第3章 MCS-51指令系统

3.1 指令系统概述

3.1.1 指令与指令系统

微型计算机的功能是从外部世界接收信息，经CPU加工、处理，然后把结果送到计算机外部。设计一台计算机，首先要提供一套具有特定功能的操作命令，这种操作命令叫做指令。CPU所能执行的各种指令的集合称为指令系统。设计一种微处理器，一般从设计指令开始。指令系统因机种不同而异。例如01001111(4FH)代码，对Z80 CPU是完成累加器A中内容传送给寄存器C；对于M6800 CPU是完成累加器A清零操作；而对MCS-51单片机却是完成累加器A和工作寄存器R7的“与”运算。

一种机器的指令系统是该机器本身所固有的，用户无法改变，只能接受应用它。虽然各种机器指令系统各不相同，但它们的指令类型、指令格式、指令基本操作以及指令寻址方式都有很多共同之处。因此，学习好一种机器的指令系统，再学习掌握其他机器指令系统就容易了。

3.1.2 程序与程序设计

计算机完成一项工作，必须按要求去顺序地执行各种操作，即一步步地执行一条条指令，这些按预定要求编排的指令序列称为程序。编排程序的过程叫做程序设计。

程序必须存放在存储器中，CPU逐条取出指令并执行之，从而完成预定任务。

例3-1 在程序存储器中存放了一个ASCII码表，通过查表，将十六进制数转换成ASCII码。设十六进制数存放在R0中的低4位，要求将转换后的ASCII码送回R0中(用MCS-51指令)。

如图 3-1 所示，程序和数据表格已存放在存储器中，都是以二进制数的形式存放的，带“·”的地址中存的是指令的操作码，这些操作码规定了机器执行什么操作；程序中还有指令的操作数，是指令的操作对象，地址 3008H～3017H 单元中存的是(0～F)的 ASCII 码。在微机应用中，大量的工作是编写程序。程序设计过程就是根据任务要求和算法，从指令系统中选取合适的指令，给出必须的操作数(或操作数地址)，加以合理的排列而得到程序的一个过程。

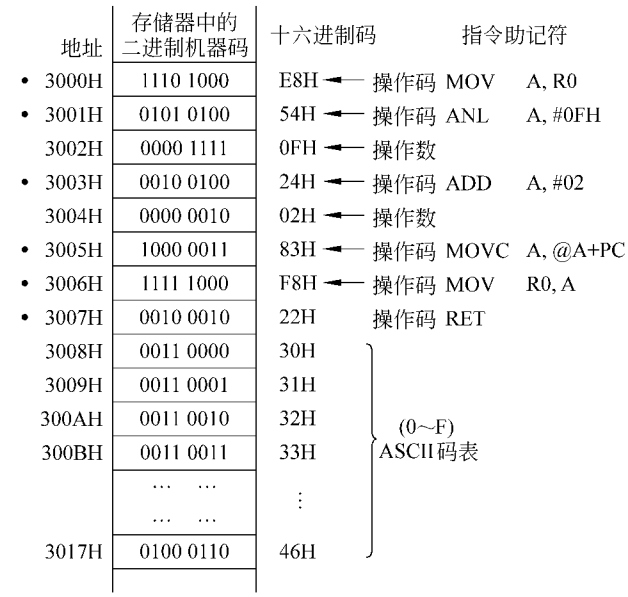


图 3-1　程序和表格已存入存储器

3.1.3　汇编语言

在例 3-1 中的二进制代码叫做指令代码。由于计算机的 CPU 只能认识和识别二进制代码，所以又称为机器码。一种计算机有几十种甚至上百种指令，若都是用二进制码表示，是很困难的。二进制代码，如果用十六进制形式书写，是很方便的，所以，通常用十六进制码表示指令码。但是仍解决不了记忆问题和阅读问题。

为了记忆和阅读方便，制造厂家对指令系统中每一条指令都给出了符号作指令助记符。如图 3-1 所示中第一个操作码“11101000”用 MOV A，R0 表示，“01010100”和“00001111”用 ANL A，#0FH 表示等，就容易记忆，容易理解，清晰可读。

