本章分为四节，主要介绍：

4.1 程序编制的方法和技巧

4.1.1程序编制的步骤

一、预完成任务的分析

首先，要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。其次，要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。

二、进行算法的优化

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。

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。

三、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，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。

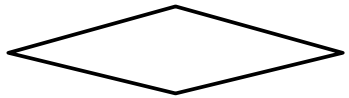
•清晰正确的流程图是编制正确无误的应用程序的基础和条件。所以，绘制一个好的流程图，是程序设计的一项重要内容。

•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。

常用的流程图符号有：开始和结束符号、工作任务符号、判断分支符号、程序连接符号、程序流向符号等

开始或结束符号

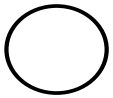
判断分支符号



程序流向符号

工作任务符号

程序连接符号



程序流向符号

4.1.2 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。

多重循环，注意各重循环的初值和循环结束条件，避免出现“死循环”现象；

通用的子程序，除了用于存放子程序入口参数的寄存器外，子程序中用到的其它寄存器的内容应压入堆栈进行现场保护，并要特别注意堆栈操作的压入和弹出的平衡；

中断处理子程序除了要保护程序中用到的寄存器外，还应保护标志寄存器。

4.1.3 汇编语言的语句格式

语句行由四个字段组成 ：

[标号：]操作码 [操作数] [；注释]

括号内的部分可以根据实际情况取舍。每个字段之间要用分隔符分隔，可以用作分隔符的符号有空格、冒号、逗号、分号等。

如：LOOP：MOV A，# 7FH ；A←7FH

一、标号

标号是语句地址的标志符号，用于引导对该语句的非顺序访问。有关标号的规定为：

- 由1～8个ASCII字符组成。第一个字符必须是字母，其余字符可以是字母、数字或其他特定字符；
- 不能使用已经定义了的符号作为标号。如指令助记符、寄存器符号名称等；
- 后边必须跟冒号。

二、操作码

操作码用于规定语句执行的操作。它是汇编语句中唯一不能空缺的部分。它由指令助记符表示。

三、操作数

操作数用于给指令的操作提供数据或地址。在一条汇编语句中操作数可能是空缺的，也可能包括一项，还可能包括两项或三项。各操作数间以逗号分隔。操作数字段的内容可能包括以下几种情况：

（1）工作寄存器名；

（2）特殊功能寄存器名；

（3）标号名；

（4）常数；

（5）符号“$”，表示程序计数器PC的当前值；

（6）表达式。

四、注释

注释只是对语句的说明。注释字段可以增加程序的可读性，有助于编程人员的阅读和维护。注释字段必须以分号“；”开头，长度不限，当一行书写不下时，可以换行接着书写，但换行时应注意要在开头使用分号“；”。

五、数据的表示形式

数据可以有以下几种表示形式：

- 二进制数，末尾以字母 B 标识。如：1000 1111B；
- 十进制数，末尾以字母 D 标识或将字母D省略。如：88D，66；
- 十六进制数，末尾以字母 H 标识。如：78H，0A8H（但应注意的是，十六进制数以字母A～F开头时应在其前面加上数字“0”。）；

ASCII码，以单引号括起来标识。如：‘AB’，‘1245’

4.2 源程序的编辑和汇编

4.2.1 源程序的编辑与汇编

一、源程序的编辑

源程序的编写要依据80C51汇编语言的基本规则，特别要用好常用的汇编命令（即伪指令），例如下面的程序段：

```
ORG   0040H
MOV  A，#7FH
MOV  R1，#44H
END
```

这里的ORG和END是两条伪指令，其作用是告诉汇编程序此汇编源程序的起止位置。编辑好的源程序应以“.ASM”扩展名存盘，以备汇编程序调用。

二、源程序的汇编

将汇编语言源程序转换为单片机能执行的机器码形式的目标程序的过程叫汇编。常用的方法有两种：

•手工汇编时，把程序用助记符指令写出后，通过手工方式查指令编码表，逐个把助记符指令翻译成机器码，然后把得到的机器码程序（以十六进制形式）键入到单片机开发机中，并进行调试。

•机器汇编是在常用的个人计算机PC上，使用交叉汇编程序将汇编语言源程序转换为机器码形式的目标程序。生成的目标程序由PC机传送到开发机上，经调试无误后，再固化到单片机的程序存储器ROM中。

