字符组成，第一个字符必须是字母，且标号名称不能是汇编语言保留字。

不同的机器对标号字段的长度和构成有不同的规定，使用时应注意。

(2) 操作码助记符：表示指令进行何种操作，用助记符形式给出。助记符一般为英语单词的缩写，是不可缺少的字段。MCS-51 单片机汇编语言指令中共有 42 种助记符，代表了 33 种不同的功能。

(3) 操作数：指令操作的对象。常见的指令操作数通常可分为目的操作数(如上例中累加器 A)和源操作数(如上例中的 5FH)，任何指令的操作都是实现“从源操作数到目的操作数”。因此，目的操作数和源操作数的书写顺序不能颠倒。操作数可以是数字(地址、数据)，也可以是标号或寄存器名等。也有些指令不需要指明操作数。

注意：目的操作数不仅是操作码操作对象的数据来源之一，而且还要具有保存操作结果的功能。

注释：对指令功能的说明，便于程序的阅读和维护，它不参与计算机的操作。

汇编指令各字段之间的标点符号应严格按照规定的格式书写。操作码助记符与目的操作数之间用空格分隔开。

例如指令：

```
START: MOV  A,5FH ;(A)←(5FH)
```

该指令表示将地址为 5FH 的内存单元的内容复制到累加器 A 中。

为了加深对指令的理解，在本书中 MCS-51 单片机指令的介绍以汇编语言指令讲述与分析为主、机器语言指令为辅，并将二者加以对照。

3.1.3 指令分类

指令一般有时间、空间和功能 3 种属性。时间属性是指一条指令执行所耗费的时间，一般用机器周期表示；空间属性是指一条指令在程序存储器中所占用的字节数；功能属性是指每条指令所实现的特定的功能。

MCS-51 单片机汇编语言指令共有 111 条，包含有 42 种助记符，对应着 255 种具体的操作。根据指令属性的不同，111 条汇编指令有 3 种不同的分类方法。

(1) 根据指令的时间属性不同，111 条汇编指令可分为 3 种：单机器周期指令(64 条)、双机器周期指令(45 条)、四机器周期指令(2 条)。

(2) 根据指令的空间属性不同，111 条汇编指令可分为 3 种：单字节指令(49 条)、双字节指令(45 条)、三字节指令(17 条)。

(3) 根据指令的功能属性不同，任何类型单片机的指令系统实现的基本功能至少应包含数据传输类指令、运算类指令(包括算术运算、逻辑运算和位运算等)以及转移类指令等。MCS-51 单片机的 111 条汇编语言指令按照功能可以细分为 5 类。

(1) 数据传输类指令(29 条)；

(2) 算术运算类指令(24 条)；

(3) 逻辑操作类指令(24 条)；

(4) 控制转移类指令(17 条)；

(5) 位操作类指令(17 条)。

3.3 节中将按照上述指令功能的分类分别加以详述。

3.2 寻 址 方 式

寻址方式是指在一条指令执行过程中，指令中操作数地址(存放位置)的寻找方式。操作数从内容上看可分为数据(地址)和指令地址两种类型，因此寻址方式可分为两类：一是数据寻址；二是指令寻址(如转移指令、调用指令)。一般来说，指令系统中，大多数的指令的寻址方式属于数据寻址，指令寻址仅仅限于部分转移类指令中。

在指令执行时，如果要对数据操作，就需要解决“数据在何处”的问题。这就涉及单片机

的寻址方式。在单片机系统中，指令操作数的存放位置通常有几种方式：一是数据直接放在指令中，这种寻址方式称为常数寻址，又称为立即寻址；二是数据存放在寄存器中，然后寄存器出现在指令中，这种寻址方式称为寄存器寻址；三是数据存放在存储器单元中，存储器单元的地址出现在指令中，这种寻址方式称为直接寻址；四是数据存放在存储器单元中，存储器单元的地址再放在寄存器中，该寄存器出现在指令中，这种寻址方式称为寄存器间接寻址或者二次寻址。寄存器间接寻址方式可以衍生出多种变形，如基址变址寻址等。在MCS-51单片机中，数据寻址方式有立即寻址、寄存器寻址、直接寻址、寄存器间接寻址和变址寻址5种。

指令寻址用在控制转移指令中，其功能是得到转移的目的位置的地址。在MCS-51单片机中，根据目的位置提供方式的不同，指令寻址可分为3种：一是绝对寻址，仅用在AJMP指令和ACALL指令中；二是长寻址，仅用在LJMP指令和LCALL指令中；三是相对寻址。前两种方式比较简单，将在学习相关指令时详细介绍，第3种寻址方式在本节中介绍。

MCS-51单片机还包含有一个1位机，通常称为布尔处理器，其位操作数的存放形式称为位寻址方式。

在本节中，将详细介绍MCS-51单片机指令系统的立即寻址、寄存器寻址、直接寻址、寄存器间接寻址、变址寻址、相对寻址以及位寻址7种寻址方式，其中前6种为MCS-51单片机8位机的寻址方式，最后一种为布尔处理器的寻址方式。

3.2.1 立即寻址

立即寻址方式是指操作数直接出现在指令中。操作数可以是8位或16位(该数称为立即数)。立即数前面加符号#，以区别于操作数地址。例如：

```
MOV  DPTR,#2345H;  (DPTR)←2345H
```

