哈爾濱工業大學 远程教育学院

第4章 MCS-51汇编语言程序设计

http://www.hitce.net

概述

汇编语言是面向机器硬件的语言，要求程序设计者对MCS-51单片机具有很好的“软、硬结合”的功底。

介绍程序设计的基本知识及如何使用汇编语言来进行基本的程序设计。

4.1 汇编语言程序设计概述

4.1.1 机器语言、汇编语言和高级语言

用于程序设计的语言基本上分为3种：机器语言、汇编语言和高级语言。

1. 机器语言

二进制代码表示的指令、数字和符号简称为机器语言

不易懂，难记忆，易出错。

2. 汇编语言

英文助记符表示的指令称为符号语言或汇编语言

将汇编语言程序转换成为二进制代码表示的机器语言程序称为汇编程序

经汇编程序“汇编（翻译、编译）”得到的机器语言程序称为目标程序，原来的汇编语言程序称为源程序。

汇编语言特点：

面向机器的语言，程序设计员须对MCS-51的硬件有相当深入的了解。

助记符指令和机器指令一一对应，用汇编语言编写的程序效率高，占用存储空间小，运行速度快，用汇编语言能编写出最优化的程序。

能直接管理和控制硬件设备（功能部件），它能处理中断，也能直接访问存储器及I/O接口电路。

汇编语言和机器语言都脱离不开具体机器的硬件，均是面向“机器”的语言，缺乏通用性。

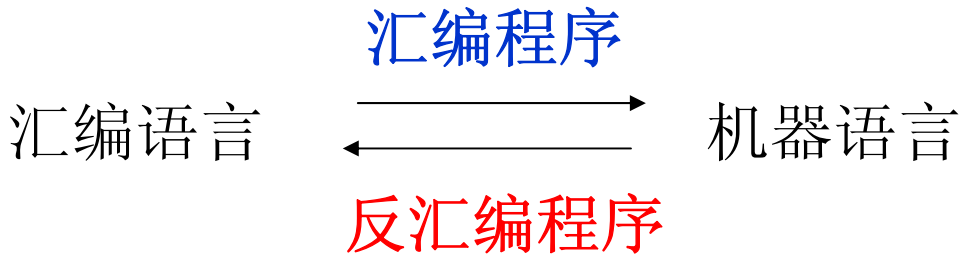
汇编语言与机器语言（机器码）

CPU执行机器语言是由8位二进制代码组成，分为1字节、2字节、3字节。

例如：RET　<=>22H

MOV　A,#0fh　<=>74H　0FH

MOV　74H,#0BH　<=>75H　74H　0BH



3. 高级语言

不受具体机器的限制,使用了许多数学公式和数学计算上的习惯用语，非常擅长于科学计算。常用的如BASIC、FORTRAN以及C语言等。

高级语言优点：通用性强，直观、易懂、易学，可读性好。

使用C语言（C51）、PL/M语言来进行MCS-51的应用程序设计。

对于程序的空间和时间要求很高的场合，汇编语言仍是必不可缺的。

C语言和汇编语言混合编程

在很多需要直接控制硬件的应用场合，则更是非用汇编语言不可

使用汇编语言编程，是单片机程序设计的基本功之一

4.1.2 汇编语言语句的种类和格式

两种基本类型：指令语句和伪指令语句

（1）指令语句

即指令系统，已在第3章介绍（共111条）

每一条指令语句在汇编时都产生一个指令代码——机器代码

（2）伪指令语句

是为汇编服务的，是指示性语句。在汇编时没有机器代码与之对应。

MCS-51的汇编语言的四分段格式如下：

标号字段　　操作码字段　　操作数字段　　注释字段

规则：

（1）标号字段和操作字码段之间要有冒号“：”相隔；

（2）操作码字段和操作数字段间的分界符是空格；

（3）双操作数之间用逗号相隔；

（4）操作数字段和注释字段之间的分界符用分号“；”相隔。

操作码字段为必选项，其余各段为任选项。

不区分大小写

例4-1　下面是一段汇编语言程序的四分段书写格式

标号字段　操作码字段　操作数字段　注释字段

```
START：  MOV    A，#00H     ；0→A
         MOV    R1，#10     ；10→R1
         MOV    R2，#00000011B   ；3→R2
LOOP：   ADD    A，R2       ；（A）+（R2）→A
         DJNZ   R1，LOOP；R1内容减1不
                                为零，则循环
         NOP
HERE：   SJMP   HERE
```

基本语法规则：

START： MOV A，#00H ；0→A

1．标号字段

是语句所在地址的标志符号

（1）标号后边必须跟以冒号“：”