源程序经过机器汇编后，形成的若干文件中含有两个主要文件，一是列表文件，另一个是目标码文件。因汇编软件的不同，文件的格式及信息会有一些不同。但主要信息如下：

列表文件：

地 址	目标码	汇编程序
		ORG 0040H
0040H	747F	MOV A，#7FH
0042H	7944	MOV R1，#44H
		END

目标码文件：

首地址	末地址	目标码
0040H	0044H	747F7944

4.2.2 伪指令

伪指令是汇编程序能够识别并对汇编过程进行某种控制的汇编命令。它不是单片机执行的指令，所以没有对应的可执行目标码，汇编后产生的目标程序中不会再出现伪指令。

一、起始地址设定伪指令 ORG

格式为：

ORG 表达式

该指令的功能是向汇编程序说明下面紧接的程序段或数据段存放的起始地址。表达式通常为16进制地址，也可以是已定义的标号地址。

```
        ORG   8000H
START：MOV   A，#30H
        … …
```

此时规定该段程序的机器码从地址8000H单元开始存放。

在每一个汇编语言源程序的开始，都要设置一条ORG伪指令来指定该程序在存储器中存放的起始位置。若省略ORG伪指令，则该程序段从0000H单元开始存放。在一个源程序中，可以多次使用ORG伪指令规定不同程序段或数据段存放的起始地址，但要求地址值由小到大依序排列，不允许空间重叠。

二、汇编结束伪指令 END

格式为：

END

该指令的功能是结束汇编。

汇编程序遇到END伪指令后即结束汇编。处于END之后的程序，汇编程序将不处理。

三、字节数据定义伪指令 DB

[标号：] DB 字节数据表

功能是从标号指定的地址开始，在ROM中定义字节数据。该伪指令将字节数据表中的数据根据从左到右的顺序依次存放在指定的存储单元中。一个数据占一个存储单元。例如：

DB “how are you？”

把字符串中的字符以ASCII码的形式存放在连续的ROM单元中。又如：

DB －2，－4，－6，8，10，18

把6个数转换为十六进制表示（FEH，FCH，FAH，08H，0AH，12H），并连续地存放在6个ROM。

该伪指令常用于存放数据表格。如要存放显示用的十六进制的字形码，可以用多条DB指令完成：

```
DB  0C0H，0F9H，0A4H，0B0H
DB  99H，92H，82H，0F8H
DB  80H，90H，88H，83H
DB  0C6H，0A1H，86H，84H
```

四、字数据定义伪指令 DW

[标号：] DW 字数据表

功能是从标号指定的地址单元开始，在程序存储器中定义字数据。该伪指令将字或字表中的数据根据从左到右的顺序依次存放在指定的存储单元中。应特别注意：16位的二进制数，高8位存放在低地址单元，低8位存放在高地址单元。例如：

```
        ORG   1400H
DATA：DW   324AH，3CH
        … …
```

汇编后，（1400H）=32H，（1401H）= 4AH，（1402H）=00H，（1403H）=3CH。

五、空间定义伪指令 DS

[标号：] DS 表达式

功能是从标号指定的地址单元开始，在程序存储器中保留由表达式所指定的个数的存储单元作为备用的空间，并都填以零值。例如：

```
        ORG  3000H
BUF：  DS    50
        … …
```

汇编后，从地址3000H开始保留50个存储单元作为备用单元。

六、赋值伪指令 EQU

符号名 EQU 表达式

功能是将表达式的值或特定的某个汇编符号定义为一个指定的符号名。例如：

```
    LEN  EQU  10
    SUM  EQU  21H
  BLOCK  EQU  22H
         CLR   A
         MOV  R7，＃LEN
         MOV  R0，＃BLOCK
LOOP：ADD  A，@R0
         INC   R0
         DJNZ  R7，LOOP
         MOV  SUM，A
         END
```

