第4章 MCS-51汇编语言程序设计

4.1 汇编语言程序设计概述

一、语言的特点

汇编语言：以助记符表示的指令。

特点：

- 助记符指令与机器指令一一对应，运行速度快，占用内存小。
- 直接与计算机硬件打交道，能管理和控制硬件设备。
- 编程比高级语言困难，程序缺乏通用性，不易移植。

二、汇编语言格式

[<标号>]:<操作码> [<操作数>];[<注释>]

1.标号：说明该语句地址的标志符号。由1~8个ASCII字符组成，头一个字符必须是字母。不能使用本汇编语言已经定义的符号。不能在同一程序中使用相同的标号。标号后面用冒号(:)。

2.操作码：规定语句执行的操作内容，以指令助记符或伪指令助记符表示。

3.操作数：给指令提供数据或地址。

4.注释：“;”后面全部为注释，用以对程序进行说明。

5.分隔符：冒号(:)用于标号之后，空格“ ”用于操作码和操作数之间，逗号“,”用于操作数之间，分号“;”用于注释之前。

三、汇编语言程序设计的特定

- 对内存、寄存器等作出具体安排。
- 对硬件结构要了如指掌。
- 软硬件结合，技巧性高。

设计过程：问题分析→确定算法→程序流程→编写程序。

4.2 汇编语言程序的基本结构

3种程序结构：顺序结构、分支结构、循环结构

一、顺序程序：无分支、无循环、不调用子程序。

例：三字节无符号数相加，

被加数在　50H　51H　52H

加数在　53H　54H　55H

和放在　50H　51H　52H

进位放入位地址00H中

```
MOV     R0，#52H          ;被加数的低字节地址
MOV     R1，#55H          ;加数的低字节地址
MOV     A，@R0
ADD     A，@R1            ;低字节相加
MOV     @R0，A            ;存低字节相加结果
DEC     R0
DEC     R1
MOV     A，@R0
ADDC    A，@R1            ;中间字节带进位相加
MOV     @R0，A            ;存中间字节相加结果
DEC     R0
DEC     R1
MOV     A，@R0
ADDC    A，@R1            ;高字节带进位相加
MOV     @R0，A            ;存高字节相加结果
MOV     00H，C
```

二、分支程序

1.单分支程序：使用条件转移指令或位控制转移指令实现。

[例 4.2] 假定在外部 RAM 中有 ST_1、ST_2 和 ST_3 共 3 个连续单元，其中 ST_1 和 ST_2 单元中分别存放着两个 8 位无符号二进制数，要求找出其中的大数并存入 ST_3 单元中。

```
START:  CLR   C              ;进位位清"0"
        MOV   DPTR,#ST1      ;设置数据指针
        MOVX  A,@DPTR        ;取第一个数
        MOV   R2, A          ;第一个数存 R2
        INC   DPTR           ;数据指针加 1
        MOVX  A, @DPTR       ;取第二个数
        SUBB  A, R2          ;两数比较
        JNC   BIG1           ;第二个数大转 BIG1
        XCH   A, R2          ;第一个数大整字节交换继续
BIG0:   INC   DPTR
        MOVX  @DPTR, A       ;存大数
        RET
BIG1:   MOVX  A, @DPTR
        SJMP  BIG0
```

