3.9 程序设计方法

汇编语言程序设计，就是采用汇编指令来编写计算机程序。在实际编程中，如何正确选择指令、寻址方式和合理使用工作寄存器，包括数据存储器单元，如何对扩展的**I/O**端口进行操作等，都是基本的汇编语言程序设计技巧。

3.9.1程序设计的步骤

在编写汇编语言程序时，应遵循程序设计简明，占用内存少，执行时间最短的原则，一般有以下几个过程：

1.分析问题

对单片机应用系统预完成的任务进行深入的分析，明确系统的设计任务、功能要求和技术指标。根据课题要求，用适当的数学方法来描述和建立数学模型。对系统的硬件资源和工作环境进行分析。这是单片机应用系统程序设计的基础和条件。

3.9 程序设计方法

2.确定算法

算法是解决具体问题的方法。应用系统经过分析、研究和明确规定后，对应实现的功能和技术指标可以利用严密的数学方法或数学模型来描述，从而把实际问题转化成由计算机进行处理的问题。了解现有条件和目标要求后再确定解决该问题的方法和步骤。同一个问题的算法可以有多种，结果也可能不尽相同，所以，应对各种算法进行分析比较，并进行合理的优化。比如，用迭代法解微分方程，需要考虑收敛速度的快慢（即在一定的时间里能否达到精度要求）。而有的问题则受内存容量的限制而对时间要求并不苛刻。对于后一种情况，速度不快但节省内存的算法则应是首选。

3.9 程序设计方法

3. 程序总体设计及流程图绘制

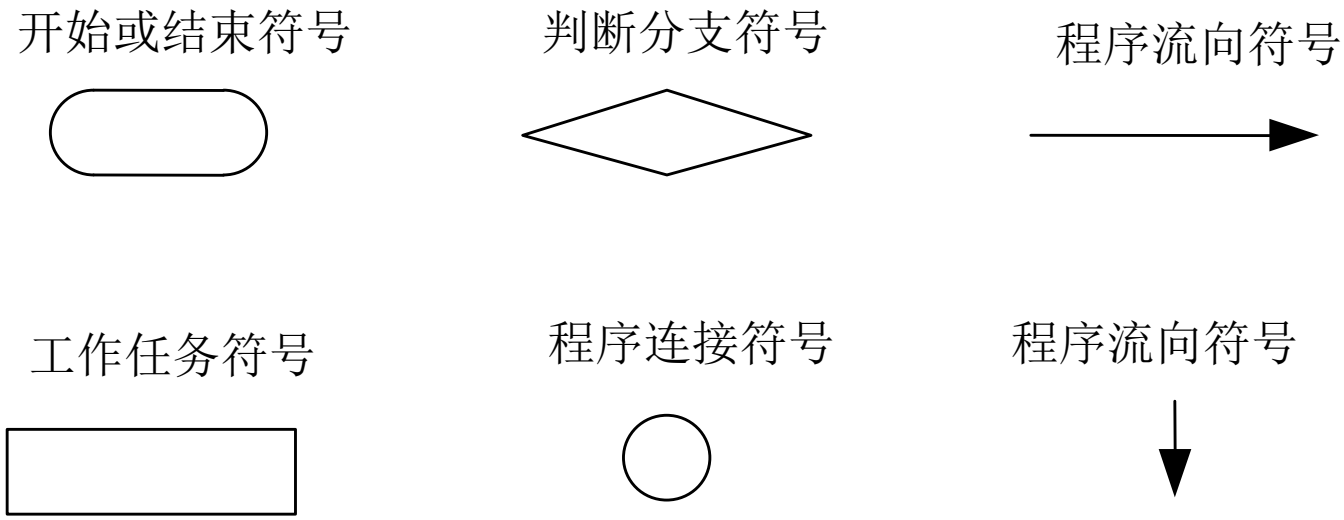
经过任务分析、算法优化后，就可以进行程序的总体构思，确定程序的结构和数据形式，合理选择和分配内存单元、工作寄存器，并考虑资源的分配和参数的计算等。然后根据程序运行的过程，勾画出程序执行的逻辑顺序，用图形符号将总体设计思路及程序流向绘制在平面图上，从而使程序的结构关系直观明了，便于检查和修改。

清晰正确的流程图是编制正确无误的应用程序的基础和条件。所以，绘制一个好的流程图，是程序设计的一项重要内容。

把算法用流程图描述出来，即用流程图中的各种图形、符号、流向线等来描述程序设计的过程，它可以清晰表达程序的设计思路。

3.9 程序设计方法

流程图可以分为总流程图和局部流程图。总流程图侧重反映程序的逻辑结构和各程序模块之间的相互关系。局部流程图反映程序模块的具体实施细节。对于简单的应用程序，可以不画流程图。但是当程序较为复杂时，绘制流程图是一个良好的编程习惯。



程序流程图常用符号

3.9 程序设计方法

4.编写程序

根据流程图中各部分的功能，选取合适的指令，按模块结构具体编写源程序。

5.程序汇编和调试

对已编写好的程序，先进行汇编。在汇编过程中，若还有语法错误，需要对源程序进行修改。汇编及调试程序通过汇编生成目标程序，经过多次上机调试运行，对程序运行结果进行分析，不断修正源程序中的错误，最后得到正确结果，达到预期目的。

6.程序优化

对源程序按结构、长度、时间等方面的指标进行优化。

3.9 程序设计方法

编制程序的方法和技巧

一、采用模块化程序设计方法

应用系统的程序由包含多个模块的主程序和各种子程序组成。各程序模块都要完成一个明确的任务，实现某个具体的功能，如：发送、接收、延时、打印和显示等。

模块化的程序设计方法具有明显的优点。把一个多功能的复杂的程序划分为若干个简单的、功能单一的程序模块，有利于程序的设计和调试，有利于程序的优化和分工，提高了程序的阅读性和可靠性，使程序的结构层次一目了然。

二、尽量采用循环结构和子程序

采用循环结构和子程序可以使程序的长度减少、占用内存空间减少。

3.9 程序设计方法

3.9.2 程序结构

编写一个应用系统的汇编语言源程序，程序结构一般采用以下3种基本控制结构，即顺序结构、分支结构和循环结构来组成，再加上使用广泛的子程序及中断服务子程序，共有5种基本结构，其中子程序及中断服务子程序的基本控制结构也是由顺序结构、分支结构和循环结构组成。