（2）由1~8个ASCII字符组成

（3）同一标号在一个程序中只能定义一次

（4）不能使用汇编语言已经定义的符号作为标号

2．操作码字段

是汇编语言指令中唯一不能空缺的部分。汇编程序就是根据这一字段来生成机器代码的。

3．操作数字段

通常有单操作数、双操作数和无操作数三种情况。如果是双操作数，则操作数之间，要以逗号隔开。

（1）十六进制、二进制和十进制形式的立即数的表示

采用十六进制形式来表示 ，某些特殊场合才采用二进制或十进制的表示形式 。

十六进制，后缀“H” 。

二进制，后缀“B” 。

十进制，后缀“D”，也可省略。

若十六进制的操作数以字符A~F中的某个开头时，则需在它前面加一个 “0”，以便在汇编时把它和字符A~F区别开来。

（2）工作寄存器和特殊功能寄存器的表示

采用工作寄存器和特殊功能寄存器的代号来表示，也可用其地址来表示。

例如，累加器可用A（或Acc）表示。也可用0E0H来表示，0E0H为累加器A的地址。

（3）美元符号$的使用

用于表示该转移指令操作码所在的地址。例如，如下指令：

```
JNB   F0,   $
```

与如下指令是等价的：

```
HERE：JNB    F0，HERE
```

再如：

```
HERE：SJMP    HERE
```

可写为：

```
SJMP    $
```

4. 注释字段

必须以分号“；”开头，换行书写，但必须注意也要以分号“；”开头。

汇编时，注释字段不会产生机器代码。

4.1.3 伪指令

伪指令的作用：在MCS-51 汇编语言源程序中向汇编程序发出的指示信息，告诉它如何完成汇编工作。

也称为汇编程序控制命令。只有在汇编前的源程序中才有伪指令。经过汇编得到目标程序（机器代码）后，伪指令已无存在的必要，所以“伪”体现在汇编时，伪指令没有相应的机器代码产生。

常用的伪指令：

1.ORG（ORiGin）汇编起始地址命令

在汇编语言源程序的开始，通常都用一条ORG伪指令来实现规定程序的起始地址。如不用ORG规定，则汇编得到的目标程序将从0000H开始。例如：

```
        ORG  2000H
START:  MOV  A,#00H
          ⋮
```

规定标号START代表地址为2000H开始。

在一个源程序中，可多次使用ORG指令，来规定不同的程序段的起始地址。但是，地址必须由小到大排列，地址不能交叉、重叠。例如：

```
        ORG  2000H
          ⋮
```

```
ORG   2500H
      ⋮
ORG   3000H
      ⋮
```

