第4章　80C51单片机汇编语言程序设计

4.1　单片机程序设计语言概述及伪指令

4.2　汇编语言程序的基本结构形式

4.3　80C51 单片机汇编语言程序设计举例

4.4　单片机汇编语言源程序的编辑和汇编（不讲）

4.5　例题

4.1 单片机程序设计语言概述

4.1.1 机器语言和汇编语言（低级语言）

机器语言: 用二进制编码表示的指令,是计算机能直接识别并执行的指令。

汇编语言: 用助记符和专门的语言规则表示指令的功能和特征。

汇编语言是对机器语言的改进，比机器语言高级。汇编语言的最大优点是助记符与机器指令一一对应。用汇编语言编写的程序占用存储空间小，运行速度快，程序效率高。

缺点: 难以记忆和使用，程序设计的技巧性较高，编程难度较大。要求使用者必须精通单片机的硬件系统和指令系统。缺乏通用性，程序不易移植。

4.1.2 单片机使用的高级语言

- 对于8051单片机，现有4种语言支持，即汇编、PL/M、C和BASIC。
- C语言最终得到广泛应用。
- 可以大大提高单片机应用系统研制的开发效率。移植性好。
- 高级语言的不足：生成的目标代码较长，导致应用程序运行速度较慢。

伪指令（课本4.5节讲述）

1、汇编语言源程序的格式

标号：操作码 操作数(0~3个)；注释

2.伪指令（所有伪指令在汇编时不产生目标代码，即不是真正的指令）

（1）ORG——起始地址伪指令。格式：ORG　16位地址
功能：规定程序段或数据块的起始地址

（2）END　　结束伪指令。格式：END
功能：用来表示程序结束汇编的位置
该伪指令后边的所有语句将不被汇编成机器码

伪指令（课本4.5节讲述）

（3）EQU——赋值伪指令。　格式: 字符名　EQU　数据或汇编符号

功能: 将该指令右边的值赋给左边的“字符名”

例，AA　EQU　R0

K1　EQU　40

MOV A,AA　;MOV A,R0

MOV A,K1　;（40）→A

（4）DATA——数据赋值伪指令　格式:　标号　DATA　　表达式

功能: 用来将右边表达式的值赋给左边的字符名

例，ORG 0030H

TMP EQU R0

RES DATA　30H

main:　MOV RES ,TMP

DATA 和EQU区别：1、DATA可以先使用再定义它可以放在程序的开头或结尾，也可以放在程序的其它位置，比EQU要灵活。2. EQU伪指令可以把一个汇编符号（如R0)赋给一个字符名称，而DATA伪指令则不能，DATA主要定义数据或地址。

（5）DB——定义字节伪指令; 格式: [标号:]　DB　8位数据或数据表

功能: 在程序存储器中从指定的地址单元开始定义一个或多个字节数据。

ORG　1000H (此语句可省略，由编译器根据程序大小自己分配更好）

例，TAB:DB　34H,0DEH,’A’,’B’,0AH,0BH

（6） DW－－定义字伪指令: 格式: 格式: [标号:] DW 16位数据或数据表

功能: 在程序存储器中从指定的地址单元开始定义一个或多个字数据。

ORG 1000H (此语句可省略，由编译器根据程序大小自己分配更好）

例，TAB:DW 345DH,0DEFFH,'AA','CB'

(7) BIT－－位地址符号指令 格式: 字符名 BIT 位地址

功能: 用来将右边位地址值赋给左边的字符名

例，ABC BIT P1.1

QQ BIT P3.2

(8) DS 定义存储空间伪指令 格式: [标号:] DS 表达式

功能: 用来从指定的地址单元开始留出一定量的字节空间作为备用空间。预留字节单元个数由表达式决定。

例， ORG 1000H

DB 32H,7AH

DS 02H

DW 1234H,58H

4.2 汇编语言程序的基本结构形式

三种基本结构形式: 顺序程序结构，分支程序结构，循环程序结构。

4.2.1 顺序程序结构

顺序结构程序是最简单的程序结构。

程序既无分支、循环，也不调用子程序，程序执行时一条接一条地按顺序执行指令。

例、3字节无符号数相加:
被加数在片内RAM50H（高字节）、
51H和52H（低字节）；
加数在片内RAM的53H（高字节）、
54H和55H （低字节）；
和存放在50H、51H和52H （低字节）
单元中， 进位存放在位寻
址区的20H位中。

```
        ORG 0000H
        LJMP MAIN
        ORG  0030H
MAIN:   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
        CLR A
        ADDC A, #00H;进位送00H位保存
        MOV R0, #20H;存放进位的单元地址
        MOV @R0, A
        SJMP $
        END
```