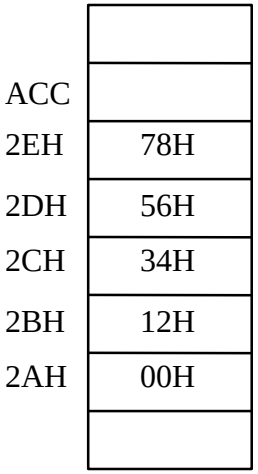
一、顺序程序设计

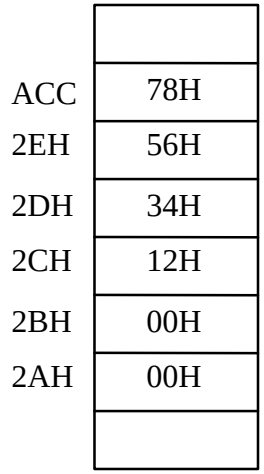
顺序结构程序是最简单、最基本的程序。顺序程序是指无分支、无循环结构的程序。其执行流程是依指令在存储器中的存放顺序进行的。顺序程序按照逻辑操作顺序，从某一条指令开始逐条顺序执行，直至某一条指令为止。

3.9 程序设计方法

顺序程序综合举例：

例3-30 数据传送。内部RAM的2AH～2EH单元中存储的数据如图所示。试编写程序实现图示的数据传送结果。





解：方法一：

```
MOV  A，2EH      ；2字节，1个机器周期
MOV  2EH，2DH    ；3字节，2个机器周期
MOV  2DH，2CH    ；3字节，2个机器周期
MOV  2CH，2BH    ；3字节，2个机器周期
MOV  2BH，#00H   ；3字节，2个机器周期
```

方法二：

```
CLR  A           ；1字节，1个机器周期
XCH  A，2BH      ；2字节，1个机器周期
XCH  A，2CH      ；2字节，1个机器周期
XCH  A，2DH      ；2字节，1个机器周期
XCH  A，2EH      ；2字节，1个机器周期
```

以上两种方法均可以实现所要求的传送任务。方法一使用14个字节的指令代码，执行时间为9个机器周期；方法二仅用了9个字节的代码，执行时间也减少到5个机器周期。实际应用中应尽量采用指令代码字节数少、执行时间短的高效率程序，即注意程序的优化。

3.9 程序设计方法

例3-31 数据交换。 将**R0**与**R7**内容互换，**R4**与内存**20H**单元内容互换。

解：参考程序如下：

```
XCHR：MOV    A，R0
      XCH    A，R7
      XCH    A，R0      ；R0与R7内容互换
      MOV    A，R4
      XCH    A，20H
      XCH    A，R4      ；R4与20H单元内容互换
```

3.9 程序设计方法

例3-32 双字节无符号数加法。设被加数存放在内部RAM的51H、50H单元，加数存放在内部RAM的61H、60H单元，相加的结果存放在内部RAM的51H、50H单元，进位位存放在位寻址区的00H位中。

解：程序段如下：

```
MOV   R0，＃50H  ；被加数的低字节地址
MOV   R1，＃60H  ；加数的低字节地址
MOV   A，@R0     ；取被加数低字节
ADD   A，@R1     ；加上加数低字节
MOV   @R0，A     ；保存低字节相加结果
INC   R0         ；指向被加数高字节
INC   R1         ；指向加数高字节
MOV   A，@R0     ；取被加数高字节
ADDC  A，@R1     ；加上加数高字节（带进位加）
MOV   @R0，A     ；存高字节相加结果
MOV   00H，C     ；保存进位 。
```

3.9 程序设计方法

二、分支程序的设计

分支结构又叫条件选择结构，根据不同情况做出判断和选择，以便执行不同的程序段。分支的意思是在两个或多个不同的操作中选择其中的一个。根据不同的条件，确定程序的走向。

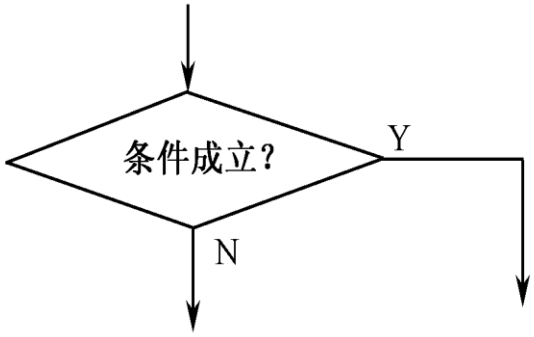
特点：

1. 程序执行流程中必然包含有条件判断，符合条件要求和不符合条件要求分别有不同的处理路径；
2. 在程序设计时，往往借助程序框图（判断框）来指明程序的走向；
3. 一般情况下，每个分支均需单独一段程序，在程序的起始地址赋予一个地址标号，以便当条件满足时转向指定地址单元去执行，条件不满足时仍顺序往下执行；
4. 程序仅有两个出口，两者选一；
5. 条件转移指令都可以用在分支程序中，包括：

1）测试条件符合转移指令

2）比较不相等转移指令

3）减1不为零转移指令



3.9 程序设计方法

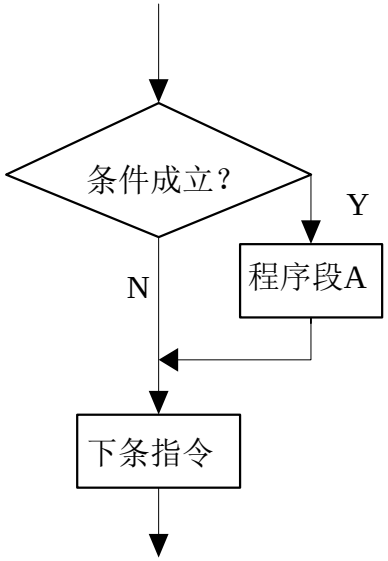
编写分支程序主要在于正确使用转移指令。分支程序有：单分支结构、双分支结构、多分支结构（散转）。