2. END(END of assembly)汇编终止命令

汇编语言源程序的结束标志，用于终止源程序的汇编工作。在整个源程序中只能有一条END命令，且位于程序的最后。

3. DB（Define Byte）定义字节命令

在程序存储器的连续单元中定义字节数据。

```
ORG 2000H
DB 30H，40H，24，“C”，“B”
```

汇编后：

（2000H）=30H

（2001H）=40H

（2002H）=18H（10进制数24）

（2003H）=43H（字符“C”的ASCII码）

（2004H）=42H（字符“B”的ASCII码）

DB功能是从指定单元开始定义（存储）若干个字节，10进制数自然转换成16进制数，字母按ASCII码存储。

4. DW（Define Word）定义数据字命令

从指定的地址开始，在程序存储器的连续单元中定义16位的数据字。例如：

```
    ORG 2000H
  DW 1246H, 7BH, 10
```

汇编后：

（2000H）=12H　；第1个字

（2001H）=46H

（2002H）=00H　；第2个字

（2003H）=7BH

（2004H）=00H　；第3个字（2005H）=0AH

（2005H）=0AH

5. EQU（EQUate）赋值命令

用于给标号赋值。赋值以后，其标号值在整个程序有效。例如：

TEST EQU 2000H

表示标号TEST=2000H，在汇编时，凡是遇到标号TEST时，均以2000H来代替。

- EQU指令用于为程序中的任意标号赋值。
- 程序中的常量通常可以用标号代替，便于修改。
- 主程序前要先用equ指令为常量标号赋值。

4.1.4 汇编语言程序设计步骤

（1）明确要求和要达到的目的

（2）确定解决问题的计算方法和步骤

（3）画出流程图

（4）*分配内存地址*

（5）按流程图编写程序

（6）上机汇编、调试、修改直至最后确定源程序

养成在程序的适当位置上加上注释的好习惯。

调试与硬件有关程序还要借助于仿真开发工具并与硬件连接。

4.2 汇编语言源程序的汇编

汇编语言源程序“翻译”成机器代码（指令代码）的过程称为“汇编”。汇编可分为手工汇编和机器汇编两类：

4.2.1 手工汇编

人工查表翻译指令。但遇到的相对转移指令的偏移量的计算，要根据转移的目标地址计算偏移量，不但麻烦，且容易出错。

4.2.2 机器汇编

用编辑软件进行源程序的编辑。编辑完成后，生成一个ASCII码文件，扩展名为“.ASM”。然后在微计算机上运行汇编程序，把汇编语言源程序翻译成机器代码。

汇编后的机器代码是在另一台计算机（这里是单片机）上运行。

MCS-51单片机的应用程序的完成，应经过三个步骤；

（1）在微计算机上，运行编辑程序进行源程序的输入和编辑；

（2）对源程序进行汇编得到机器代码；

（3）通过微计算机的串行口（或并行口）把机器代码传送到用户样机（或在线仿真器）进行程序的调试和运行。

第（1）步，只需在微计算机上使用通用的编辑软件即可完成。

第（2）步所用的汇编程序可在购买单片机的仿真开发工具时，由厂商提供。

第（3）步骤的实现要借助于单片机仿真开发工具进行。

反汇编——分析现成产品的程序，要将二进制的机器代码语言程序翻译成汇编语言源程序。

4.3 汇编语言实用程序设计

4.3.1 汇编语言程序的基本结构形式

常采用以下几种基本结构：

顺序结构、分支结构和循环结构，再加上广泛使用的子程序和中断服务子程序。

1. 顺序结构
2. 分支结构

   程序中含有转移指令

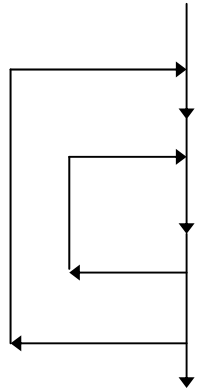
   无条件分支，有条件分支。

   有条件分支又分为：单分支结构和多分支结构。
3. 循环结构
4. 子程序
5. 中断服务子程序

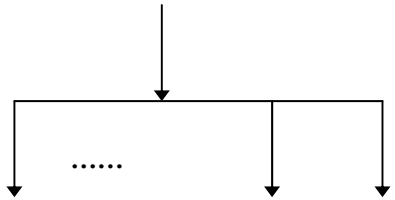
程序结构：

顺序结构

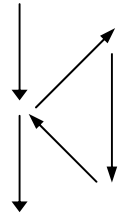
循环结构



分支结构



子程序结构



4.3.2 子程序的设计

一、子程序设计原则和应注意的问题

一种能完成某一特定任务的程序段。其资源要为所有调用程序共享。因此，子程序在结构上应具有独立性和通用性。

在编写子程序时应注意以下问题

1. 子程序的第一条指令的地址称为子程序的入口地址。该指令前必须有标号。
2. 主程序调用子程序

   两条子程序调用指令：

   （1）短调用指令：ACALL addr11

   （2）长调用指令：LCALL addr16
3. 注意设置堆栈指针和现场保护
4. 最后一条指令必须是RET指令
5. 子程序可以嵌套，即子程序可以调用子程序
6. 在子程序调用时，还要注意参数传递的问题

二、 子程序的基本结构

```
MAIN:   ┆                ; MAIN为主程序或调用程序标号
        ┆
      LCALL  SUB  ; 调用子程序SUB
        ┆
        ┆

SUB:      PUSH  PSW      ; 现场保护
          PUSH  ACC      ;
      子程序处理程序段
          POP   ACC      ; 现场恢复
          POP   PSW      ;
          RET            ; 最后一条指令必须为RET
```

4.3.3 查表程序设计

数据补偿、修正、计算、转换等各种功能，具有程序简单、执行速度快等优点。

查表就是根据自变量x，在表格中寻找y，使y=f(x)。

执行查表指令时，发出读程序存储器选通脉冲/PSEN。

在MCS-51的指令系统中，给用户提供了两条极为有用的查表指令：

(1) MOVC A，@A+DPTR

(2) MOVC A，@A+PC

指令“MOVC A，@A+DPTR”

完成把A中的内容作为一个无符号数与DPTR中的内容相加，所得结果为某一程序存储单元的地址，然后把该地址单元中的内容送到累加器A中。

**建议使用该指令

指令“MOVC A，@A+PC”

以PC作为基址寄存器，PC的内容和A的内容作为无符号数，相加后所得的数作为某一程序存储器单元的地址，根据地址取出程序存储器相应单元中的内容送到累加器A中。

指令执行完,PC的内容不发生变化，仍指向查表指令的下一条指令。

优点在于预处理较少且不影响其它特殊功能寄存器的值，所以不必保护其它特殊功能寄存器的原先值。

缺点在于该表格只能存放在这条指令的地址XXXX以下的00～FFH之中。表格所在的程序空间受到了限制。

*初学阶段不建议使用

MOVC A，@+DPTR 这条指令的应用范围较为广泛，一般情况下，大多使用该指令，使用该指令时不必计算偏移量，使用该指令的优点是表格可以设在64K程序存储器空间内的任何地方，而不像 MOVC A，@A+PC那样只设在PC下面的256个单元中，使用较方便。

查表指令的使用

例 1　求0~9的平方

```
        ORG    0000H
         LJMP  A1
        ORG    0080H
A1:     NOP
        NOP
        MOV   SP,#60H
        MOV   DPTR,#2000H
        MOV   A,#03H；将0~9之内任意一个数赋值给A。
        MOVC A,@A+DPTR
A2:     SJMP  A2
        ORG    2000H
        DB    00h,01h,04h,09h,10h,19h,24h,31h,40h,51h
         END
```