该程序的功能是，把BLOCK单元开始存放的10个无符号数进行求和，并将结果存入SUM单元中。

七、位地址符号定义伪指令 BIT

格式为：

符号名　BIT　位地址表达式

功能是将位地址赋给指定的符号名。其中，位地址表达式可以是绝对地址，也可以是符号地址。

例如：

```
ST    BIT   P1.0
```

将P1.0的位地址赋给符号名ST，在其后的编程中就可以用ST来代替P1.0。

4.3 基本程序结构

4.3.1 顺序程序

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。

一、数据传送

例 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。

ACC	
2EH	78H
2DH	56H
2CH	34H
2BH	12H
2AH	00H

ACC	78H
2EH	56H
2DH	34H
2CH	12H
2BH	00H
2AH	00H

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
```

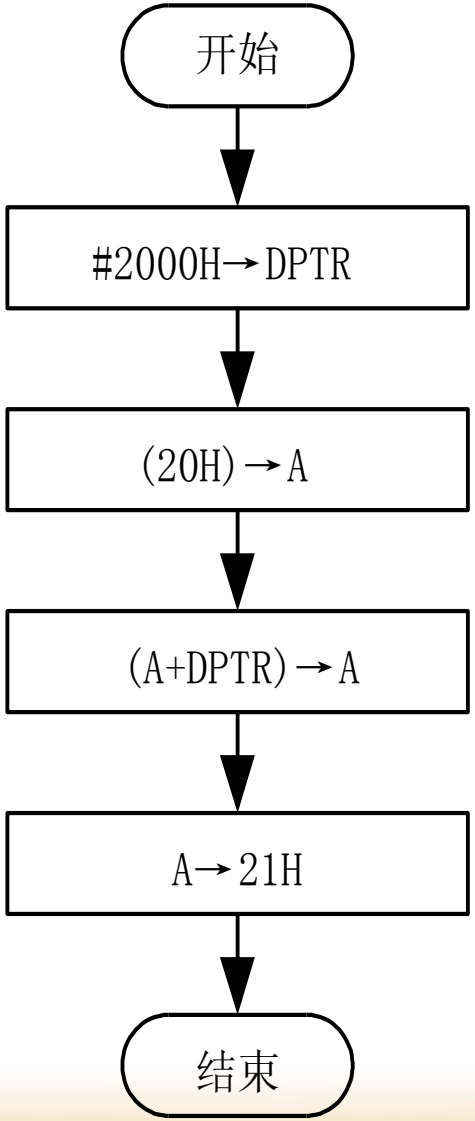
方法二：

```
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，执行时间也减少到了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。

例　有一变量存放在片内RAM的20H单元，其取值范围为：00H～05H。要求编制一段程序，根据变量值求其平方值，并存入片内RAM的21H单元。程序如下：

```
        ORG   1000H
START:  MOV  DPTR，#2000H
        MOV   A，20H
        MOVC  A，@A+DPTR
        MOV   21H，A
        SJMP   $
        ORG   2000H
TABLE:  DB    00，01，04，09，16，25
        END。
```



•在程序存储器的一片存储单元中建立起该变量的平方表。用数据指针DPTR指向平方表的首址，则变量与数据指针之和的地址单元中的内容就是变量的平方值。

•采样MOVC A，@A+PC指令也可以实现查表功能，且不破坏DPTR的内容，从而可以减少保护DPTR的内容所需的开销。但表格只能存放在MOVC A，@A+PC指令后的256字节内，即表格存放的地点和空间有一定限制。

三、简单运算

由于80C51指令系统中只有单字节加法指令，因此对于多字节的相加运算必须从低位字节开始分字节进行。除最低字节可以使用ADD指令外，其他字节相加时要把低字节的进位考虑进去，这时就应该使用ADDC指令。

例　双字节无符号数加法。

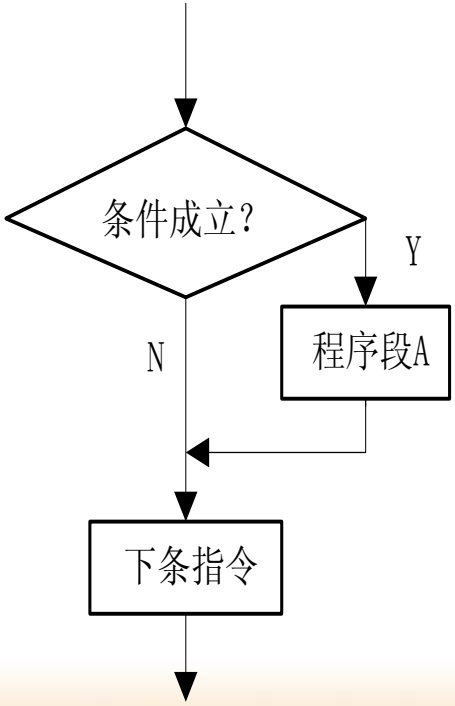
设被加数存放在内部RAM的51H、50H单元，加数存放在内部RAM的61H、60H单元，相加的结果存放在内部RAM的51H、50H单元，进位存放在位寻址区的00H位中。