的含义是把16位数2345H送入数据指针寄存器DPTR中，用符号表示为(DPTR)←2345H。又如：

```
MOV  A,#41H;  (A)←41H
MOV  A,41H;  (A)←(41H)
```

其中，第一条指令的含义是将立即数41H送累加器A中，第二条指令的含义是将片内RAM地址为41H单元中的内容送累加器A。可见，数41H之前有无#符号，其含义是不同的。

3.2.2 寄存器寻址

寄存器寻址是指以通用寄存器的内容作为操作数。在指令的助记符中直接以寄存器的名字来表示操作数的地址。在MCS-51单片机的CPU中，并没有专门的硬件通用寄存器，而是把片内数据RAM中的一部分(00H～1FH)作为工作寄存器来使用，每次都可以使用其中的任意一组，并以R0～R7来命名。至于具体使用哪一组，可在寄存器寻址指令前，通过PSW中的RS1和RS0来设定。例如：

```
MOV  A,R0
ADD  A,R0
```

这两条指令都属于寄存器寻址。前一条指令是将 R0 寄存器的内容传送到累加器 A，后一条则是对 A 和 R0 的内容做加法运算。

在 MCS-51 单片机中，用于寄存器寻址方式的寄存器有 Rn（n＝0，1，2，…，7）、A、B、DPTR 和 C（位累加器）。

3.2.3 直接寻址

在指令中，直接给出操作数地址的寻址方式称为直接寻址，此时，指令的操作数部分就是操作数地址。例如：

```
MOV  A,3AH;
```

其中，源操作数 3AH 表示直接地址，即片内 RAM 的 3AH 单元。指令的功能是把片内 RAM 3AH 单元的内容传送到累加器 A，并可表示为(A)←(3AH)。

适用于直接寻址方式访问的存储单元有以下几种。

(1) 内部数据 RAM 的低 128B。例如：

```
MOV  A,78H
ORL  A,77H
```

其中，77H 和 78H 都是片内 RAM 单元的地址。

(2) 特殊功能寄存器。例如：

```
MOV  TCON,A
```

其中，TCON 属于特殊功能寄存器，为符号地址，它所代表的直接地址是 88H。

特殊功能寄存器的访问只能采用直接寻址方式。

(3) 特殊功能寄存器中可位寻址的位地址空间。例如：

```
SETB  EA
```

其中，EA 是 IE 寄存器的第 7 位，它相应的直接地址为 0AFH。

(4) 内部数据 RAM 地址空间子集的 128 位（位地址空间）。例如：

```
MOV    C,7EH;
SETB   20H
```

在单片机的指令系统中，直接寻址方式很有用，尤其是按位的直接寻址可使程序设计变得简单、灵活。

在实际使用中，还应注意区别如下两点：第一，要注意直接寻址与寄存器寻址的区别。例如，指令 INC A 和指令 INC ACC 并不相同。在指令 INC A 中，A 属于寄存器寻址，A 对应的机器码是 04H。而在指令 INC ACC 中，ACC 代表累加器 A 的直接地址 E0H，属于直接寻址，所产生的机器码是 05E0H。但两条指令的功能是一致的，都是将累加器 A 的内容加 1。第二，在直接寻址中，要注意字节地址与位地址的区别。试比较

```
MOV  A,20H
```

与

```
MOV  C,20H
```

在前一条指令中,20H 是字节地址,因为目的操作数在 A 中,是 8 位数据;后一条指令的 20H 是位地址(代表 24H 单元的 D0 位),因为目的操作数在进位位 C 中,是一位的数据。由此可见,区分的方法在于指令的形式。

3.2.4 寄存器间接寻址

寄存器间接寻址是指指令中一个寄存器的内容是操作数的地址。此类指令执行时,先取得该寄存器的内容作地址,然后再到该地址对应的存储单元取得操作数。这是一种二次寻址方式,故称为间接寻址。指定的寄存器前用@标识(在寄存器前加@表示寄存器间接寻址)。

可用于间接寻址的寄存器有以下几种。

(1) R0 或 R1,用 R*i* 表示。可寻址内部低 128B 的 RAM 单元、增强型 51 单片机高 128B 内部 RAM 和外部 RAM 的 256 个单元。

格式:

```
MOV  A,@Ri
```

其中,R*i* 为 R0 或 R1。

例如:

```
MOV  R0,#80H
MOV  R1,#0BBH
MOV  A,@R0
MOV  A,@R1
```

(2) DPTR 可访问外部 RAM 64KB 空间。

格式:

```
MOVX  A,@DPTR
```

例如:

```
MOV   DPTR,#1234H
MOVX  A, @DPTR
```

顺便指出,在执行 PUSH(压栈)和 POP(出栈)指令时,需采用堆栈指针 SP 作寄存器间接寻址。

3.2.5 变址寻址

变址寻址是以某个寄存器的内容为基本地址,然后在这个基本地址基础上加上地址偏

移量才是真正的操作数地址。在 MCS-51 系统中没有专门的变址寄存器，而是采用数据指针 DPTR 或程序计数器 PC 的内容作为基本地址，地址偏移量则是累加器 A 的内容，并以 DPTR 或者 PC 的内容与 A 的内容之和作为实际的操作数地址。因此，这种寻址方式有两类：第一类用 PC 作为基地址加变址寄存器 A 的内容形成操作数地址。例如指令：

```
MOVC  A,@A+PC            ;(PC)←(PC)+1,(A)←((A)+(PC))
```