例 1　求0~9的平方

```
        ORG   0000H
        LJMP  A1
        ORG   0080H
A1:     NOP
        NOP
        MOV   SP,#60H
        MOV   DPTR,#tab
        MOV   A,#03H；将0~9之内任意一个数赋值给A。
        MOVC A,@A+DPTR
A2:     SJMP  A2
tab:    DB    00h,01h,04h,09h,10h,19h,24h,31h,40h,51h
        END
```

例1：子程序编写的求平方程序：根据累加器A中的数x（0～9之间）查x的平方表y，根据x的值查出相应的平方y。x和y均为单字节数。

```
LLL:  PUSH    DPH              ；保存DPH
      PUSH    DPL              ；保存DPL
      MOV     DPTR，#TAB1
      MOVC    A，@A+DPTR
      POP     DPL              ；恢复DPL
      POP     DPH              ；恢复DPH
      RET
TAB1: DB  00H，01H，04H，09H，10H
      DB  19H，24H，31H，40H，51H
```

4.3.4 关键字查找程序设计

顺序检索和对分检索

一、顺序检索

从第1项开始逐项顺序查找，判断所取数据是否与关键字相等。

例2 从50个字节的无序表中查找一个关键字××”H。

```
        ORG   1000H
    MOV     30H，#××H；关键字××H送30H单元
    MOV     R1，#50      ；查找次数送R1
    MOV     A，#0        ；修正值送A
```

```
        MOV    DPTR，#TAB4   ；表首地址送DPTR
LOOP：PUSH   ACC
LOOP2：MOVC   A，@ A+DPTR   ；查表结果送A
        CJNE   A，30H，LOOP1；（30H）不等于关键
                                字则转LOOP1
        MOV    R2，DPH  ；已查到关键字，把该字
                        ；的地址送R2，R3
        MOV    R3，DPL   ；
Loop3：RET
LOOP1：INC    DPTR        ；修改数据指针DPTR
         CLR    ACC
       DJNZ    R1，LOOP2 ；R1≠0，未查完，继续查找
```

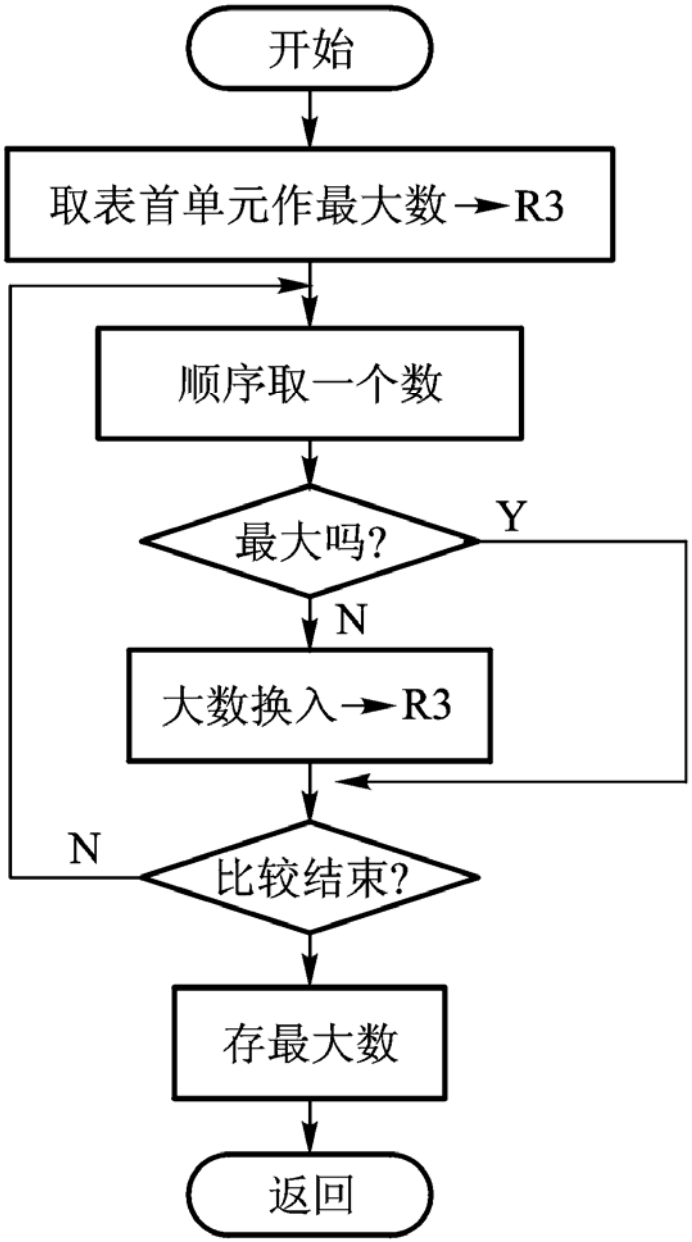
```
        MOV   R2，#00H  ；R1=0，清“0”R2、R3
        MOV   R3，#00H  ；表中50个数已查完
        AJMP Loop3     ；从子程序返回
TAB4：DB …，…，…  ；50个无序数据表
```