4.2.2 分支程序结构

分支结构也称为选择结构。

为分支需要，程序设计时应给程序段的起始地址赋予一个地址标号，以供选择分支使用。分支结构又可分为单分支结构和多分支结构。

1. 单分支程序结构

单分支程序结构即二中选一，是通过条件判断实现的。一般都使用条件转移指令对程序的执行结果进行判断

（1）单分支结构举例

假定在外部RAM中有ST1.ST2和ST3共3个连续单元，其中ST1和ST2单元中存放着两个无符号二进制数，要求找出其中的大数并存入ST3单元中。

```
        ORG  0000H
        LJMP  START
        ORG  0030H
START:  CLR C                ;进位位清0
        MOV DPTR，#ST1       ;设置数据指针
        MOVX A，@DPTR        ;取第1个数
        MOV R2, A            ;第1个数存于R2
        INC DPTR             ;数据指针加1
        MOVX A, @DPTR        ;取第2个数
        SUBB A, R2           ;两数比较
        JNC BIG1             ;若第2个数大，则转向BIG1
        XCH A, R2            ;若第1个数大，则整字节交换
BIG0:   INC DPTR
        MOVX @DPTR, A        ;存大数
        RET
BIG1:   MOVX A, @DPTR
```

（2）多重单分支结构举例

多重单分支结构中，通过一系列条件判断，进行逐级分支。为此可使用比较转移指令CJNE实现。

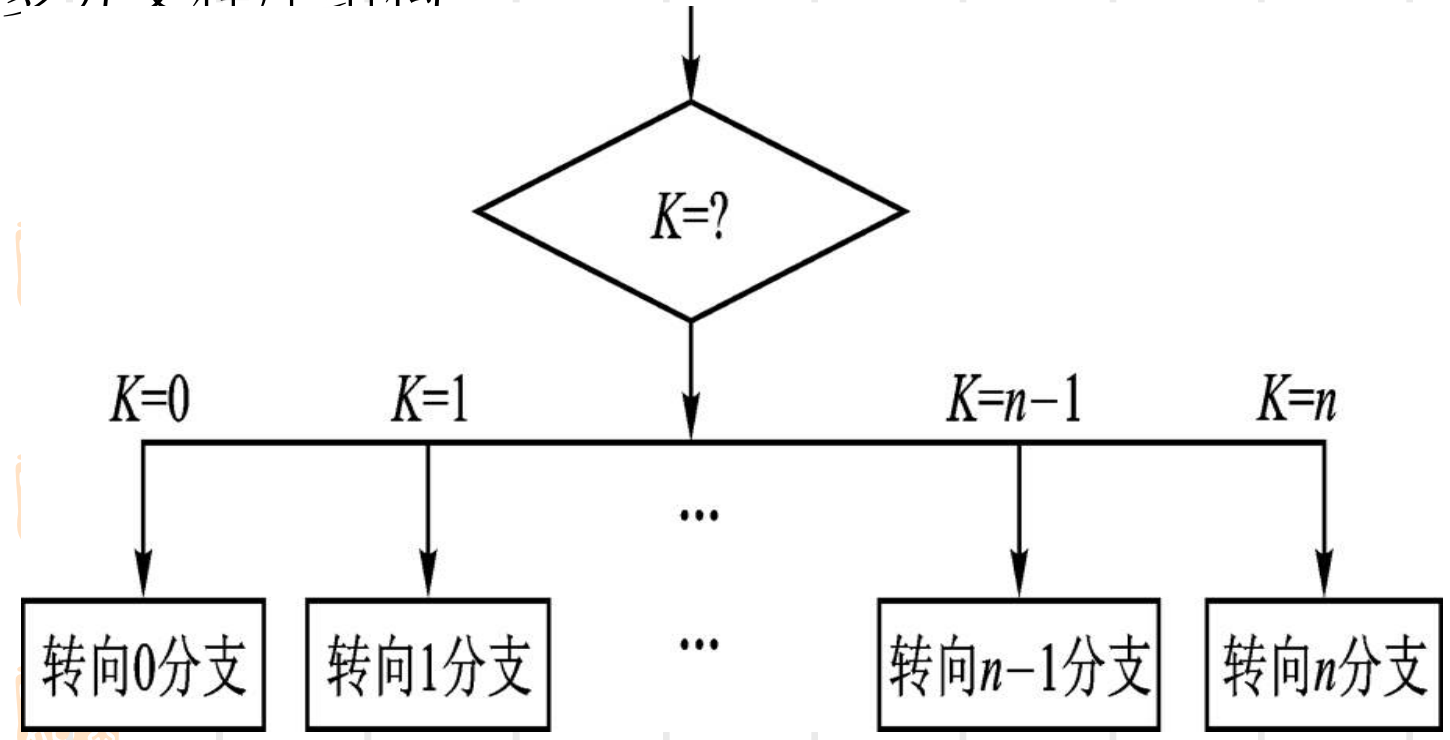
例: 假定采集的温度值Ta放在累加器A中。此外，在内部RAM 54H单元存放温度下限值T54，在55H单元存放温度上限值T55，若Ta＞T55,程序转