分支程序的设计要点如下：

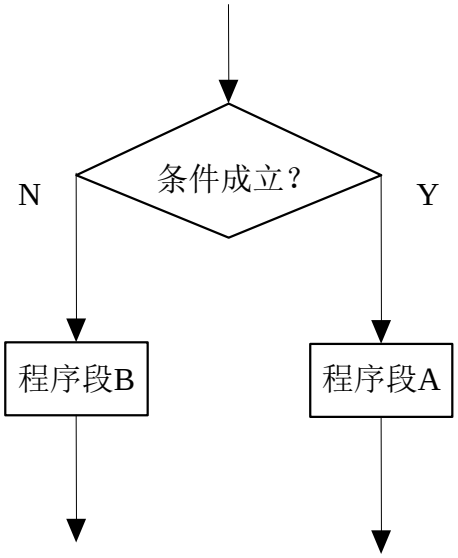
（1）建立测试条件。

（2）选用合适的条件转移指令。

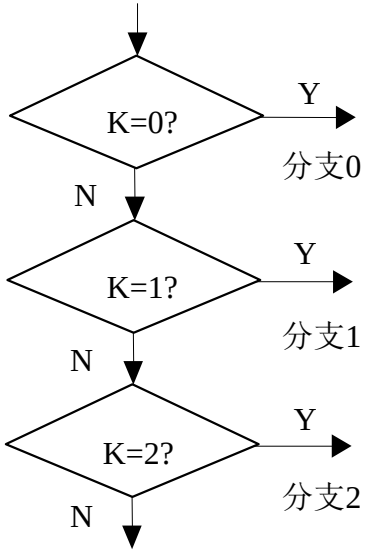
（3）在转移的目的地址处设定标号。



单分支结构



双分支结构



多分支结构

3.9 程序设计方法

分支程序综合举例

1. 单分支程序

当程序的判断仅有两个出口，两者选一，称为单分支结构。通常用条件转移指令来选择并确定程序的分支出口。

例3-33 编程实现，设a存放在累加器A中，b存放在寄存器B中，结果Y存放在A中。若a≥0，则Y=a-b；若a＜0，则Y=a+b。

解： 分析，这里的关键是判a是正数，还是负数；可通过判断ACC.7确定。程序如下：

```
    BR:  JB   ACC.7，MINUS    ；负数，转到MINUS
         CLR  C               ；清进位位
         SUBB A，B            ；A-B
         SJMP DONE
MINUS:   ADD  A，B            ；A+B
DONE:    SJMP $               ；等待
```

3.9 程序设计方法

例3-34 设内部RAM 40H和41H单元中存放2个8位无符号二进制数，试编程找出其中的小数存人30H单元中。

解：
```
        MOV  A，40H
        CJNE  A，41H，LOOP   ；取2个数进行比较
LOOP：  JC  LOOP1            ；根据CY值，判断单分支出口
        MOV  A，41H          ；41H单元中是小数
LOOP1：MOV  30H，A           ；40H单元中是小数
```

例3-35 设对addr1，addr1+1的双字节数取补后存入addr2和 addr2+1单元中，其中高位字节在高地址单元中。8位微机对双字节数取补需分两次进行。首先对低字节数取补，然后判其结果是否为全“0”。若为“0”，则高字节数取补；否则，高位字节数取反。

3.9 程序设计方法

```
解：  START：MOV  R0，#addr1  ；原码低字节地址码送R0
              MOV R1，#addr2  ；补码低字节地址码送R1
              MOV  A，@R0     ；原码低字节内容送A
              CPL   A
              INC   A         ；A内容取反加l，即取补
              MOV  @R1，A     ；低字节补码存addr2单元
              INC   R0        ；指向原码高字节
              INC   R1        ；指向补码高字节
              JZ    LOOP1     ；当（A）＝0，转LOOP1
              MOV  A，@R0     ；原码高字节送A
              CPL   A         ；高字节内容取反
              MOV  @R1，A     ；字节反码存（addr2+1）单元
              SJMP  LOOP2     ；转LOOP2，结束
     LOOP1：MOV  A，@R0       ；低字节补码为0
              CPL   A         ；对高字节数取补
               INC  A
               MOV  @R1，A    ；高字节补码存（addr2+1）单元
      LOOP2：…
```

3.9 程序设计方法

2. 双分支程序

例**3-36**　设变量 **x** 以补码的形式存放在片内**RAM**的**30H**单元，变量 **y** 与 **x** 的关系是：当 **x** 大于**0**时，**y =x**；当 **x =0**时，**y =20H**；当 **x** 小于**0**时，**y =x+5**。编制程序，根据 **x** 的大小求**y**并送回原单元。

解：程序段如下：

```
START: MOV  A，30H
       JZ    NEXT
       ANL   A，#80H   ；判断符号位
       JZ    LP
       MOV  A，#05H
       ADD  A，30H
       MOV  30H，A
       SJMP  LP
NEXT: MOV  30H，#20H
  LP: SJMP  $
```

3.9 程序设计方法

例3-37 编程实现：

$Y=a^2+b$ （当b≥10时）

$Y=a^2-b$ （当b<10时）

解：程序段如下：

```
START: MOV   A，#a
       MOV   B，A
       MUL   AB                ；（B）（A）= a²
       MOV   R0，A             ；（R1）（R0）= a²
       MOV   R1，B
       MOV   A，#b
       CJNE  A，#0AH，MMN      ；b≠10则转移
  MM:  ADD   A，R0             ；b≥10，a²+b=Y
       MOV   R0，A
       MOV   A，#00H
       ADDC  A，R1
       MOV   R1，A
       SJMP  MMNN
```

3.9 程序设计方法

```
MMN：  JNC     MM          ；无借位（即b>10）转MM
       MOV     R3，A       ；R3←b
       MOV     A，R0
       CLR     C
       SUBB    A，R3       ；（R1）（R0）←a²-b
       MOV     R0，A
       MOV     A，R1
       SUBB    A，#00H
       MOV     R1，A
MMNN：MOV      Y0，R0      ；（Y1）（Y0）←结果
       MOV     Y1，R1
 HERE：SJMP    HERE
```

注：Y1、Y0需用位定义伪指令赋值。

3.9 程序设计方法

3. 多分支程序

当程序的判别部分有两个以上的出口流向时，称为多分支选择结构。

51系列单片机指令系统没有多分支转移指令，无法使用单条指令完成多分支转移。要实现多分支转移，一般有两种形式。

（1）使用多条**CJNE**指令，通过逐次比较，实现多分支程序转移

（2）使用散转指令 **JMP @A＋DPTR**实现多分支程序转移

散转指令由数据指针**DPTR**决定多分支转移程序的首地址，由累加器**A**中内容动态地选择对应的分支程序，可从多达**256**个分支中选一。

例 **3-38** 统计从**P1**口输入的字串中正数、负数、零的个数。设**R0**、**R1**、**R2**三个工作寄存器分别为统计正数、负数、零的个数的计数器。

解：根据题意可得程序如下：

3.9 程序设计方法

```
START: CLR  A
       MOV  R0，A
       MOV  R1，A
       MOV  R2，A
       MOV  P1，#0FFH   ；置P1口为输入状态
ENTER: MOV  A，P1       ；从P1口读取一个数
       JZ   ZERO        ；该数为0，转ZERO
       JB   ACC.7，NEG  ；该数为负，转NEG
       INC  R0     ；该数不为0、不为负则必为正数，R0内容加1
       SJMP ENTER       ；循环自P1口取数
 ZERO: INC  R2          ；零计数器加1
       SJMP ENTER
  NEG: INC  R1          ；负数计数器加1
       SJMP ENTER
```