其中，指令操作码助记符号 MOVC 表示从程序存储器取数据。CPU 在读取本指令时，PC 已执行加 1 操作，指向下一条指令的首字节，所以作为基地址的已不是原来的 PC 值，而是 PC+1 值。A 中存放偏移量。偏移量加基地址就构成了所要读取的数据的地址。下面一段程序有助于对问题的说明：

```
地址          机器代码          源程序
0100H          7402          MOV   A,#02H
0102H          83            MOVC  A,@A+PC
0103H          00            NOP
0104H          00            NOP
0105H          32            DB    32H
```

当执行到“MOVC A,@A+PC”时，(A)= 02H，(PC)= 0103H，因此@A+PC 所指的地址应为 0103H+02H=0105H。该指令执行的结果是把 0105H 单元的内容 32H 送到 A 中。

第二类变址寻址是用 DPTR 作为基地址，A 作为变址寄存器，由@A+DPTR 形成操作数的地址。例如：

```
          MOV   A,#01H
          MOV   DPTR,#TABLE
          MOVC  A,@A+DPTR
TABLE:    DB    41H
          DB    42H
```

在该段程序中，表的首地址 TABLE 送到 DPTR 中作基地址，偏移量 01H 送到 A 中，因此@A+DPTR 形成操作数的地址为 TABLE+01H。执行该指令的结果是将数据 42H 送到 A 中。这两类变址寻址的方法，特别适用于对固定在程序存储器中的常数表格进行查表操作，因此上述两类指令又常称为查表指令。

必须指出，由于这两类查表指令各自的基址寄存器不同，其查表范围也不同。第一类查表指令的基址寄存器是程序计数器 PC。由于 PC 的内容是不能随意变更的，故查表范围只能由累加器 A 的内容来决定，所以使用本指令的表格只能存放在以 PC 当前值为起始地址的 256B 范围内。显然，这使表格的地址空间分配受到严格限制。以 DPTR 为基址寄存器的第二类查表指令，由于可以给 DPTR 赋予任意值，所以查表范围可达全部 64KB 的程序存储器空间。在每次查表前，只需对 DPTR 设置表首地址即可。如果 DPTR 已有他用，则在对 DPTR 装入新值前必须先保存其原值。

3.2.6 相对寻址

相对寻址是以 PC 的当前值为基准，加上指令中给出的相对偏移量 rel 形成有效的转移

地址。这里所讲的当前 PC 值是指执行相对转移指令时 PC 的值。一般将相对转移指令所在的地址称为源地址，转移后的地址称为目的地址，故有

目的地址 ＝ 源地址 ＋ 转移指令字节数 ＋ rel

其中，rel 是一个带符号的 8 位二进制数，常以补码的形式出现，因此程序的转移范围是以 PC 的当前值为起始地址，相对偏移范围为－128～127B。例如执行指令：

```
SJMP  rel
```

的机器码是 80H、rel 共 2B。设指令所在地址为 2000H，rel 的值为 54H，则转移地址为 2000H＋02H＋54H＝2056H，故指令执行后，PC 的值变为 2056H。

顺便指出一点，在 MCS-51 系统中，指令助记符号中的 rel 值和指令机器码中的 rel 值是相同的，不像某些系统中，这两者之间相差一个指令字节数。

3.2.7 位寻址

为了更好地面向控制，MCS-51 单片机内部专门设置了一个独立的位处理器——布尔处理器。它设有独立的位处理指令系统(共 17 条)。位处理指令采用位寻址方式来获得操作数，因此其操作数就是 8 位二进制数中的某一位。在指令系统中，位地址用 bit 表示。

在 MCS-51 系统的内部数据 RAM 有两个可以按位寻址的区域。其一是 20H～2FH 共 16 个单元中的每一位，共 128 位(对应的位地址是 00H～7FH)，都可以单独作为操作数；其二是某些特殊功能寄存器。凡是单元地址能被 8 整除的特殊功能寄存器都可以进行位寻址，其位寻址是 80H～FFH 中的一部分。