4.3.5 数据极值查找程序设计

在指定的数据区中找出最大值（或最小值）。

进行数值大小的比较，从这批数据中找出最大值（或最小值）并存于某一单元中。

例3　片内RAM中存放一批数据，查找出最大值并存放于首地址中。设R0中存首地址，R2中存放字节数，程序框图如下图所示。


开始
取表首单元作最大数→R3
顺序取一个数
最大吗?
Y
N
大数换入→R3
N
比较结束?
存最大数
返回

程序如下：

```
        MOV   R2，n     ；n为要比较的数据字节数
        MOV   A，R0     ；存首地址指针
        MOV   R1，A
        DEC   R2        ；得到比较的次数
        MOV   A，@R1
LOOP:    MOV  R3，  A
         INC  R1
         CLR  C
         SUBB A，@R1    ；两个数比较
         JNC  LOOP1     ；C=0，A中的数大，跳LOOP1
         MOV  A，@R1    ；C=1，则大数送A
         SJMP LOOP2
LOOP1:MOV    A，R3
LOOP2:DJNZ   R2，LOOP  ；是否比较结束？
        MOV   @R0，  A  ；存最大数
        END
```

4.3.6 数据排序程序设计

升序排，降序排。仅介绍无符号数据升序排。

冒泡法：相邻数互换的排序方法，类似水中气泡上浮 。排序时从前向后进行相邻两个数的比较，前面的数大于后面的数时，就将两个数互换；否则不互换。

假设有7个原始数据的排列顺序为：6、4、1、2、5、7、3。第一次冒泡的过程是：

6、4、1、2、5、7、3　　；原始数据的排列

4、6、1、2、5、7、3　　；大于，互换

4、1、6、2、5、7、3　　；大于，互换

4、1、2、6、5、7、3　　；大于，互换

4、1、2、5、6、7、3　　；大于，互换

4、1、2、5、6、7、3 ；小于，不互换

4、1、2、5、6、3、7 ；大于，互换，第一次冒泡结束

如此进行，各次冒泡的结果如下：

第1次冒泡结果：4、1、2、5、6、3、7

第2次冒泡结果：1、2、4、5、3、6、7

第3次冒泡结果：1、2、4、3、5、6、7

第4次冒泡结果：1、2、3、4、5、6、7 ；已完成排序

第5次冒泡结果：1、2、3、4、5、6、7

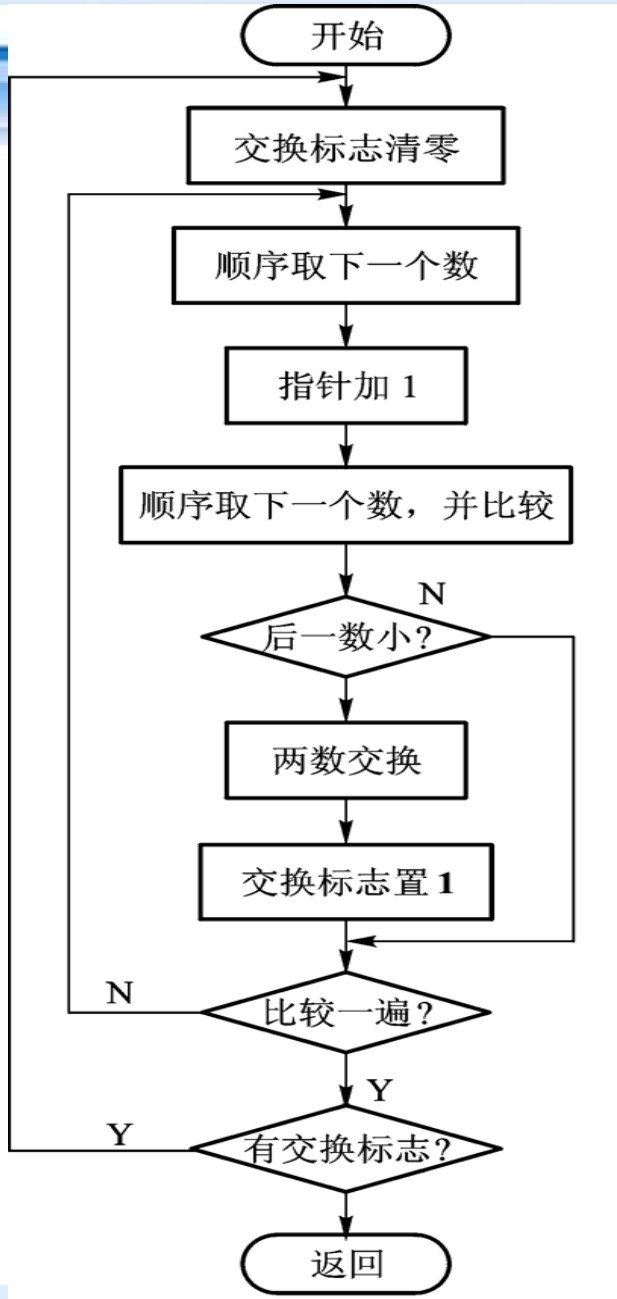
第6次冒泡结果：1、2、3、4、5、6、7

对于n个数，理论上应进行（n-1）次冒泡，有时不到（n-1）次就已完成排序 。

如何判定排序是否已完成，看各次冒泡中是否有互换发生，如果有数据互换，则排序还没完成。

在程序设计中，常使用设置互换标志的方法，该标志的状态表示在一次冒泡中是否有互换进行。

例4　一批单字节无符号数，以R0为首地址指针，R2中为字节数，将这批数进行升序排列。程序框图如下图所示。


开始
交换标志清零
顺序取下一个数
指针加 1
顺序取下一个数，并比较
N
后一数小?
两数交换
交换标志置 1
N
比较一遍?
Y
Y
有交换标志?
返回

```
SORT:   MOV   A，R0      ；取首地址
        MOV   R1，A
        MOV   A，R2      ；字节数送入R5
        MOV   R5，A
        CLR   F0         ；互换标志位F0清零
        DEC   R5         ；
        MOV   A，@R1     ；

LOOP:   MOV   R3，A      ；
        INC   R1         ；
        CLR   C          ；
        MOV   A，@R1     ；比较大小
        SUBB  A，R3      ；
        JNC   LOOP1      ；
        SETB  F0         ；互换标志位F0置1
        MOV   A，R3      ；
        XCH   A，@R1     ；两个数互换
        DEC   R1         ；
        XCH   A，@R1     ；
        INC   R1
```

```
LOOP1：    MOV    A，@R1
           DJNZ   R5，LOOP
           JB     F0，SORT
           RET
```

4.3.7 分支转移程序设计

特点是程序中含有转移指令，转移指令又分为无条件转移和有条件转移，因此分支程序也可分为无条件分支转移程序和有条件分支转移程序。有条件分支转移程序按结构类型来分，又分为单分支转移结构和多分支转移结构。