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

2. 多分支程序结构

多分支程序结构流程中具有两个以上条件可供选择。可供使用的是变址寻址转移指令“JMP @A+DPTR”，但使用该指令实现多分支转移时，需要有数据表格配合。

多分支程序结构



（1）通过数据表实现程序多分支(不讲)

```
        MOV A, n                ;分支序号送A
        MOV DPTR, #BRTAB        ;地址表首址
        MOVC A, @A+DPTR         ;查表
        JMP @A+DPTR             ;转移
BRTAB:  DB BR0-BRTAB            ;地址表
        DB BR1-BRTAB
 ⋮
        DB BRn-BRTAB
BR0:    …                       分支程序
BR1:    …
 ⋮
BRn:    …
```

（2）通过转移指令表实现程序多分支（不讲）

```
        MOV A, n
        RL A;分支序号值乘以2
        MOV DPTR, #BRTAB;转移指令表首址
        JMP @A+DPTR
BRTAB:  AJMP BR0;转分支程序0
        AJMP BR1;转分支程序1
  ⋮     … …
        AJMP BR127;转分支程序127
```

（3）其他实现程序多分支的方法（不讲）

```
        MOV DPTR, #BRTAB      ;分支入口地址表首址
        MOV A, R0
        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 ACC              ;低位地址入栈
        MOV A, R1             ;恢复A值
        MOVC A, @A+DPTR       ;取高位地址
        PUSH ACC              ;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
```

4.2.3 循环程序结构

循环结构是重复执行某个程序段。

使用条件转移指令通过条件判断来实现和控制循环。

举例: 通过查找结束标志(回车符: ASCII码0DH)以统计字符串长度的循环程序。

```
        MOV R2, #0FFH;设置长度计数器初值
        MOV R0, #3FH;设置字符串指针初值
LOOP:   INC R2
        INC R0
```

4.3 80C51 单片机汇编语言程序设计举例

4.3.1 算术运算程序

1. 加减法运算

（1）两个不带符号的多字节数相减

举例: 将40H开始存放的 10 个字节的数与 50H开始存放的10 个字节的数相减（假设被减数大于减数,低字节在前）。

思路: 设被减数指针为 R0, 减数指针为 R1, 差数放回被减数单元, R5 存放字节个数, 则程序如下:

用keil进行仿真

```
        ORG 0000H
        AJMP  SUB
        ORG 0030H
SUB:    MOV   R0, #40H;
        MOV   R1, #50H
        MOV   R5, #10
        CLR   C
SUB1:   MOV   A, @R0
        SUBB  A, @R1
        MOV   @R0, A
        INC   R0
        INC   R1
        DJNZ  R5, SUB1
        RET
        END
```

（2）两个不带符号的多字节数相减

举例: 设有两个N字节数分别存放在内部RAM单元中，低字节在前，高字节在后，分别由R0指定被减数单元地址，由R1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      ;从低字节开始逐个取被减数字节
        SUBB A, @R1     ;两数相减
        MOV @R0, A      ;存字节相减的差
        INC R0
        INC R1
        DJNZ R2, LOOP;减法是否完成
        JC QAZ          ;若最高字节有借位，则转溢出处理
        RET
```

2. 乘法运算

由于乘法指令“MUL AB”是对单字节的，即单字节数的乘法运算使用一条指令就可以完成；但对多字节数的乘法运算，则必须通过程序实现。

举例：假定要进行两个双字节无符号数乘法运算，被乘数和乘数分别存放于内部RAM的R2、R3单元和R6.R7单元中(其中R2和R6分别为高位字节)，相乘的结果(积）依次存放在R4、R5、R6.R7单