用助记符(操作码)，操作数(或其地址)，标号编写的程序称之为汇编(符号)语言程序。

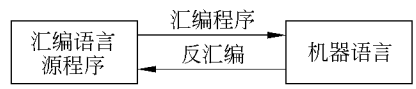
例 3-2　用汇编语言重新书写例 3-1 中的程序如下。

```
标　　号　　操作码　　操作数　　注　　释
          ORG       3000H
ASCCB:    MOV       A,R0          ;取数
          ANL       A,#0FH        ;屏蔽高 4 位
```

```
        ADD     A,#2                    ;变址调整
        MOVC    A,@A+PC                 ;查表
        MOV     R0,A                    ;送结果
        RET                             ;返回主程序
ASCTAB: DB      30H,31H,32H,33H,34H
        DB      35H,36H,37H,38H,39H
        DB      41H,42H,43H,44H,45H,46H
```

从例 3-2 中，可以看出，它与例 3-1 的机器语言程序功能完全相同，但其可理解性，可记忆性，可读性均较机器语言编写的程序要好。然而，无论多么高明的汇编程序，在执行时必须翻译成该机器的指令代码(也叫做目标代码)，才能识别执行。

完成由汇编语言到机器代码的翻译有两种方法。其一是当编好源程序后，根据指令表将每一条指令人工翻译成对应的机器代码，向机器输入机器码，这个过程叫做“人工汇编”；其二是把汇编语言编写的源程序直接输入到计算机中，由机器中一个软件将汇编语言源程序翻译成机器代码，这个软件叫汇编程序，这个过程叫自动汇编。在剖析现成产品的程序时，有时还需要把二进制码机器语言翻译成汇编语言，这个过程称为反汇编。



为了完成汇编语言的汇编工作，汇编程序给程序员在程序的格式上作了一些规定。

例 3-2 给出了汇编语言程序的标准格式。一个汇编语言程序有若干行组成，每行包含一条指令。每行分为 4 个区段：

[标号:] 操作码 [操作数] [;注释]

每行除操作码部分是必须的外，其他带中括号的区段是任选项。

在 4 个区段之间要用分隔符分开，标号后接一个冒号，操作码与操作数之间有一空格隔开，注释段用一分号开始。另外，使用汇编语言编写的程序(源程序)不要求每行的各个区段都一一对齐。然而，如果各个区段都对齐了，程序就更清晰、更可读。

1. 标号

标号是该指令的符号地址，可根据需要设定。标号必须以字母开始，以冒号“：”结束，所用字符一般不超过 8 个(视汇编程序版本不同而异)。系统中保留使用的字符或字符组不能用作标号。如各种 SFR 名、硬、伪指令助记符等。一旦某个标号赋给某个语句，则其他语句的操作数可以直接引用该标号，以便寻址或控制程序转移。标号在每条语句中是任选项。

2. 操作码

由指令系统的助记符，伪(软)指令助记符组成。操作码是汇编语言程序每一句所必须的部分，它决定语句的操作性质。操作码与操作数之间用空格分开。

3. 操作数

操作数可以是数字，这些数字可以是二进制数，以 B(binary)结尾；可以是十进制数，以 D(decimal)结尾或无表示；可以是十六进制数，以 H(hexadecimal)结尾。若数字大于 9，则数字应以“0”开头。这些操作数可以是操作数地址，也可以是立即数。操作数区段可

以是标号或寄存器名，用以指示操作数地址。操作数又可分为目的操作数和源操作数两种，两者之间用逗号“,”分开。另外，有些语句中无操作数，只是一个命令。

4. 注释

以分号“;”开头，计算机在汇编时对这部分不予处理，是程序员对指令操作的解释，可有可无。它由任何可以打印的 ASCII 码字符组成。一般用英文或某种简洁的方式解释本行语句的意义。但不一定对每行都加以解释，仅在某些关键行加注释，以便使程序更可读。注释不影响汇编结果，不译成任何机器代码。

编程者(程序员)在编程序时，一定要严格按规定的格式书写程序。

3.1.4　伪指令

汇编程序中提供了一套伪指令，以支持汇编的运行。这些伪指令仅在汇编过程中起控制作用，不产生可执行目标代码，与机器指令代码无一一对应关系，只能被汇编程序识别。汇编后，目标程序中不再出现伪指令，故又称为软指令。