一、分支转移结构

1. 单分支转移结构

仅有两个出口，两者选一。

例5 求单字节数的二进制补码

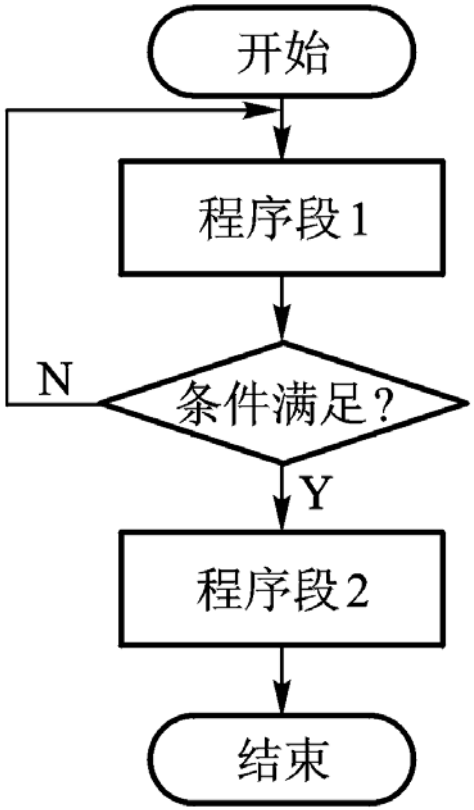
补码：8位二进制数最高位为0，则补码为其自身；

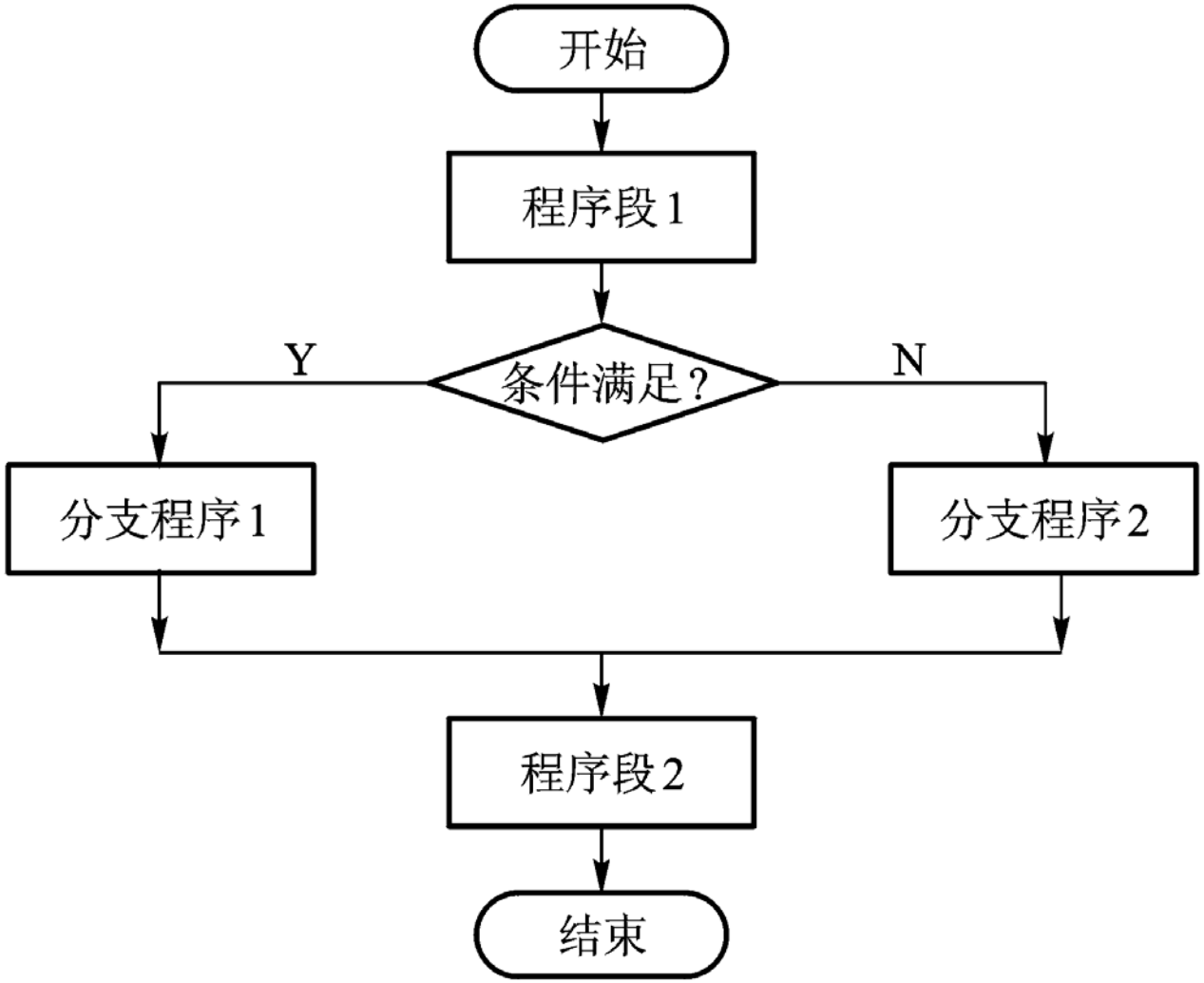
若最高位为1，则其补码为，最高位保持不变，其他各位按位取反后加1。

参考程序：

```
CMPT:    JNB    Acc.7,loop;  (A.7)=0，不需转换
         MOV    C，Acc.7 ；符号位保存
         CPL    A                ；（A）求反，加1