2.多分支程序：

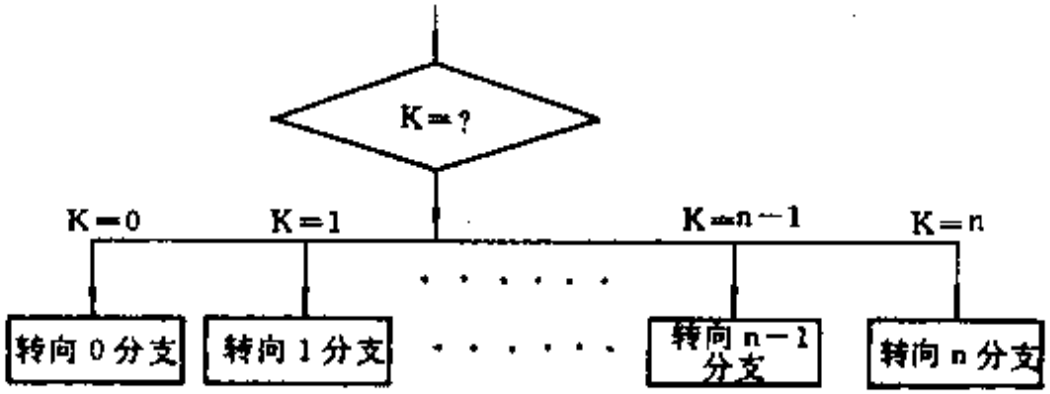

图 4.1　多分支程序转移

4种实现方法：

方法1：使用CJNE，逐条对比分支序号。

[例 4.3]　某温度控制系统，采集的温度值(Ta)放在累加器 A 中。此外，在内部 RAM 54H 单元存放控制温度下限值(T54)，在 55H 单元存放控制温度上限值(T55)。若 Ta＞T55，程序转向 JW(降温处理程序)；若 Ta＜T54，则程序转向 SW(升温处理程序)；若 T55≥Ta≥T54，则程序转向 FH(返回主程序)。有关程序段如下：

```
        CJNE    A,55H,LOOP1     ;Ta≠T55,转向 LOOP1
        AJMP    FH              ;Ta=T55,返回
LOOP1:  JNC     JW              ;若(CY)=0,表明 Ta>T55,转降温处理程序
        CJNE    A,54H,LOOP2     ;Ta≠T54,转向 LOOP2
        AJMP    FH              ;Ta=T54,返回
LOOP2:  JC      SW              ;若(CY)=1,表明 Ta<T54,转升温处理程序
FH:     RET                     ;T55≥Ta≥T54,返回主程序
```

方法2：使用查地址表方法

[例 4.4] 有 BR0、BR1、BR2 和 BR3 共 4 个分支程序段，各分支程序段的功能依次是从内部 RAM 取数、从外部 RAM 低 256B 范围取数、从外部 RAM 4 KB 范围取数和从外部 RAM 64 KB 范围取数。并假定 R_0 中存放取数地址低 8 位地址，R_1 中存放高 8 位地址，R_3 中存放分支序号值。假定以 BRTAB 作差值表首地址，BR0_BRTAB～BR3_BRTAB 为差值。

```
        MOV     A，R3                ;分支转移值送 A
        MOV     DPTR，#brtab         ;差值表首址
        MOVC    A，@A+DPTR           ;查表
        JMP     @A+DPTR              ;转移
BRTAB:  DB      BR0_BRTAB            ;差值表
        DB      BR1_BRTAB
        DB      BR2_BRTAB
        DB      BR3_BRTAB

BR0:    MOV     A，@R0               ;从内部 RAM 取数
        SJMP    BRE
BR1:    MOVX    A，@R0               ;从外部 RAM 256B 取数
        SJMP    BRE
BR2:    MOV     A，R1                ;从外部 RAM 4KB 取数
        ANL     A，#0FH              ;高位地址取低 4 位
        ANL     P2，#0F0H            ;清 P2 口低 4 位
        ORL     P2，A                ;发高位地址
        MOVX    A，@R0
BR3:    MOV     DPL，R0              ;从外部 RAM 64KB 取数
        MOV     DPH，R1
        MOVX    A，@DPTR
BRE:    SJMP    $
```

方法3：使用查转移指令表方法

[例 4.5]　假定键盘上有 3 个操作键，功能说明如下表：

键功能	键值	处理程序
读数据	01	DS
写数据	02	XS
插入	03	CR

下面是有关此键盘的译码程序段：

```
        ⋮
        MOV     DPTR,#3000H          ;3000H 为基址
        CLR     C                    ;进位位 CY 清"0"
        RLC     A                    ;A 带进位位循环左移
        JMP     @A+DPTR              ;转操作键处理程序
3000H
3001H
3002H   AJMP    DS                   ;转读数据程序
3003H
3004H   AJMP    XS                   ;转写数据程序
3005H
3006H   AJMP    CR                   ;转插入程序
        ⋮
```