这里介绍一些常用的伪指令。

1. 起点命令 ORG

格式：

```
ORG  ××××H
```

给程序起始地址或数据块的起始地址赋值命令。总是出现在每段源程序或数据块的开始，它指明此语句后面的程序或数据块的起始地址为××××H。在一个源程序中可多次使用 ORG 命令，以规定不同程序段或数据块的起始位置，所规定的地址从小到大，不允许重叠。

例如：

```
            ORG    8000H
START:      MOV    A,#74H
             ⋮
```

表示源程序的入口地址为 8000H，即程序从 8000H 开始执行。

2. 结束命令 END

格式：

```
END
```

汇编程序结束标志，该命令附在一个源程序的结尾。在 END 之后所写的指令，汇编时不予处理，因此一个源程序只能有一个 END 命令。

3. 定义字节命令 DB

格式：

```
标号：DB  字节常数或字符
```

从指定单元开始，定义了若干个 8 位存储单元，以存放指令给出的数据或字符，字符若用引号括起来，则表示 ASCII 码。

例如：

```
        ORG     8000H
TAB:    DB      45H,73,'5','A'
TAB1:   DB      101B
```

这里数据块的首址由 ORG 命令定义，即 TAB＝8000H，则有：

(8000H)＝45H
(8001H)＝49H
(8002H)＝35H
(8003H)＝41H
(8004H)＝05H

由 DB 命令定义的标号可以任选，DB 所确定的单元地址有两种方法。

① 若 DB 命令是在其他源程序之后，则源程序的最后一条指令地址之后，就是 DB 定义的数据或数据表格。

② 由 ORG 定义数据块首址。

4. 定义字命令 DW

格式：

标号：DW　字或字表

从指定单元开始，定义若干个字(双字节数)。

例如：

```
        ORG         8000H
HETAB:  DW          7234H,8AH,10
```

汇编后则：

(8000H)＝72H
(8001H)＝34H
(8002H)＝00H
(8003H)＝8AH
(8004H)＝00H
(8005H)＝0AH

5. 定义空间命令 DS

格式：

标号：DS　数据或字符表达式

从指定单元开始，由数据或表达式确定保留若干个字节内存空间备用。

例如：

```
ORG     8000H
DS      08H
DB      30H,8AH
```

即 8000H～8007H 单元保留备用

(8008H)＝30H

(8009H)＝8AH

以上 DB、DW、DS 伪指令只对程序存储器起作用。

6. 等值命令 EQU

格式：

```
字符名称  EQU  数据或汇编符号
```

此命令把一个数据或特定的汇编符号赋予标号段规定的字符名称。为“取代”之意，即以数据或汇编符号取代字符名称。用 EQU 定义的字符必须先定义后使用，这些被定义的字符名称可用作数据地址，位地址或立即数。

例如：

```
        ORG     8000H
AA      EQU     R6              ;AA 与 R6 等值
        MOV     A,AA            ;A(R6)
         ⋮
```

7. 数据地址赋值命令 DATA

格式：

```
字符名称  DATA  数据或表达式
```

此命令把数据地址或代码地址赋予标号段规定的字符名称。

例如：

```
INDEXJ    DATA    8389H
```

定义了 INDEXJ 这个字符名称的地址为 8389H，主要用于程序的模块式调试。

例如：

```
          ORG     8000H
INDEXJ    DATA    8096H
          LJMP    INDEXJ
          END
```

等价于

```
ORG       8000H
LJMP      8096H
END
```

被定义的字符名称也可先使用后定义。

DATA 和 EQU 的区别在于用 DATA 定义的字符名称作为标号登记在符号表中，故可先使用后定义；而用 EQU 定义的字符名称必须先定义后使用，其原因是 EQU 不定义在符号表中。