3.9 程序设计方法

例3-39 设变量X的值存放在内部RAM的30H单元中，编程求解下列函数式，将求得的函数值Y存入40H单元。

$$Y=\begin{cases} X+1 & (X\geq 100) \\ 0 & (10\leq X<100) \\ X-1 & (X<10) \end{cases}$$

解：自变量X的值在三个不同的区间所得到的函数值Y不同，编程时要注意区间的划分。程序如下：

```
        MOV  A，30H              ；取自变量X值
        CJNE  A，#10，LOOP       ；与10比较，A中值不改变
 LOOP：JC  LOOP2                 ；若X< 10，转LOOP2
        CJNE  A，＃100，LOOP1    ；与100比较
LOOP1：JNC  LOOP3                ；若X>100，转LOOP3
        MOV  40H，#00H           ；因10≤X<100，故Y=0
        SJMP  EXIT
LOOP2：DEC  A                    ；因X<10，故Y=X-1
        MOV  40H，A
        SJMP  EXIT
LOOP3：INC  A                    ；若X>100，故Y=X+1
        MOV  40H, A
  EXIT：RET
```

3.9 程序设计方法

例3-40 比较内部RAM I、 J单元中A、 B两数的大小。 若A=B，则使内部RAM的位K置 1；若A≠B，则大数存M单元，小数存N单元。 设A、 B数均为带符号数，以补码数存入I、 J中。

解：参考子程序如下：

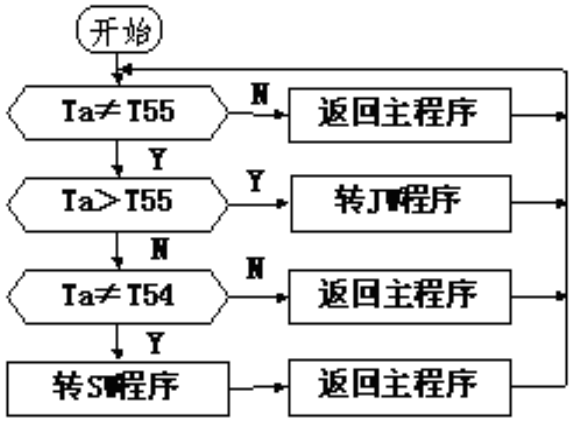
```
        MOV  A，  I          ；A数送累加器A
        ANL   A, ＃80H        ；判A数的正负
        JNZ    NEG           ；A<0 则转至NEG
        MOV   A，J           ；B数送累加器A
        ANL    A，＃80H       ；判B数的正负
        JNZ    BIG1          ；A≥0，B＜0，转BIG1
        SJMP   COMP          ；A≥0，B≥0，转COMP
  NEG:  MOV   A，J           ；B数送累加器A
        ANL   A，＃80H        ；判B数的正负
        JZ    SMALL          ；A＜0，B≥0，转SMALL
 COMP：MOV   A，  I          ；A数送累加器A
        CJNE   A，J，  BIG     ；A≠B则转BIG
        SETB   K             ；A=B，位K置 1
        RET
   BIG: JC    SMALL          ；A<B转SMALL
  BIG1: MOV  M，I            ；大数A存入M单元
        MOV  N，  J          ；小数B存入N单元
        RET
SMALL：MOV    M，J           ；大数B存入M单元
        MOV    N，I          ；小数A存入N单元
        RET
```

3.9 程序设计方法

例**3-41** 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**的**54H**单元存放控制温度下限值（**T54**），在内部**RAM** **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）。

解：根据题意，程序流程图如图。

```
        CJNE  A，55H，LOOP1    ；Ta≠T55，转LOOP1
        AJMP  FH     ；Ta=T55，返回主程序
LOOP1：JNC   JW     ；（CY）=0，Ta＞T55，转JW
        CJNE  A，54H，LOOP2    ；Ta≠T54，转LOOP2
        AJMP  FH     ；Ta=T54，返回主程序
LOOP2：JC    SW     ；（CY）=1，Ta＜ T54，转SW
   FH：  RET          ；T55≥Ta≥T54，返回主程序
```



3.9 程序设计方法

例3-42 设变量X存放在VAR单元中，函数Y存放在FUNC单元。编写按照下式要求给Y赋值的程序。

$$Y=\begin{cases}1 & X>0\\ 0 & X=0\\ -1 & X<0\end{cases}$$

解：由于X为有符号数，因此可以根据它的符号位来决定其正负。 判别符号位是 0 还是1，可利用JB或JNB指令；而判别X是否为 0，则可直接用累加器判零指令JZ。 完成本例任务的程序框图由顺序结构+分支结构组成，并且在分支结构中又嵌套了另一个分支结构，从而形成了三分支而归一的流程。参考程序如下：

```
BR1：MOV  A，VAR          ；取出X送A
     JZ   COMP            ；若X=0 则转移到COMP
     JNB  ACC.7，POSI     ；若X>0则转移到POSI
     MOV  A，#0FFH        ；若X<0 则A=-1
     SJMP COMP            ；转分支结构出口
POSI：MOV  A，#01H        ；X>0 时A=1
COMP：MOV  FUNC，A        ；存函数Y值
HERE：SJMP  HERE          ；结束程序
```

3.9 程序设计方法

例 3-43 3 个无符号单字节整数分别存于R1、 R2、 R3 中，找出其中最大数放于R0 中。

解：首先将 R0 清零，然后进行（R1）与（R0）减法，若（R1）-（R0）>0，则（R1）>（R0）， 把（R1）送R0；否则（R0）保持不变。 再将（R0）分别与（R2）和（R3）比较，比较处理的方法与上面相同，这样比较 3 次后，R0 中即为 3 数中的最大数。 参考程序如下：

```
BR2：MOV   R0，#00H      ；R0 清零
     MOV   A，R1         ；第一个数(R1)送A
     ACALL  COMP         ；比较（R1）与（R0）大小
     MOV   A，R2         ；第二个数（R2）送A
     ACALL  COMP         ；比较（R2）与（R0）大小
     MOV  A，  R3        ；第三个数（R3）送A
     ACALL  COMP         ；比较（R3）与（R0）大小
HERE：SJMP   HERE
COMP：MOV   R4，A        ；R4 暂存A的内容
      CLR   C            ；清进位位C
      SUBB  A，R0        ；（A）-（R0）
      JC  M1             ；（A）<（R0）转至M1
      MOV  A，R4         ；恢复A值
      MOV  R0，  A       ；（A）>（R0）时大数存R0
  M1：RET
```