方法4：使用堆栈操作

```
        MOV     DPTR,#brtab      ;分支入口地址表首址
        MOV     A,R3
        RL      A                ;分支转移值乘以2
        MOV     R1,A             ;暂存A值
        INC     A
        MOVC    A,@A+DPTR        ;取低位地址
        PUSH    A                ;低位地址入栈
        MOV     A,R1             ;恢复A值
        MOVC    A,@A+DPTR        ;取高位地址
        PUSH    A                ;高位地址入栈
        RET                      ;分支入口地址装入PC
BRTAB:  DW      BR0              ;分支程序入口地址表
        DW      BR1
        ⋮       ⋮
        DW      BR127
```

三、循环程序

使用条件转移指令控制循环是否继续或结束。

[例 4.7] 把内部 RAM 中起始地址为 data 的数据串传送到外部 RAM 以 buffer 为首地址的区域，直到发现“$”字符的 ASCII 码为止。同时规定数据串的最大长度为 32 个字节。

```
        MOV     R0，#data           ;data 数据区起始地址
        MOV     DPTR，#buffer       ;buffer 数据区起始地址
        MOV     R1，#20H            ;最大数据串长
LOOP:   MOV     A，@R0              ;取数据
        SUBB    A，#24H             ;判是否为“$”符
        JZ      LOOP1
        MOVX    @DPTR，A            ;数据传送
        INC     DPTR
        INC     R0
        DJNZ    R1，LOOP            ;循环控制
LOOP1:  RET                         ;结束
```

4.3 MCS-51单片机汇编语言程序设计举例

一、算术运算程序

1. 不带符号的多个单字节数加法

例如有多个单字节数，依次存放在外部 RAM 21H 开始的连续单元中，要求把计算结果存放在 R_1 和 R_2 中(假定相加的和为二字节数)，其中 R_1 为高位，R_2 为低位。

```
        MOV   R0, #21H        ;设置数据指针
        MOV   R3, #N          ;字节个数
        MOV   R1, #00H        ;和的高位清"0"
        MOV   R2, #00H        ;和的低位清"0"
LOOP:   MOVX  A, @R0          ;取一个加数
        ADD   A, R2           ;单字节数相加
        MOV   R2, A           ;和的低位送 R2
        JNC   LOOP1
        INC   R1              ;有进位则和的高位加 1
LOOP1:  INC   R0              ;指向下一单元
        DJNZ  R3, LOOP
```

2. 不带符号的两个单字节数减法

设有两个N字节无符号数分别存放在内部RAM的单元中，低字节在前，高字节在后，分别由R_0指定被减数单元地址，由R_1指定减数单元地址，其差存放在原被减数单元中。

```
        CLR     C                ;清进位位
        MOV     R2，#N           ;设定字节数
LOOP:   MOV     A，@ R0          ;从低位取被减的一个字节
        SUBB    A，@ R1          ;两数相减
        MOV     @R0，A           ;存字节相减的差
        INC     R0
        INC     R1
        DJNZ    R2，LOOP             ;两数相减完否
        JC      QAZ                  ;最高字节有借位转溢出
        RET
```

基础

二、数值转换程序

1. 十六进制数转换为ASCII码。

[例 4.11]　在内部 RAM 的 hex 单元中存有 2 位十六进制数，试将其转换为 ASCII 码，并存放于 asc 和 asc+1 两个单元中。

主程序(MAIN)：

```
        MOV     SP,#3FH
MAIN:   PUSH    hex             ;十六进制数进栈
        ACALL   HASC            ;调用转换子程序
        POP     asc             ;第一位转换结果送 asc 单元
        MOV     A, hex          ;再取原十六进制数
        SWAP    A               ;高低半字节交换
        PUSH    ACC             ;交换后的十六进制数进栈
        ACALL   HASC
        POP     asc+1           ;第二位转换结果送 asc+1 单元
```

子程序(HASC)：

```
HASC:   DEC     SP              ;跨过断点保护内容
        DEC     SP
        POP     ACC             ;弹出转换数据
        ANL     A，#0FH         ;屏蔽高位
        ADD     A，#7           ;修改变址寄存器内容
        MOVC    A，@A+PC        ;查表
        PUSH    ACC             ;查表结果进栈
        INC     SP              ;修改堆栈指针回到断点保护内容
        INC     SP
        RET
ASCTAB: DB      “0,1,2,3,4,5,6,7”   ;ASCII码表
        DB      “8,9,A,B,C,D,E,F”
```