在 MCS-51 系统中，位地址的表示可以采用以下 4 种方式。

(1) 直接使用 00H～FFH 范围内的某一位的位地址来表示。

(2) 采用第几单元第几位的表示方法。例如，25H.5 表示 25H 单元的 D5 位。这种表示方法可以避免查表或计算，比较方便。

(3) 对于特殊功能寄存器，可直接用寄存器名加位数的表示法，例如 TCON.3、P1.0 等。

(4) 位名称方式，例如 PSW 中的 D7 位：CY。

3.3 MCS-51 单片机指令集

按照指令的功能，可以把 MCS-51 单片机指令系统的 111 条指令分为 5 类。

(1) 数据传送类指令(29 条)。

(2) 算术运算类指令(24 条)。

(3) 逻辑操作类指令(24 条)。

(4) 控制转移类指令(17 条)。

(5) 位操作类指令(17 条)。

在介绍指令之前，先对指令中的操作数约定符号进行简单介绍。

Rn，n＝0，1，…，7：当前被选定寄存器组的 8 个工作寄存器 R0～R7。

Ri，i＝0，1：当前被选定寄存器组的两个工作寄存器 R0 和 R1。

direct：为 8 位内部数据存储器单元的地址，它可以是内部 RAM(00H～7FH)某个单

元，或是一个特殊功能寄存器(SFR)的地址。

#data：为指令中的 8 位立即数。

#data16：为指令中的 16 位立即数。

addr16：表示 16 位目标地址，用于 LCALL 和 LJMP 指令，能调用或转移到 64KB 程序存储器地址空间的任何地方。

addr11：为 11 位目标地址，用于 ACALL 和 AJMP 指令。

rel：带符号的 8 位偏移地址，用于 SJMP 指令和所有条件转移指令中。偏移量从下一条指令的第一个字节单元开始计算，偏移量的取值范围为－128～127。

bit：表示位地址。

@：为寄存器间接寻址符号，如@R*i* 表示用寄存器 R*i* 间接寻址。

/：为位操作的前缀，表示对该位操作数取反，如 /bit。

(X)：表示 X 单元中的内容。

((X))：表示以 X 单元中的内容为地址的单元中的内容。

←：表示左边的内容被右边的内容代替。

↔：表示数据交换。

$：表示当前指令的地址。

3.3.1 数据传送类指令

1. 概述

数据传送操作是单片机系统中最频繁、最基本的操作，因此单片机指令系统中的数据传送类指令通常是最多的一类指令，也是编程时使用最多的一类指令。同样，MCS-51 指令系统中的数据传送指令也最多，有 29 条。

数据传送指令一般是把源操作数传送到目的操作数，指令执行后，源操作数内容不变，目的操作数单元的内容被源操作数内容取代。数据传送指令不影响标志位 C、AC、OV(写 PSW 除外)。这类指令的汇编语言格式如下：

```
操作码助记符  目的操作数[,源操作数]
```

数据传送类指令的操作码助记符共有 8 种，按照功能可将其分为 5 类。

(1) 访问片内 RAM 指令助记符：MOV。

(2) 访问片外 RAM 指令助记符：MOVX。

(3) 访问片内外程序存储器指令助记符：MOVC。

(4) 访问堆栈指令助记符：PUSH 和 POP。

(5) 数据交换类指令助记符：XCH、XCHD 和 SWAP。

数据传送类指令操作码助记符只有 8 种，为何其指令却有 29 条，原因就在于其操作数因其具有的多种寻址方式而带来的多样性。由 3.2 节可知，MCS-51 单片机之 8 位机指令操作数的寻址方式中，除仅用于转移类指令的指令寻址外，其他类指令的操作数均为数据寻址，即立即寻址、寄存器寻址、直接寻址、寄存器间接寻址、变址寻址 5 种，其中，变址寻址可以看作是寄存器间接寻址的一种变形。因此，绝大多数指令操作数的寻址方式可以归为立即寻址、寄存器寻址、直接寻址和寄存器间接寻址 4 类。

在 MCS-51 单片机中，寄存器只有 A、B、DPTR、R*n* 这 4 种，而寄存器 B 仅用于乘、除两条指令中；可以用于间接寻址的寄存器有 R*i*、DPTR 两种。根据由操作数寻址方式得到的操作数的具体形式，结合源操作数和目的操作数的特点，可以分析出，作为源操作数的具体形式有累加器 A，工作寄存器 R*n*，直接地址 direct，间接寻址寄存器@DPTR/@R*i* 和立即寻址＃data8/＃data16 这 5 种；可以作为目的操作数的具体形式有累加器 A、工作寄存器 R*n*、直接地址 direct、间接寻址寄存器@DPTR/@R*i* 和 DPTR 这 5 种。

MCS-51 单片机汇编语言指令系统将 42 种操作码助记符结合上述操作数寻址方式的具体形式构造出了 MCS-51 单片机的 5 类功能指令共计 111 条。理解并熟记 42 种操作码助记符以及各种操作数的具体形式，同时注意分析不同功能指令的组合特点，可以帮助人们准确快速地记忆理解这 111 条汇编指令。在 MCS-51 单片机指令中，目的操作数与源操作数不允许出现完全相同的操作数形式，如同为累加器 A；若一方操作数为 R*n*，@R*i* 中任何一种形式的工作寄存器时，则另一方操作数不允许使用任何形式的工作寄存器，如源操作数为 R7，则目的操作数不能再使用工作寄存器 R*n*，以及@R0 或@R1。

根据数据传送类指令操作码助记符的不同功能分类，29 条指令可以分为 5 类。

(1) 片内 RAM 数据传送指令(16 条)。

(2) 片外 RAM 数据传送指令(4 条)。

(3) 片内外程序存储器访问指令(2 条)。

(4) 堆栈操作指令(2 条)。

(5) 数据交换指令(5 条)。

2. 指令详解

(1) 片内 RAM 数据传送指令(16 条)。

根据目的操作数的不同，又可分为以下 5 类。

① 以累加器 A 为目的操作数的指令(4 条)，如表 3-1 所示。

表 3-1　以累加器 A 为目的操作数的指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOV　A,R*n*	1110 1*rrr*	E8H～EFH	(A)←(R*n*)，*n*＝0～7
MOV　A,direct	1110 0101 direct	E5H direct	(A)←(direct)
MOV　A, @R*i*	1110 011*i*	E6H～E7H	(A)←((R*i*))，*i*＝0,1
MOV　A,＃data	0111 0100 ＃data	74H ＃data	(A)←＃data