         ADD    A，#1     ；
         MOV  Acc.7，C     ；符号位存A的最高位
loop:    RET
```

此外，单分支选择结构还有下图所示的几种形式：




开始
程序段1
条件满足？
Y
N
分支程序1
分支程序2
程序段2
结束

2. 多分支转移结构

程序的判别部分有两个以上的出口流向。

指令系统提供了非常有用的两种多分支选择指令:

间接转移指令: JMP @A+DPTR;

比较转移指令: CJNE A,direct,rel;

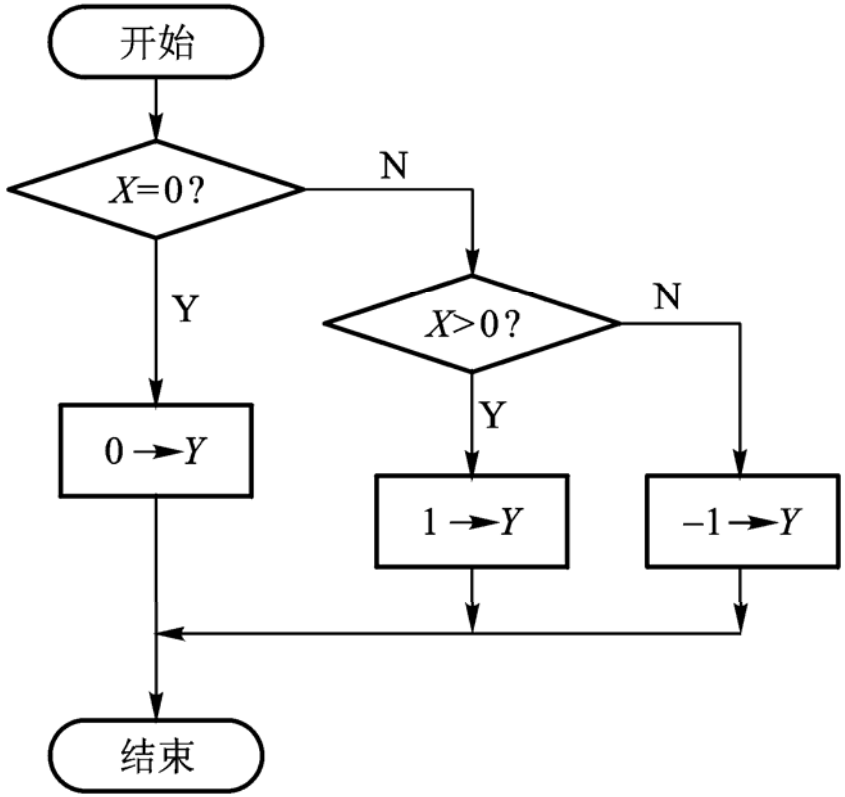
CJNE A,#data,rel;