2. ASCII码转换为十六进制数。

例如把外部 RAM 30H～3FH 单元中的 ASCII 码依次转换为十六进制数，并存入内部 RAM 60H～67H 单元之中。

```
MAIN:   MOV    R0,#30H      ;设置 ASCII 码地址指针
        MOV    R1,#60H      ;设置十六进制数地址指针
        MOV    R7,#08H      ;需拼装的十六进制数字节个数
AB:     ACALL  TRAN         ;调用转换子程序
        SWAP   A            ;A 高低 4 位交换
        MOVX   @R1,A        ;存放外部 RAM
        INC    R0
        ACALL  TRAN         ;调用转换子程序
        XCHD   A,@R1        ;十六进制数拼装
        INC    R0
        INC    R1
        DJNZ   R7,AB        ;继续
HALT:   AJMP   HALT
```

子程序(TRAN)：

```
TRAN:   CLR    C                ;清进位位
        MOVX   A,@R0            ;取 ASCII 码
        SUBB   A,#30H           ;减 30H
        CJNE   A,#0AH,BB
        AJMP   BC
BB:     JC     DONE
BC:     SUBB   A,#07H           ;大于等于 0AH,再减 07H
DONE:   RET                     ;返回
```

三、定时程序

1. 单循环定时

```
        MOV     R5，#TIME
LOOP：  NOP
        NOP
        DJNZ    R5，LOOP
```

2. 较长时间定时，多重循环

```
        MOV     R5，#TIME1
LOOP2： MOV     R4，#TIME2
LOOP1： NOP
        NOP
        DJNZ    R4，LOOP1
        DJNZ    R5，LOOP2
        RET
```

3. 调整定时时间，增加或减少NOP指令

4. 多种定时要求的实现，调用基本延时程序

```
        MOV   R0，#05H          ;5 s 延时
LOOP1:  LCALL DELAY
        DJNZ  R0，LOOP1
        ⋮
        MOV   R0，#0AH          ;10 s 延时
LOOP2:  LCALL DELAY
        DJNZ  R0，LOOP2
        ⋮
        MOV   R0，#14H          ;20 s 延时
LOOP3:  LCALL DELAY
        DJNZ  R0，LOOP3
```

四、查表程序

设4×4键盘的键码处理入口地址为

键码	入口地址
0	RK0
1	RK1
2	RK2
⋮	

```
        MOV     DPTR,#BS        ;子程序入口地址表首址
        RL      A               ;键码值乘以 2
        MOV     R2, A           ;暂存 A
        MOVC    A,@A+DPTR       ;取得入口地址低位
        PUSH    ACC             ;进栈暂存
        MOV     A,R2
        INC     A
        MOVC    A, @A+DPTR      ;取得入口地址高位
        MOV     DPH, A
        POP     DPL
        CLR     A
        JMP     @A+DPTR         ;转向键处理子程序
BS:     DB      RK0L            ;处理子程序入口地址表
        DB      RK0H
        DB      RK1L
        DB      RK1H
        DB      RK2L
        DB      RK2H
        ……
```

4.4 MCS-51汇编语言的伪指令

伪指令：程序员发给汇编程序的命令

1. 起始地址命令：规定目标程序的起始地址。

[<标号:>] ORG <地址>

例如：

```
        ORG   8000H
START:  MOV   A, #00H
          :
```

2. 汇编终止命令：终止源程序的汇编工作。

[<标号:>] END [<表达式>]