3.9 程序设计方法

例3-44　根据R7的内容转向相应的处理程序。设R7的内容为0～N，对应的处理程序的入口地址分别为PP0～PPN。

解：程序段如下：

```
START：MOV  DPTR，#TAB    ；置分支入口地址表首址
       MOV  A，R7         ；分支转移序号送A
       ADD  A，R7         ；分支转移序号乘以2
       MOV  R3，A         ；暂存于R3
       MOVC A，@A+DPTR ；取高位地址
       XCH  A，R3
       INC  A
       MOVC A，@A+DPTR ；取低位地址
       MOV  DPL，A        ；处理程序入口地址低8位送DPL
       MOV  DPH，R3       ；处理程序入口地址高8位送DPH
       CLR  A
       JMP  @A+DPTR
  TAB：DW   PP0
       DW   PP1
       … … …
       DW   PPN
```

3.9 程序设计方法

3.9.3 循环程序设计方法

循环结构是重复执行一系列操作，直到某个条件出现为止。采用循环结构程序设计可以有效地缩短程序，减少程序占用的内存空间，提高程序的紧凑性和可读性。循环程序结构是分支程序中的一个特殊形式。

一、循环结构的组成

循环程序结构一般由4部分组成：初始化部分、循环处理部分、循环控制部分和循环结束部分。

（1）初始化部分：位于循环程序开头，用来设置循环处理之前的初始状态，如循环次数的设置、变量初值的设置、地址指针的设置等。有些情况下还要进行现场保护。

（2）循环处理部分：又称为循环体，位于循环内，是循环程序的主体，是重复执行的数据处理程序段，它是循环程序的核心部分。要求编写得尽可能简练，以提高程序的执行速度。

（3）循环控制部分：这部分用来控制循环继续与否。一般由循环次数的修改、循环修改和条件语句等组成，用于控制循环次数和修改每次循环的相关参数，如地址指针等。

3.9 程序设计方法

实现方法主要有循环计数控制法和条件控制法。

◆循环次数不确定的情况：满足条件就结束循环。采用条件控制法。

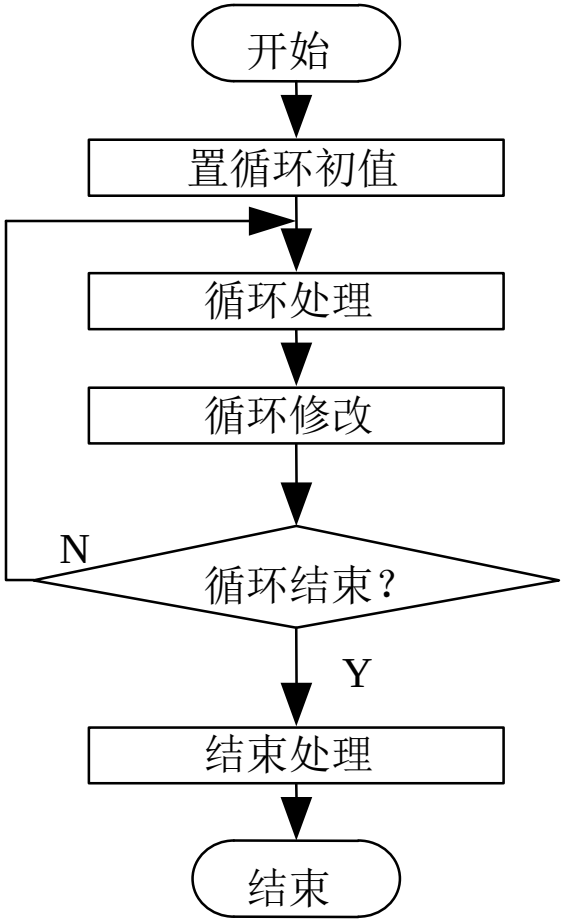
◆循环次数已知的情况：采用计数控制法。

（4）结束部分：这部分是对循环程序全部执行结束后的结果进行分析、处理和保存。有些情况下需恢复现场。

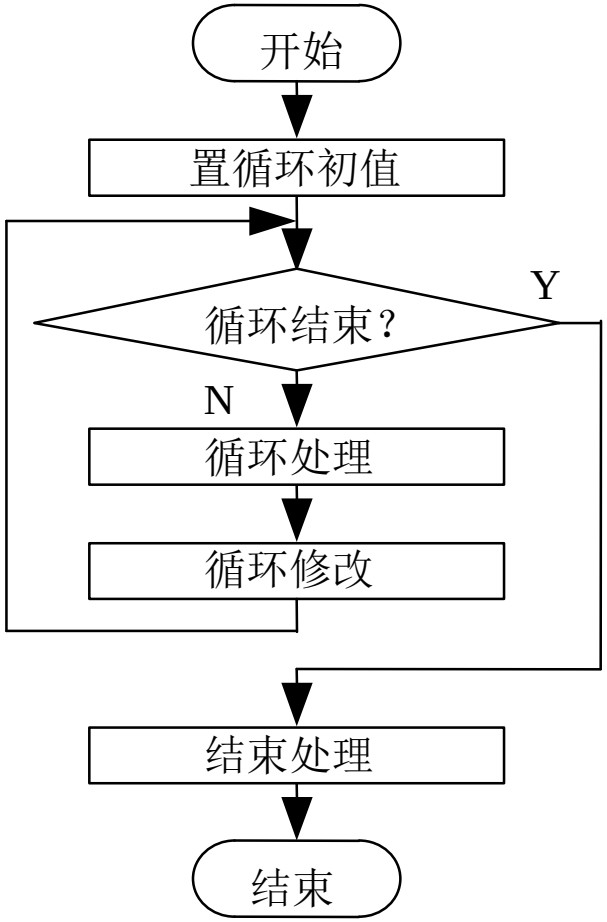
循环程序有先处理后判断和先判断后处理两种基本结构。图a）为先处理后判断的结构，不管条件如何，它至少执行一次循环体。图b）为先判断后处理的结构，先检查控制条件是否成立，决定循环程序是否执行。

对循环次数的控制有多种：循环次数是已知的，可用循环次数计数器控制循环；若循环次数是未知的，可以按条件控制循环。

3.9 程序设计方法



a）先处理后判断



b）先判断后处理

3.9 程序设计方法

先处理后判断的结构通常使用计数循环方法实现，故也称计数循环结构形式。计数循环程序的特点是循环次数已知，必须在初始化部分设定计数的初值，循环控制部分依据计数器的值决定循环次数。一般均设置为减“1”计数器，每循环一次自动减“1”，直到回0时结束循环。可用以下减1不为0 指令实现：

```
DJNZ    Rn，rel        ；以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
```