CJNE Rn,#data,rel;

CJNE @Ri,#data,rel;

最简单的分支转移程序的设计，一般常采用逐次比较法，就是把所有不同的情况一个一个的进行比较，发现符合就转向对应的处理程序。这种方法的主要缺点是程序太长，有n种可能的情况，就需有n个判断和转移。

例求符号函数的值。
程序框图如图4-6所示。



4-6

程序略。

提高：使用转移指令的注意事项

无条件分支程序

- LJMP 指令，跳转的程序可位于程序存储器中任意位置

有条件分支程序

- *根据*已经执行的程序中标志位、ACC或内部RAM的*某些位的结果*决定程序的流向
- JZ/JNZ、CJNE、DJNZ、位控制转移类指令（JC、JNC、JB、JNB、JBC）
- 跳转的程序位置有要求，必须位于当前指令的-128~127范围之内，如果超过该范围需要采取必要的措施

4.3.8 循环程序设计

特点是程序中含有可以反复执行的程序段，该程序段通常称为循环体。例如求100个数的累加和，则没有必要连续安排100条加法指令，可以只用一条加法指令并使其循环执行100次。

（1）可大大缩短程序长度（2）使程序所占的内存单元数量少（3）使程序结构紧凑和可读性变好。

一、循环程序的结构

循环结构程序主要由以下四部分组成。

1. 循环初始化

循环初始化程序段用于完成循环前的的准备工作。例如，循环控制计数初值的设置、地址指针的起始地址的设置、为变量预置初值等。

2．循环处理

循环程序结构的核心部分，完成实际的处理工作，是需反复循环执行的部分，故又称循环体。这部分程序的内容，取决于实际处理问题的本身。

3．循环控制

在重复执行循环体的过程中，不断修改循环控制变量，直到符合结束条件，就结束循环程序的执行。循环结束控制方法分为循环计数控制法和条件控制法

4．循环结束

这部分是对循环程序执行的结果进行分析、处理和存放。

二、循环结构的控制

计数循环结构

计数循环控制结构是依据计数器的值来决定循环次数，一般为减“1”计数器，计数器减到“0”时，结束循环。计数器的初值是在初始化时设定。

MCS-51的指令系统提供了功能极强的循环控制指令：

DJNZ　Rn,rel；工作寄存器作控制计数器

DJNZ　direct,rel；以直接寻址单元作控制计数器。

最常见的多重循环是由DJNZ指令构成的软件延时程序，它是常用的程序之一。

例6　50ms延时程序。

延时程序与MCS-51指令执行时间有很大的关系。在使用12MHz晶振时，一个机器周期为1μs,执行一条DJNZ指令的时间为2μs。这时，可用双重循环方法写出下面如下的延时50ms的程序：

```
DEL:   MOV    R7,#200
DEL1:  MOV    R6,#125
DEL2:  DJNZ   R6,DEL2          ; 125*2=250μs
       DJNZ   R7,DEL1          ; 0.25ms*200=50ms
       RET
```

以上延时程序不太精确，它没有考虑到除“DJNZ　R6,DEL2 ”指令外的其它指令的执行时间。

*一般应用软件延时获得的时间不是很准确。

*软件延时程序，不允许有中断，否则将严重影响定时的准确性。

编写循环嵌套程序的注意事项

允许外重循环嵌套内重循环

循环体不能交叉

不能从循环程序外部跳入循环程序内部

4.3.9 码制转换程序设计

在单片机应用程序的设计中，经常涉及到各种码制的转换问题。在单片机系统内部进行数据计算和存储时，经常采用二进制码，具有运算方便、存储量小的特点。

在输入/输出中，按照人的习惯均采用代表十进制数的BCD码（用4位二进制数表示的十进制数）表示。

一、二进制码到BCD码的转换

BCD码有两种形式：一种是1个字节放1位BCD码，它适用于显示或输出，一种是压缩的BCD码，即1个字节放两位BCD码，可以节省存储单元。

00000000=0　00000001=1

。 。 。 。 。

00001001=9

压缩BCD码

00000000=00　00000001=01

。 。 。 。 。

00011001=19

。 。 。 。 。 。

10011001=99

BCD码与ASCII码对照

0——48　　1——49

2——50　　3——51

4——52　　5——53

6——54　　7——55

8——56　　9——57

例7：将20H单元内的压缩BCD码变换成相应ASCII码放在21H、22H中

```
Org      0h
Ljmp     2000h
ORG      2000H
MOV      R0,#22H
MOV      @R0,#00
MOV      A,20H
XCHD     A,@R0
ORL      22H,#30H
SWAP     A
ORL      A,#30H
MOV      21H,A
z：SJMP z
END
```

15BYTE ,11T

```
MOV A,20H
MOV B,#10H
DIV AB
ORL B,#30H
MOV 22H,B
ORL A,#30H
MOV 21H,A
SJMP $
END
```

15BYTE 14T

谢谢大家！