3. 赋值命令：给字符名称赋值，可以是**8**位或**16**位二进制数，可作地址或立即数。

<字符名称> EQU <赋值项>

4. 定义字节命令：从指定地址开始，在程序存储器的连续单元中定义字节数据。

[<标号:>] DB <8位数表>

例如：

```
        ORG   8100H
TAB:    DB    0C0H,0F9H,0A4H,0B0H
        :
```

5. 定义数据字命令：从指定地址开始，在程序存储器的连续单元中定义**16**位数据字。

[<标号:>] DW <16位数表>

6. 定义存储区命令：从指定地址开始，在程序存储器中保留指定数目的字节单元作为存储区。

[<标号:>] DS <表达式>

7. 位定义命令：给字符名称赋以位地址。

<字符名称> BIT <位地址>

例如：

AQ BIT P1.0

4.5 单片机汇编语言源程序的编辑和汇编

步骤1，机器编辑：在微型机上使用编辑软件进行源程序的编辑，生成一个由汇编指令和伪指令组成的ASCII码文件。

步骤2，交叉汇编：在微型机上使用汇编程序将汇编语言源程序转换为机器码表示的目标程序。

步骤3，串行传送：使用串行通信，把目标程序传送到单片机，进行程序的调试和运行。

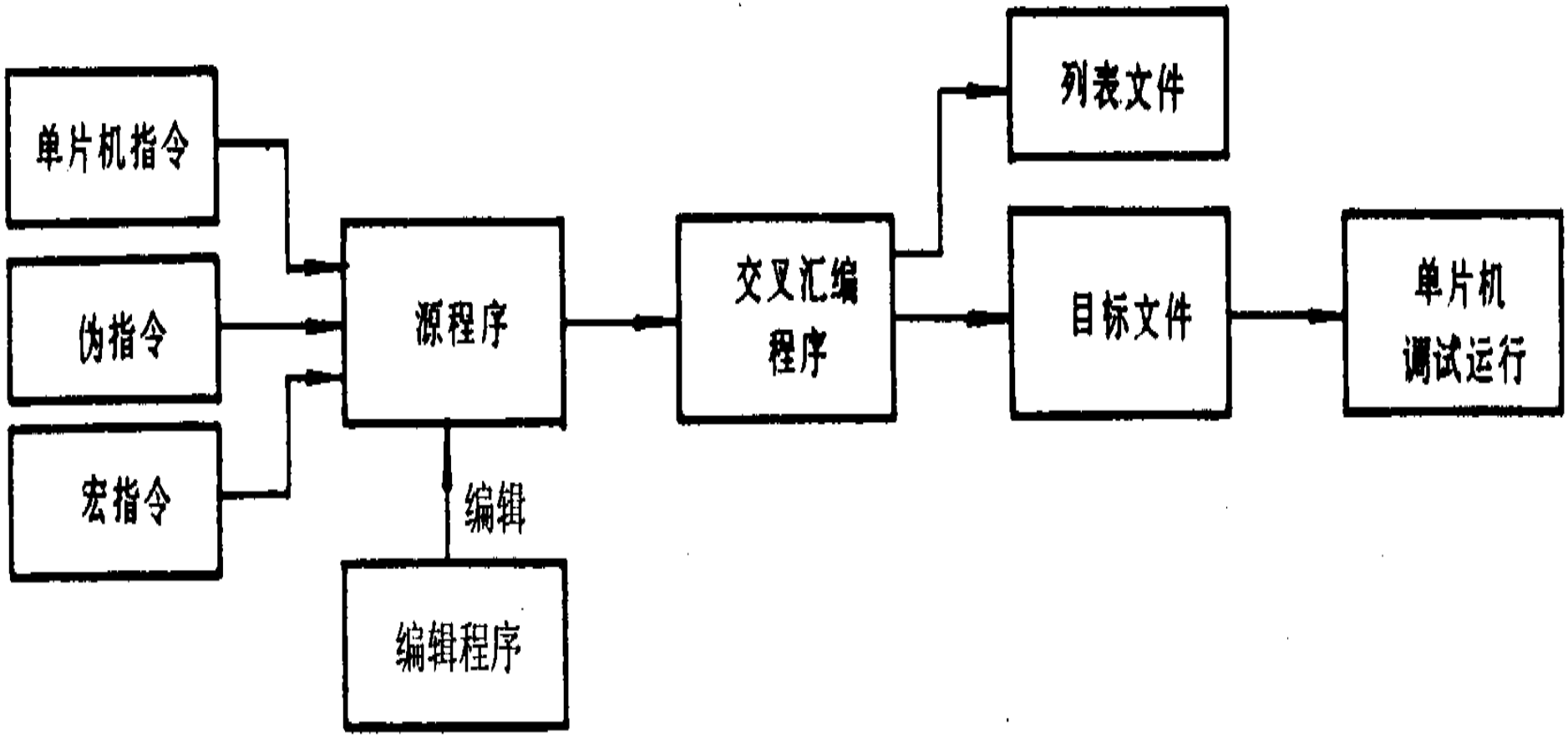

图 4.9　单片机汇编语言程序生成过程