先判断后处理的结构一般常用比较转移指令或条件判跳指令进行控制和实现。

根据循环程序的结构不同也可分为单重循环和多重循环。在一个循环程序的循环体中不包含另外的循环结构称为单重循环。多重循环就是在循环内套循环的结构形式，也称循环嵌套。多重循环的执行过程是从内向外逐层展开的。内层执行完全部循环后，外层则完成一次循环，逐次类推。层次必须分明，层次之间不能有交叉，否则将产生错误。

3.9 程序设计方法

二、循环结构程序综合举例

1. 单重循环程序

1）先处理后判断（计数循环）

例 **3-45** 工作单元清零。 在应用系统程序设计时，有时经常需要将存储器中各部分地址单元作为工作单元，存放程序执行的中间值或执行结果，工作单元清零工作常常放在程序的初始化部分中。设有**50**个工作单元，其首址为外部存储器**8000H**单元，编写其工作单元清零程序。

解：参考程序如下：

```
CLEAR：CLR  A          ■
       MOV  DPTR，  #8000H   ；工作单元首址送指针 ■
       MOV  R2，#50          ；置循环次数 ■
CLEAR1：MOVX  @DPTR，A ■
       INC   DPTR            ；修改指针 ■
       DJNZ  R2，  CLEAR1    ；控制循环 ■
       RET
```

3.9 程序设计方法

例 **3-46** 软件延时。执行一段循环程序，而循环程序执行的时间即为延时时间。

```
DELAY：MOV  R2 ＃data      ；预置计数循环控制常数
DELAY1：DJNZ  R2，DELAY1   ；当（R2）≠0，转向本身
        RET
```

根据**R2**的不同初值（**0** ～**FFH**），可实现**4**～**514**个机器周期的延时。如，

```
DELAY：MOV  R2，＃0AH     ；给R2赋循环初值
DELAY1：DJNZ  R2，  DELAY1 ；（R2）←（R2）-1，若（R2）≠0则循环
```

由于**DJNZ R2，DELAY1** 为双字节双周期指令，当单片机主频为 **12 MHz**时，执行一次该指令需 **2**个机器周期即**2 μs**。 因此，**R2** 中置入循环次数为 **10** 时，执行该循环指令可产生**20 μs** 的延时时间。

3.9 程序设计方法

例**3-47　50ms**延时程序。若晶振频率为**12MHz**，则一个机器周期为**1μs**。执行一条**DJNZ**指令需要**2**个机器周期，即**2μs**。

解：采用计数循环法实现延时，循环次数可以通过计算获得，并选择先处理后判断的循环结构。程序段如下：

```
   DEL：MOV  R7，#200   ；1 μs
  DEL1：MOV  R6，#123   ；1 μs
        NOP             ；1 μs
  DEL2：DJNZ  R6，DEL2  ；2μs，计（2×123）μs
        DJNZ  R7，DEL1  ；2μs，
        RET
```

共计 [（2×123＋2＋ 2）×200+1] μs，即**50.001ms**

3.9 程序设计方法

例3-48 将内部RAM中从DATA单元开始的 10 个无符号数相加，相加结果送SUM和SUM+1 单元保存。

解: 相加结果超过8位二进制数，相应的程序如下：

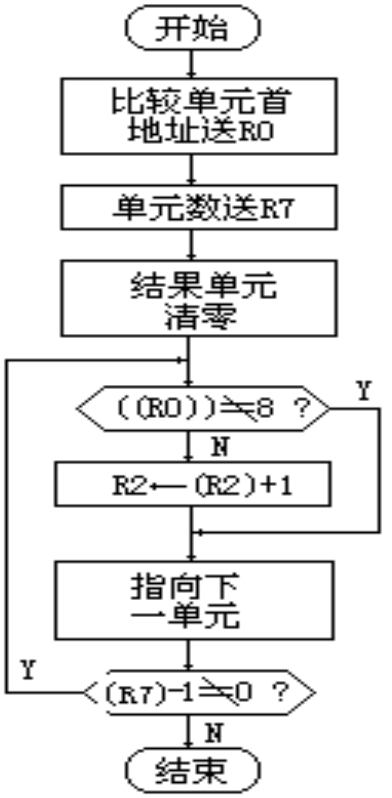
```
        MOV   R2，#00H        ；给 R2 置初值
        MOV   R0，#0AH        ；给 R0 置计数器初值
        MOV   R1，#DATA       ；数据块首址送R1
        CLR   A               ；A清零
LOOP：  ADD   A，@R1          ；加一个数
        JNC   NEXT            ；判断部分和是否超过8位
        INC   R2              ；存部分和高字节
NEXT ： INC   R1              ；修改地址，指向下一个数
        DJNZ  R0，LOOP        ；R0 减 1，不为零循环
        MOV   SUM，A          ；存 10 个数相加和低字节
        MOV   SUM+1，R2       ；存 10 个数相加和高字节
```

3.9 程序设计方法

例3-49 内部RAM 30H开始的20个连续单元中，存放有20个数，统计等于8的单元个数，结果放在R2中。

解：分析，取一个数与8比较，若相等R2加1，不相等跳过。并作20次重复即可。程序流程图如图。程序如下：

```
START：MOV   R0，#30H    ；设数据区指针
       MOV  R7，#20      ；设置循环计数器
       MOV   R2，#0      ；设置统计计数器
 LOOP：CJNE  @R0，#08H，NEXT
       INC   R2
 NEXT：INC   R0
       DJNZ  R7，LOOP
       SJMP  $
```



3.9 程序设计方法

例3-50 设内部RAM存有一无符号数数据块，长度为128字节，在以30H单元为首址的连续单元中。试编程找出其中最小的数，并放在20H单元。

解：程序如下：

```
        MOV  R7，#7FH     ；设置比较次数
        MOV  R0，#30H     ；设置数据块首址
        MOV  A，@R0       ；取第一个数
        MOV  20H，A       ；第一个数暂存于20H单元，作为最小数
LOOP1：INC  R0
        MOV  A，@R0       ；依次取下一个数
        CJNE  A，20H，LOOP
LOOP：  JNC  LOOP2        ；两数比较后，其中小的数放在20H单元
        MOV  20H，A
LOOP2：DJNZ  R7，LOOP1    ；R7中内容为零则比较完
        SJMP  $
```