上述指令的注释中，符号()←()表示左边的内容被右边的内容代替；如(A)←(R*n*)，表示累加器 A 中的内容被 R*n* 的内容取代，即 R*n* 的内容送累加器 A 中。其余注释类推，以下同。第一条机器码指令中的 *rrr* 为工作寄存器地址，*rrr* ＝ 000～111，对应工作寄存器 R0～R7。R*i* 为间接寻址寄存器，*i*＝0 或 1，即 R*i* 为 R0 或 R1。如上所述，在给出机器码指令时，多字节指令的机器码在存储器中的存放顺序按以下规则说明：

每条指令的第一字节操作码放在第一行，第二字节操作数放在第二行，第三字节放在第三行；如双字节指令“MOV A,direct”，该指令的第一字节是 E5H，第二行的 direct 为其第二

字节。以下机器码指令描述类同。

【例 3-1】 已知(R1)= 30H,(R6) = 18H,(30H)= 78H,(60H) = 12H,则执行下列指令后,各指令累加器 A 中的值是多少？并写出各条指令的机器码。

```
MOV  A,@R1
MOV  A,60H
MOV  A,#56H
MOV  A,R6
```

解：指令执行结果如下。

第 1 行：(A)= 78H ;机器码为 E7H。

第 2 行：(A)= 12H ;机器码为 E560H。

第 3 行：(A)= 56H ;机器码为 7456H。

第 4 行：(A)= 18H ;机器码为 EEH。

② 以工作寄存器 R*n* 为目的操作数的指令(3 条),如表 3-2 所示。

表 3-2 以工作寄存器 R*n* 为目的操作数的指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
MOV　R*n*,A	1111 1*rrr*	F8H～FFH	(R*n*)←(A)
MOV　R*n*,direct	1010 1*rrr* direct	A8H～AFH direct	(R*n*)←(direct)
MOV　R*n*,#data	0111 1*rrr* #data	78H～7FH #data	(R*n*)←#data

【例 3-2】 已知(40H)=30H,求执行下面指令后,(R7)和(40H)的值。

```
MOV R7,40H
```

解：指令执行结果如下：

(R7)=30H,(40H)= 30H。

③ 以直接地址 direct 为目的操作数的指令(5 条),如表 3-3 所示。

表 3-3 以直接地址 direct 为目的操作数的指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
MOV　direct,A	1111 0101 direct	F5H direct	(direct)←(A)
MOV　direct,R*n*	1000 1*rrr* direct	88H～8FH direct	(direct)←(R*n*)
MOV　direct1,direct2;	1000 0101 direct2 direct1	85H direct2 direct1	(direct1)←(direct2)

续表

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOV　direct,@R*i*	1000 011*i* direct	86H～87H direct	(direct)←((R*i*))
MOV　direct,＃data	0111 0101 direct ＃data	75H direct ＃data	(direct)←＃data

上述最后一条指令是一条三字节指令，机器码在存储器中的存放顺序是，75H 操作码为第一字节，direct 为第二字节，立即数 ＃data 为第三字节。

【例 3-3】 已知(A)＝21H，(R0)＝ 5AH，(5AH)＝ 0B8H，(3FH)＝ 19H，则执行下列指令后，求目的操作数(68H)的值及相应指令的机器码。

```
MOV 68H,A;
MOV 68H,@R0;
MOV 68H,3FH;
MOV 68H,#56H;
MOV 68H,R0  ;
```

解：指令执行结果如下。

第 1 行：(68H)＝21H；机器码为 F568H。

第 2 行：(68H)＝0B8H；机器码为 8668H。

第 3 行：(68H)＝19H；机器码为 853F68H。

第 4 行：(68H)＝56H；机器码为 756856H。

第 5 行：(68H)＝5AH；机器码为 8868H。

④ 以间接地址@R*i* 为目的操作数的指令(3 条)，如表 3-4 所示。

表 3-4　以间接地址@R*i* 为目的操作数的指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOV　@R*i*，A	1111 011*i*	F6H～F7H	((R*i*))←(A)
MOV　@R*i*，direct	1010 011*i* direct	A6H～A7H direct	(R*i*))←(direct)
MOV　@R*i*，＃data	0111 011*i* ＃data	76H～77H ＃data	((R*i*))←＃data

【例 3-4】 已知(R1)＝ 32H，(32H)＝ 82H，(40H)＝ 90H，则执行指令后，(R1)、(40H)、(32H)的值是多少？

```
MOV @R1,40H
```

解：指令执行结果如下：

(R1)＝32H，(40H)＝ 90H，(32H)＝90H。

⑤ 16 位数据传送指令(1 条),如表 3-5 所示。

表 3-5　16 位数据传送指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOV　DPTR,#data16	1001 0000 高位字节 低位字节	90H 高位字节 低位字节	(DPTR)←#data16