8. 位地址符号命令 BIT

格式：

字符名称 BIT 位地址

该命令把位地址赋予标号段的字符名称。

例如：

```
A1    BIT    P1.0
A2    BIT    P1.1
```

这里位地址 P1.0、P1.1 分别赋给标号段的字符 A1、A2，在编程中可将字符 A1、A2 当作位地址用。

3.1.5 MCS-51 指令系统的特点

MCS-51 指令系统用 42 种助记符表示了 33 种指令功能。有的功能需要几种助记符表示，如数据传送用 MOV、MOVC、MOVX 三种助记符表示。每一种指令对应机器操作码多达 8 种，如 MOV A，Rn 就有 8 种操作码(当 Rn：为 R0～R7 时操作码分别为 E8H～EFH)。这样，MCS-51 指令系统就有 111 条指令，其中单字节指令 49 条，双字节指令 45 条，三字节指令 17 条。在 111 条指令中有 64 条是单周期(12 个振荡周期)指令，45 条是双周期(24 个振荡周期)指令，只有乘、除法指令需 4 个周期(48 个振荡周期)。若晶振为 12MHz 则指令的执行时间分别为 1μs、2μs、4μs。

MCS-51 指令系统在存储空间和执行时间上的效率是比较高的。是一种简明、易掌握、功能强的指令系统。

MCS-51 指令系统中有一个处理布尔变量的指令子集。这个指令子集在设计处理大量位变量程序时十分有效、方便，使 MCS-51 单片机更适合于实时控制，使其指令系统大增特色。

1. 布尔处理机

为了充分地满足工业控制的需要，MCS-51 的设计者在单片机内部设置了功能很强的位处理机，即布尔处理机。

布尔处理机硬件主要由以下几部分支持。

(1) 布尔运算器 ALU。

(2) 布尔累加器 CY(PSW.7)。

(3) 布尔 RAM 区。

片内数据存储器 RAM 20H～2FH 字节(如图 2-15 所示)的 128 位，位地址为 00H～7FH；特殊功能寄存器(直接地址能被 8 整除的 12 个 SFR，如图 2-16 所示)的 93 位(其中 3 位未定义)，位地址分布在 80H～FFH 区间。共有 221 个布尔 RAM 单元构成布尔 RAM 区。

(4) 布尔 I/O 口。

P0～P3 口的每位都可独立地进行输入输出操作，构成布尔 I/O 口。

(5) 布尔指令子集。

由 17 条布尔指令组成，可对各种布尔变量进行处理，如置位、清除、求反、跳转、传送和逻辑运算等。

完善的布尔处理机，提供了最优化程序设计手段，免去了繁琐的数据传送、字节屏蔽、测试分支等操作，可以把复杂的组合逻辑直接转化为 MCS-51 软件，提高了抗干扰能力，加快了运算速度，降低了成本，充分地满足了实时控制的需要。

2. 寻址方式(7 种)

- 立即寻址；
- 直接寻址；
- 寄存器寻址；
- 寄存器间接寻址；
- 基址寄存器加变址寄存器的间接寻址；
- 相对寻址；
- 位寻址。

3. 指令分类(5 类)

- 数据传送(29 条)；
- 算术运算(24 条)；
- 逻辑运算(24 条)；
- 控制转移(17 条)；
- 布尔处理(17 条)。

在分类介绍指令之前，把描述指令的符号意义作一简单介绍。

Rn——当前选中的寄存器区的 8 个工作寄存器 R0～R7(n＝0～7)。

Ri——当前选中的寄存器区中可作间址寄存器的 2 个寄存器 R0、R1(i＝0,1)。

direct——8 位内部数据存储器单元的地址。可以是内部 RAM 单元的地址(0～127/255)或专用特殊功能寄存器 SFR 的地址，如 I/O 端口、控制寄存器、状态寄存器等。