3.9 程序设计方法

例 3-51 从BLOCK单元开始有一个无符号数数据块，其长度存于LEN单元，试求出数据块中最大的数并存入MAX单元。

解：数据块长度已知，采用单重计数循环的程序结构。 程序如下：

```
LOOP:  MOV   R0，＃BLOCK  ；数据块首址送R0
       MOV   R1，LEN      ；数据块长度送R1
       MOV   MAX，＃00H   ；存最大数单元清零
LOOP1: MOV   A，  MAX     ；（A）←（MAX）
       CLR   C            ；清C
       SUBB  A，  @R0     ；（MAX）-（（R0））
       JNC   NEXT         ；若（MAX）>（（R0）），则转移
       MOV   MAX，  @R0   ；若（MAX）<（（R0）），则（MAX）←（（R0））
NEXT:  INC   R0           ；修改地址指针
       DJNZ  R1，  LOOP1  ；若（R1）≠0则循环搜索
       RET
```

3.9 程序设计方法

例3-52 数据块移动。将2000H单元开始的一批数据传送到从3000H开始的单元中，数据长度在内部RAM的30H中。

解：程序如下：

```
        MOV  DPTR，#2000H   ；源数据区首址
        PUSH  DPL           ；源数据区首址压栈保护
        PUSH  DPH
        MOV  DPTR，#3000H   ；目的数据区首址
        MOV  R6，DPL        ；目的数据区首址存入寄存器
        MOV  R7，DPH
  LP：POP  DPH              ；取数据区地址指针
        POP  DPL
        MOVX  A，@DPTR      ；取源数
        INC  DPTR
        PUSH  DPL
        PUSH  DPH
        MOV  DPL，R6        ；取目的数据区地址指针
        MOV  DPH，R7
        MOVX  @DPTR，A      ；存入目的数据区
        INC  DPTR
        MOV  R6，DPL
        MOV  R7，DPH
        DJNZ  30H，LP       ；若数据块未移完，则继续
   POP  DPH
        POP  DPL
        SJMP  $
```

注意：2000H和3000H都在外部RAM或I/O中，使用地址指针均为DPTR，注意DPTR的保护、恢复。

3.9 程序设计方法

2）先判断后处理（条件控制循环）

根据控制循环结束的条件，决定是否继续循环程序的执行。所谓的结束条件可以是搜索到某个参数（比如回车符“**CR**”），也可以是发生的某种变化（如故障引起电路电平变化 ）等，什么时侯结束循环是不可预知的。

例**3-53** 设有一字符串以回车符为结束标志，并存放在内部**RAM40H**单元开始的连续存储单元中，编写测试字符串长度的程序。

解：分析：为测试字符串的长度，应使用逐个字符依次与回车符（**ODH**）比较的方法。为此需要设置一个字符串指针和一个长度计数器，字符串指针用于指定字符，长度计数器用于累加字符串的长度。如比较不相等，则长度计数器和字符串指针都加**1**，以继续往下比较；如果比较相等，则表示该字符为回车符，字符串结束，长度计数器的值就是字符串的长度。程序如下：

```
        MOV R7，＃0FFH  ；设置长度计数器初值
        MOV R0，＃3FH   ；设置字符串指针初值
LOOP：  INC R7
        INC R0
        CJNE  @R0，＃0DH，LOOP
        RET
```

3.9 程序设计方法

例3-54 将内部RAM的一个数据块（首址为DATAI）传送到外部RAM（首址为DATA2），遇到传送的数据为零时停止。

解：程序如下：

```
START：MOV  R0，＃DATA1      ；置内部RAM数据指针
       MOV  DPTR，＃DATA2    ；置外部RAM数据指针
LOOP1：MOV  A，@R0           ；内部RAM单元内容送A
       JZ  LOOP2             ；判传送数据是否为零，A为零则转移
       MOVX  @DPTR，  A      ；传送数据不为零，送外部RAM
       INC  R0               ；修改地址指针
       INC  DPTR
       SJMP   LOOP1          ；继续传送
LOOP2：RET                   ；结束传送，返回主程序
```

3.9 程序设计方法

例3-55 将内部**RAM**中起始地址为**data**的数据串传送到外部**RAM**中起始地址为**buffer**的存储区域内，直到发现‘**$**’字符停止传送。

解：由于循环次数事先不知道，但循环条件可以测试到。所以，采用先判断后处理的结构比较适宜。程序段如下：

```
        MOV   R0，#data
        MOV   DPTR，#buffer
LOOP0：MOV   A，@R0
        CJNE   A，#24H，LOOP1 ；判断是否为‘$’字符
        SJMP   LOOP2          ；是‘$’字符，转结束
LOOP1：MOVX  @DPTR，A        ；不是‘$’字符，执行传送
        INC    R0
        INC    DPTR
        SJMP   LOOP0          ；传送下一数据
LOOP2：… …
```

3.9 程序设计方法

例3-56 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，数据串的最大长度在内存20H中。

解：程序如下：

```
        MOV   R0，#DATA          ；数据区首地址
        MOV   DPTR，#BUFFER      ；数据区长度指针
LOOP：MOV  A，@R0
        CJNE  A，#24H ，LOOP2    ；判是否为“$”符(24H)
        SJMP LOOP1               ；是“$”符，则结束
LOOP2：MOV  A，@R0               ；不是“$”符，则传送
        MOVX  @DPTR，A
        INC   R0
        INC   DPTR
        DJNZ  20H，LOOP          ；数据串未查完，继续
LOOP1：  …
```

注：第一个是条件循环控制，以找到ASCII码“$”符为循环结束条件；第二个是计数循环结构，万一找不到ASCII码“$”符，则由数据串的最大长度作为计数循环控制。

3.9 程序设计方法

2. 多重循环程序

例 **3-57** 设计 **100 ms**延时程序。

解**:** 如前所述，计算机执行一条指令需要一定的时间，由一些指令组成一段程序，并反复循环执行，利用计算机执行程序所用的时间来实现延时，这种程序称为延时程序。 如当系统使用 **12 MHz**晶振时，一个机器周期为 **1 μs** ，执行一条双字节双周期**DJNZ**指令的时间为 **2 μs,** 因此，执行该指令 **50 000** 次，就可以达到延时 **100 ms**的目的。 对于 **50 000** 次循环可采用外循环、 内循环嵌套的多重循环结构。程序如下：

```
START：  MOV  R6，#0C8H     ；外循环 200 次
LOOP1：  MOV  R7，#0F8H     ；内循环 248 次
         NOP                ；时间补偿
LOOP2：  DJNZ  R7，  LOOP2  ；延时 2 μs×248=496 μs
         DJNZ  R6，LOOP     ；延时 500 μs×200=100 ms
         RET
```

以上程序执行**MOV Rn，#data** 指令的时间为 **1 μs**，**DJNZ**指令 **2 μs**，**NOP** 指令**1 μs**，所以，内循环延迟时间：**1 μs+1 μs+2 μs×248=498 μ**， 外循环延迟时间：**1 μs+**（内环延时**+2 μs**）×**200=100.001 ms**。