这是 8 位 MCS-51 单片机指令中唯一的一条 16 位数据传送指令,其功能是把 16 位常数送入 DPTR。DPTR 由 DPH 和 DPL 组成。这条指令执行的结果是,将高 8 位立即数送入 DPH,低 8 位立即数送入 DPL。在译成机器码时,也是高位字节在前,低位字节在后。

例如:

```
MOV DPTR,#56A1H
```

的机器码是 90 56 A1H。

【例 3-5】 已知(30H)=40H,(40H)= 10H,(10H)=08H,(P1)= 0CAH 有如下程序,则程序执行后(R1)、(A)、(B)、(P2)、(40H)、(30H)的值是多少?

```
MOV R1,#30H
MOV A,@R1
MOV R1,A
MOV B,@R1

MOV @R1,P1
MOV P2,P1
MOV 10H,#20H
MOV 30H,10H
```

解:指令执行结果如下:

(A)=(R1)= 40H,(B)= 10H,(40H)= 0CAH,(P2)= 0CAH,(30H)= 20H。

(2) 片外 RAM 数据传送指令(4 条),如表 3-6 所示。

表 3-6　片外 RAM 数据传送指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOVX　A,@DPTR	1110 0000	E0H	(A)←((DPTR))
MOVX　@DPTR,A	1111 0000		((DPTR))←(A)
MOVX　A,@R*i*	1110 001*i*	E2H～E3H	(A)←((R*i*))
MOVX　@R*i*,A	1111 001*i*		((R*i*))←(A)

访问片外数据存储器只能采用间接寻址方式,而且无论读写都需要借助于片内寄存器 A、间接寻址寄存器 DPTR 或 R*i*,且片外 RAM 可读写,因此共有上述 4 条指令。

【例 3-6】 编程把外部 RAM 的 0FAH 单元的内容传送到片内 RAM 的 30H 单元。

方法 1(用 DPTR 作为地址指针)：

```
MOV   DPTR,#0FAH        ;设置地址指针
MOVX  A,@DPTR           ;取出 0FAH 单元的内容送累加器 A 中
MOV   30H,A             ;把 A 中的内容即 0FAH 单元的内容送 30H 单元
```

方法 2(用 R0 作为地址指针)：

```
MOV  R0,#0FAH           ;设置地址指针的低 8 位
MOV  P2,#00H            ;送出源指针的高 8 位
MOVX A,@R0              ;取出 0FAH 单元的内容送累加器 A 中
MOV  30H,A              ;把 A 中的内容即 0FAH 单元的内容送 30H 单元
```

(3) 片内外程序存储器访问指令(2 条)。片内外程序存储器访问指令又称为查表指令,如表 3-7 所示。

表 3-7　片内外程序存储器访问指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MOVC　A,@A+DPTR	1001 0011	93H	(A)←((A)+(DPTR))
MOVC　A,@A+PC	1000 0011	83H	(A)←((A)+(PC))

助记符 MOVC 表示指令要访问的是程序存储区 ROM。这两条指令均采用变址寻址方式,以 DPTR 或本条指令执行完以后的 PC 值作基地址,累加器 A 的内容作偏移量,它们的和就是要访问的程序存储器单元的地址。

【例 3-7】 在 ROM 区从 0800H 开始的 15 个单元中依次存放着 1～15 的平方数表,要求查表求工作寄存器 R1 的内容(其值不超过 15)的平方值,并回送给 R1。

程序如下：

```
MOV  DPTR,#800H         ;设置地址指针
MOV  A,R1
MOVC A,@A+DPTR          ;查表
MOV  R1,A
```

需要指出的是,若用 PC 作基地址,PC 是由指令在程序中的位置确定的,不可随意更改,否则会影响程序的正常执行。

(4) 堆栈操作指令(2 条),如表 3-8 所示。

表 3-8　堆栈操作指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
PUSH　direct	1100 0000 direct	C0H direct	(SP)←(SP)+1 ((SP))←(direct)
POP　direct	1101 0000 direct	D0H direct	(direct)←((SP)) (SP)←(SP)−1

注意：堆栈操作指令的操作数只采用直接寻址方式。

【例 3-8】 设(SP)＝32H，内部 RAM 的 30H～32H 单元的内容分别为 20H、23H、01H。执行下列指令：

```
POP DPH    ;((SP))=(32H)=01H→(DPH),(SP)-1→(SP),
           ;即把 32H 堆栈单元的内容 01H 送 DPH,SP 内容减 1
           ;当前值为 (DPH)=01H,(SP)=31H
POP DPL    ;((SP))=(31H)=23H→(DPL),(SP)-1→(SP)
           ;即把 31H 堆栈单元的内容 23H 送 DPL,SP 内容减 1
           ;当前值: (DPL)=23H,(SP)=30H
POP SP     ;(SP)-1→(SP),((SP))=(30H)=20H→(SP)
           ;先把 SP 内容减 1,后将原 SP 内容 20H 送 SP
```

执行结果为(DPTR)＝0123H，(SP)＝20H。

(5) 数据交换指令(5 条)。

① 字节交换指令(3 条)，如表 3-9 所示。

表 3-9 字节交换指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
XCH　A，R*n*	11001*rrr*	C8H～CFH	(A)↔(R*n*)
XCH　A，direct	11000101 direct	C5H direct	(A)↔(direct)
XCH　A，@R*i*	1100011*i*	C6H～C7H	(A)↔((R*i*))