＃data——包含在指令中的 8 位立即数。

＃data16——包含在指令中的 16 位立即数。

addr16——16 位目的地址。用于 LCALL 和 LJMP 指令中，它的地址范围是 64KB 程序存储器地址空间。

addr11——11 位目的地址。用于 ACALL 和 AJMP 指令中，它的地址必须放在与下一条指令的第一个字节同一个 2KB 程序存储器区地址空间之内。

rel——8 位带符号的偏移量(字节)。用于 SJMP 和所有的条件转移指令中。偏移字节相对于下一条指令的第一个字节计算，在－128～＋127 范围内取值。

DPTR——数据地址指针，可用作 16 位间址寄存器。

bit——内部 RAM 或专用寄存器中的直接寻址位。

A——累加器。

B——专用寄存器，用于 MUL 和 DIV 指令中。

C——进位或借位标志，或布尔处理机中的累加器。

@——间址寄存器或基址寄存器的前缀，如@Ri，@A，@DPTR。

/——位操作数的前缀，表示对该位操作数取反，如/bit。

(X)—— X 中的内容。

((X))—— 由 X 寻址的单元中的内容。

← ——箭头左边的内容被箭头右边的内容所代替。

各类指令采用先说明该类指令的共同特征，然后按助记符逐条描述指令，包括助记符、操作码、执行的具体内容及短小应用例子，最后以列表形式小结该类指令。

3.2 MCS-51 指令的寻址方式

指令由操作码与操作数组成，除操作码外，另一组成部分是操作数（包括源操作数和目的操作数），用何种方式寻找参与运算的操作数或操作数的真实地址，被称为指令的寻址方式。

寻址方式的多少，直接反映了机器指令系统功能的强弱，寻址方式越多，其功能越强，灵活性越大。这是衡量计算机性能的重要指标之一。

MCS-51 单片机共有 7 种寻址方式，分别叙述如下。

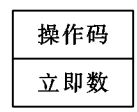
3.2.1 立即寻址

1. 寻址空间

• 程序存储器。

指令的操作数以指令字节的形式存放在程序存储器中。即操作码后紧跟一个被称为立即数（8 位或 16 位）的操作数。是在编程时由程序员给定，并存放在程序存储器中的常数，这种寻址方式称为立即寻址。

2. 指令形式



例如：

```
MOV  A,#30H              ;A←#30H
```

指令的功能是把操作码后面的立即数 30H 送入 A 中，执行过程如图 3-2 所示。

例如：

```
MOV   DPTR,#8000H        ;DPTR←#8000H
```

指令立即数为 16 位，其功能是把立即数高 8 位送入 DPH，低 8 位送入 DPL。指令执行过程如图 3-3 所示。

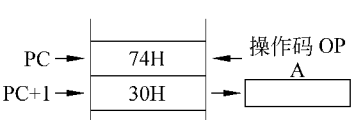


图 3-2 指令 MOV A，#30H 执行过程

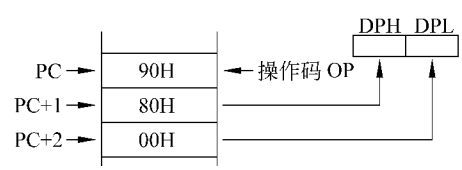


图 3-3 指令 MOV DPTR，#8000H 执行过程

3.2.2　直接寻址

1. 寻址空间

- 内部 RAM 的低 128 字节；
- 特殊功能寄存器 SFR(直接寻址是访问 SFR 的唯一方式)。

操作码后面的一个字节是实际操作数地址。这种直接在指令中给出操作数真实地址的方式称为直接寻址。

2. 指令有三种形式

例如：

```
MOV   A,30H          ;A←(30H)
```

这是数据传送指令，30H 是内部 RAM 地址，功能是把 30H 单元内容读入 A 中，如图 3-4 所示。

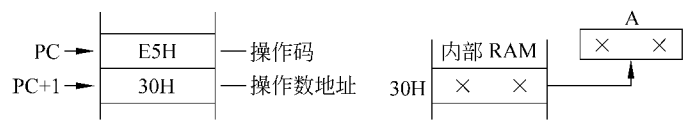


图 3-4　指令 MOV　A,30H 执行过程

例如：

```
ANL   30H,#30H       ;30H←(30H) ∧ #30H
```

这是逻辑“与”操作指令，操作码后面第一个 30H 是操作数地址，第二个 30H 是参加“与”运算的立即数，“与”的结果存入 30H 单元中，执行过程如图 3-5 所示。

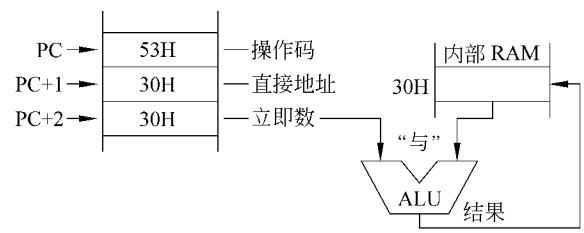


图 3-5　指令 ANL　30H,＃30H 执行过程