3.9 程序设计方法

3.9.4子程序设计和参数传递方法简介

一、 子程序的概念 ■

在解决实际问题时，经常会遇到一个程序中多次使用同一个程序段，例如延时程序、查表程序、算术运算程序等功能相对独立的程序段。为了节约内存，通常把这些基本操作功能编制为程序段作为独立的子程序，以供不同程序或同一程序反复调用。调用子程序的程序称为主程序或调用程序。被调用的程序称为子程序。

二、子程序的特点

1.避免在几个地方对同样一种操作进行重复编程；

2.简化了程序的逻辑结构；

3.缩短了程序长度，节省了程序存储空间；

4.便于调试。

3.9 程序设计方法

三、子程序的调用与返回

子程序主要特点是，在执行过程中需要由其它程序来调用，执行完后又需要把执行流程返回到调用该子程序的主程序。

主程序调用子程序的过程：在主程序中需要执行这种操作的地方执行一条调用指令（**LCALL**或**ACALL**），然后程序转到子程序，当完成规定的操作后，再执行子程序最后一条**RET**指令返回到主程序断点处，继续执行下去。

（**1**）子程序的调用

子程序的起始地址：子程序的第一条指令地址，通常称为子程序首地址或入口地址，往往采用标号加以表示，调用指令的下一条指令地址，通常称为返回地址或断点。

程序的调用过程：单片机执行**ACALL**或**LCALL**指令后，首先将当前的**PC**值（调用指令的下一条指令的首地址）压入堆栈（低**8**位先进堆栈，高**8**位后进堆栈），然后将子程序的起始地址送入**PC**，转去执行子程序。

3.9 程序设计方法

（2）子程序的返回

返回指令：**RET**。设置在子程序的末尾，表示子程序执行完毕。它的功能是自动将断点地址从堆栈弹出送**PC**，从而实现程序返回原程序断点处继续往下执行。

主程序的断点地址：子程序执行完毕后，返回主程序的地址称为主程序的断点地址，它在堆栈中保存。

子程序的返回过程：子程序执行到**RET**指令后，将压入堆栈的断点地址弹回给**PC**（先弹回**PC**的高**8**位，后弹回**PC**的低**8**位），使程序回到原先被中断的主程序地址（断点地址）去继续执行。

注意：中断服务程序是一种特殊的子程序，它是在计算机响应中断时，由硬件完成调用而进入相应的中断服务程序。**RETI**指令与**RET**指令相似，区别在于**RET**是从子程序返回，**RETI**是从中断服务程序返回。

子程序调用时要注意两点：一是现场的保护和恢复；二是主程序与子程序的参数传递。

3.9 程序设计方法

四、现场保护与恢复

在子程序执行过程中常常要用到单片机的一些通用单元，如工作寄存器**R0~R7**、累加器**A**、数据指针**DPTR**，以及有关标志和状态等。而这些单元中的内容在调用结束后的主程序中仍有用，所以需要进行保护，称为现场保护。

在调用子程序时，单片微机只是自动保护断点地址。**PSW**、**A**、**B**等可通过压栈指令进栈保护。工作寄存器保护采用选择不同工作寄存器组的方式来达到的。

当子程序执行完后，即返回主程序时，应先将上述内容送回到来时的寄存器中去，这后一过程称为恢复现场。恢复现场通常在从子程序返回之前将堆栈中保存的内容弹回各自的寄存器。

保护与恢复的方法有以下两种：

1）在主程序中实现，其特点是结构灵活；

2）在子程序中实现，其特点是程序规范、清晰。

注意，无论采用哪种方法，保护与恢复的顺序要对应。

3.9 程序设计方法

五、参数传递

子程序调用中的特殊问题是参数传递。由于子程序是主程序的一部分，所以，在程序的执行时必然要发生数据上的联系。子程序调用中，主程序应通过某种方式先把有关的参数（即子程序的入口参数）存入约定的位置，子程序在执行时，可以从约定的位置取得参数，当子程序执行完，又需要将得到的结果（即子程序的出口参数）通过某种方式再存入约定的位置，返回主程序后，主程序可以从这些约定的位置上取得需要的结果，这就是参数的传递。

入口参数：子程序需要的原始参数。主程序在调用子程序前将入口参数送到约定的存储器单元（或寄存器）中，然后子程序从约定的存储器单元（或寄存器）中获得这些入口参数。

出口参数：子程序根据入口参数执行程序后获得的结果参数。子程序在结束前将出口参数送到约定的存储器单元（或寄存器）中，然后主程序从约定的存储器单元（或寄存器）中获得这些出口参数。

3.9 程序设计方法

子程序的参数传递方法有以下几种：

（1）利用工作寄存器或累加器

将入口参数或出口参数放在工作寄存器或累加器中。在这种方式中，要把预传递的参数存放在累加器**A**或工作寄存器**R0~R7**中。即在主程序调用子程序时，应事先把子程序需要的数据送入累加器**A**或指定的工作寄存器中，当子程序执行时，可以从指定的单元中取得数据，执行运算。反之，子程序也可以用同样的方法把结果传送给主程序。优点是程序简单、运算速度较快，缺点是工作寄存器有限。

（2）利用存储器

入口参数或出口参数在存储器中，利用指针寄存器间址。当传送的数据量比较大时，可以利用存储器实现参数的传递。在这种方式中，事先要建立一个参数表，用指针指示参数表所在的位置。当参数表建立在内部**RAM**时，用**R0**或**R1**作参数表的指针。当参数表建立在外部**RAM**时，用**DPTR**作参数表的指针。

优点是能有效节省传递数据的工作量。

3.9 程序设计方法

（3）利用堆栈

利用堆栈传递参数是在子程序嵌套中常采用的一种方法。在调用子程序前，主程序可用**PUSH**指令将子程序中所需数据压入堆栈，进入执行子程序时，子程序用**POP**指令访问堆栈，同时可把结果参数压入回堆栈。返回后，主程序再用**POP**指令从堆栈中弹出数据。优点是简单，能传递的数据量较大，不必为特定的参数分配存储单元。

（4）利用位地址传送子程序参数。

一般说来，当相互传递的数据较少时，采用寄存器传递方式可以获得较快的传递速度；当相互传递的数据较多时，宜采用存储器或堆栈方式传递；如果是子程序嵌套时，最好是采用堆栈方式。

3.9 程序设计方法

六、子程序嵌套

在子程序的执行过程中，可能出现在子程序中再次调用其它子程序的情况。这种子程序调用子程序的现象通常称为子程序嵌套。**51**系列单片机允许多重嵌套。