这 3 条指令为累加器 A 的内容与源操作数的内容互换。

② 半字节交换指令(1 条)，如表 3-10 所示。

表 3-10 半字节交换指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
XCHD　A，@R*i*	1101011*i*	D6H～D7H	(A_3～A_0)↔((Ri_3～Ri_0))

本条指令是累加器 A 内容的低 4 位与源操作数内容的低 4 位互换，高 4 位不变。

③ 累加器 A 的低 4 位与高 4 位互换指令(1 条)，如表 3-11 所示。

表 3-11 累加器 A 的低 4 位与高 4 位互换指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
SWAP　A	11000100	C4H	(A_3～A_0)↔(A_7～A_4)

【例 3-9】 编程将片外 RAM 地址为 1000H 单元的内容与片内 RAM 地址为 60H 单元的内容互换。

程序如下：

```
MOV  DPTR,#1000H      ;设置地址指针
MOVX A,@DPTR          ;读取外部 RAM 的 1000H 单元的值
XCH  A,60H            ;交换
MOVX @DPTR,A          ;60H 单元内容写回片外 RAM 的 1000H 单元
```

3.3.2 算术运算类指令

1. 概述

算术运算类指令主要是用于实现对 8 位无符号数进行加、减、乘、除等算术运算；借助 OV 溢出标志位，可对有符号数进行补码运算；借助 CY 进位标志位，可进行多精度（字节）加、减运算；借助十进制调整指令可以对压缩 BCD 数进行运算。此外，还可以实现加 1、减 1 运算。

算术运算指令都会影响程序状态寄存器 PSW 中的有关标志位，如进位标志位 CY 位、溢出标志位 OV 位、辅助进位位 AC 位和奇偶校验位 P 等。

MCS-51 单片机算术运算类指令共有 24 条，有 8 种操作码助记符（以下简称助记符），根据功能不同可以将其分为 6 类。

（1）加法指令助记符：ADD、ADDC。

（2）减法指令助记符：SUBB。

（3）加 1 和减 1 指令助记符：INC、DEC。

（4）乘法指令助记符：MUL。

（5）除法指令助记符：DIV。

（6）十进制调整指令助记符：DA。

在 MCS-51 单片机的算术运算指令中，加、减、乘、除这 4 种指令的目的操作数有个共同特征：目的操作数只有一种具体形式，即累加器 A。

根据指令助记符的功能分类，24 条算术运算指令可以分为 6 类。

（1）加法指令（8 条）。

（2）减法指令（4 条）。

（3）加 1 和减 1 指令（9 条）。

（4）乘法指令（1 条）。

（5）除法指令（1 条）。

（6）十进制调整指令（1 条）。

2. 指令详解

（1）加法指令（8 条）。加法指令分为不带进位加法指令和带进位加法指令两种。

① 不带进位加法指令（4 条），如表 3-12 所示。

表 3-12　不带进位加法指令

汇编语言格式	机器码格式	十六进制机器码格式	操　作
ADD　A,R*n*	0010 1*rrr*	28H～2FH	(A)←(A)＋(R*n*)
ADD　A，direct	0010 0101 direct	25H direct	(A)←(A)＋(direct)

续表

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
ADD　A，@R*i*	0010 011*i*	26H～27H	(A)←(A)＋((R*i*))
ADD　A，＃data	0010 0100 ＃data	24H ＃data	(A)←(A)＋＃data

这 4 条指令使得累加器 A 可以和内部 RAM 的任何单元的内容进行相加，也可以和一个 8 位立即数相加，相加结果存放在 A 中。无论是哪一条加法指令，参加运算的都是两个 8 位二进制数。对于指令的使用者来说，这些 8 位二进制数既可以看作无符号数(0～255)，也可看作带符号数，即补码数(－128～127)，这完全由使用者事先设定。但计算机在进行加法运算时，总是按以下规定进行：在求和时，总是把操作数直接相加，而无须进行任何变换。例如，假设相加的结果为 1011 1011，若认为是无符号数相加，则这个结果代表十进制数 187；若认为是带符号数相加，则它代表十进制数－69；在相加过程中，当 D6 到 D7 有进位，而 D7 到 CY 无进位；或者 D6 到 D7 无进位，但 D7 到 CY 有进位时，均使溢出标志位 OV＝1。

假定使用者处理的是无符号数，那么 8 位二进制数表示的范围为 00H～FFH(即 0～255)，超过此范围就溢出。在这种情况下，采用(CY)＝1 表示数据有溢出。如果使用者处理的是带符号数，则数的 D7 位用作符号位，数的表示范围为－128～127。这种情况下，数据溢出与否不能用 CY 表示，而要用 OV 表示，而且以(OV)＝1 表示数据溢出。加法指令还会影响辅助进位标志位 AC 和奇偶标志位 P。在相加过程中，当 D3 位产生对 D4 的进位时，(AC)＝1。如果相加后 A 中 1 的个数为偶数，则(P)＝0，否则(P)＝1。

【例 3-10】 已知(A)＝0C3H，(R0)＝0AAH。则执行指令

```
ADD A,R0
```

后，A、CY、OV、AC 和 P 的值是多少。

(A)：　1100　0011
＋)(R0)：　1010　1010
1　0110　1101

结果为(A)＝6DH，(CY)＝1，(OV)＝1，(AC)＝0，(P)＝1。

② 带进位加法指令(4 条)，如表 3-13 所示。

表 3-13　带进位加法指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
ADDC A，R*n*	0011 1*rrr*	38H～3FH	(A)←(A)＋(R*n*)＋(CY)
ADDC　A，direct	0011 0101 direct	35H direct	(A)←(A)＋(direct)＋(CY)
ADDC　A，@R*i*	0011 011*i*	36H～37H	(A)←(A)＋((R*i*))＋(CY)
ADDC　A，＃data	0011 0100 ＃data	34H ＃data	(A)←(A)＋＃data＋(CY)

这 4 条指令的操作，除了指令中所规定的两个操作数相加外，还要加上进位标志位 CY 的值。带进位加法指令主要用于多字节加法。

【例 3-11】 30H、31H 单元和 40H、41H 单元各存放一双字节数（低位在前，高位在后），编程实现这两个双字节数的相加，和保存在 30H 和 31H 单元中（设和仍为双字节数）。

程序如下：

```
MOV  A,30H
ADD  A,40H      ;低字节求和
MOV  30H,A
MOV  A,31H
ADDC A,41H      ;高字节求和
MOV  31H,A
```

（2）减法指令（4 条）。MCS-51 系统的减法指令只有带借位减，共有 4 条指令，如表 3-14 所示。

表 3-14　减法指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
SUBB A,R*n*	1001 1*rrr*	98H～9FH	(A)←(A)－(CY)－(R*n*)
SUBB A,direct	1001 0101 direct	95H direct	(A)←(A)－(CY)－(direct)
SUBB A,@R*i*	1001 011*i*	96H～97H	(A)←(A)－(CY)－((R*i*))
SUBB A,＃data	1001 0100 ＃data	94H ＃data	(A)←(A)－(CY)－＃data

因为在减法操作中必须减去 CY，所以在不需要 CY 参与运算时应先把 CY 清“0”，指令为 CLR C，这条指令属于位操作指令，将在后续小节进行介绍。两数相减时，如果 D7 有借位，则 CY 置“1”；否则清“0”。若 D3 位有借位，则 AC 置“1”；否则清“0”。两数看作带符号数相减时，还要考查 OV 标志。标志位 OV 位的判断标准如下。

正数减正数或负数减负数都不可能溢出，故一定有 OV＝0。

若正数减负数，差值为负（符号位为 1），则一定溢出，故 OV＝1。

若负数减正数，差值为正（符号位为 0），也一定溢出，使 OV＝1。

【例 3-12】 被减数存于 40H、41H 单元，减数存于 30H、31H 单元，编程实现两个数的相减，要求差存于 40H、41H 单元中（低位在前，高位在后）。

程序如下：

```
CLR  C
MOV  A,40H
SUBB A,30H    ;低字节求差
MOV  40H,A
MOV  A,41H
```

```
SUBB    A,31H          ;高字节求差
MOV     41H,A
```

(3) 加 1 和减 1 指令(9 条)。

① 加 1 指令(5 条),如表 3-15 所示。

表 3-15　加 1 指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
INC　A	0000 0100	04H	(A)←(A)+1
INC　R*n*	0000 1*rrr*	08H～0FH	(R*n*)←(R*n*)+1
INC　direct	0000 0101 direct	05H direct	(direct)←(direct)+1
INC　@R*i*	0000 011*i*	06H～07H	((R*i*))←((R*i*))+1
INC　DPTR	1010 0011	A3H	(DPTR)←(DPTR)+1

这组指令的功能是将操作数所指定的单元内容加 1,加法按无符号数二进制进行。除 INC A 指令会影响奇偶标志位 P 外,其余加 1 指令均不影响各个标志位。注意,当用本指令使并行 I/O 接口的内容加 1 时,原始值从 I/O 接口的数据锁存器读入,而不是从 I/O 接口的引脚读入。

② 减 1 指令(4 条),如表 3-16 所示。

表 3-16　减 1 指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
DEC A	0001 0100	14H	(A)←(A)−1
DEC R*n*	0001 1*rrr*	18H～1FH	(R*n*)←(R*n*)−1
DEC direct	0001 0101 direct	15H direct	(direct)←(direct)−1
DEC @R*i*	0001 011*i*	16H～17H	((R*i*))←((R*i*))−1

这组指令的功能是将操作数所指定的单元内容减 1,与加 1 指令一样,除 DEC A 指令会影响奇偶标志位 P 外,其余减 1 指令均不影响各个标志位。注意,执行并行 I/O 接口内容的减 1 操作,是将该口的锁存器内容读出减 1,再写入该锁存器,而不是对该 I/O 接口引脚上的内容进行减 1 操作。

(4) 乘法指令(1 条),如表 3-17 所示。

表 3-17　乘法指令

汇编语言格式	机器码格式	十六进制机器码格式	操　　作
MUL　AB	1010 0100	A4H	$\left.\begin{matrix}(A)_{7\sim0}\\(B)_{15\sim8}\end{matrix}\right\}$←(A)×(B)