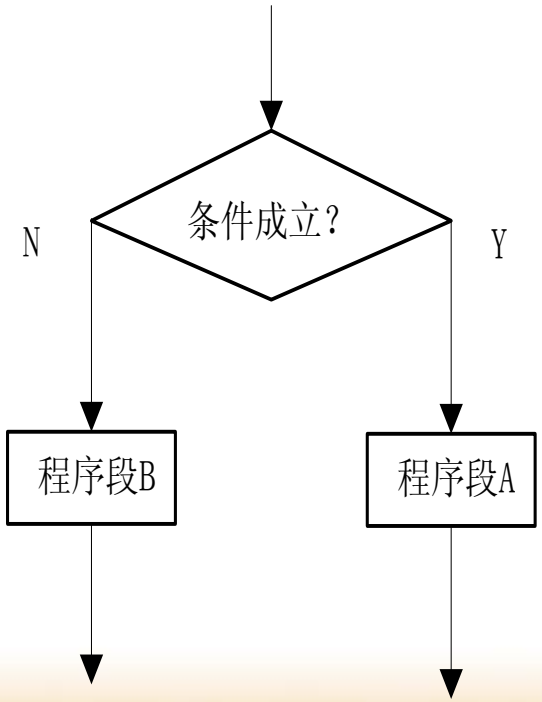
程序段如下：

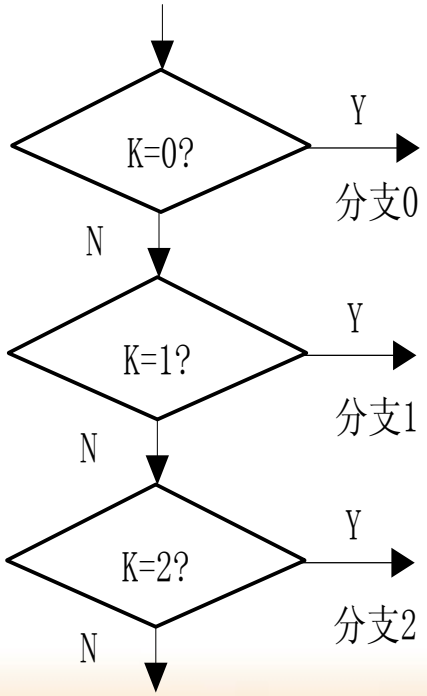
```
MOV   R0，＃50H ；被加数的低字节地址
MOV   R1，＃60H ；加数的低字节地址
MOV   A，@R0   ；取被加数低字节
ADD   A，@R1   ；加上加数低字节
MOV   @R0，A   ；保存低字节相加结果
INC   R0       ；指向被加数高字节
INC   R1       ；指向加数高字节
MOV   A，@R0   ；取被加数高字节
ADDC  A，@R1   ；加上加数高字节（带进位加）
MOV   @R0，A   ；存高字节相加结果
MOV   00H，C   ；保存进位 。
```

4.3.2 分支程序

分支结构可以分成单分支、双分支和多分支几种情况：







一、单分支程序

例　求双字节补码。

设在内部RAM的addr1和addr+1单元存有一个双字节数（高位字节存于高地址单元）。编写程序将其读出取补后再存入addr2和addr2+1单元。

首先对低字节取补，然后判其结果是否为全“0”。若是，则高字节取补，否则高字节取反。

```
START：MOV  R0，#addr1 ；原码低字节地址送R0
       MOV  R1，#addr2 ；补码低字节地址送R1
       MOV  A，@R0  ；原码低字节送A
       CPL  A       ；A内容取补
       INC  A
       MOV  @R1，A  ；存补码低字节
       INC  R0      ；调整地址，指向下一单元
       INC  R1
       JZ   ZERO    ；（A）=0时转ZERO
       MOV  A，@R0  ；原码高字节送A
       CPL  A
       MOV  @R1，A  ；高字节反码存入addr2+1单元
       SJMP LOOP1
ZERO：MOV  A，  @R0  ；高字节取补存入addr2+1单元
       CPL  A
       INC  A
       MOV  @R1，A
LOOP1: RET
```

二、双分支程序

例　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。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  $
```

三、多分支程序

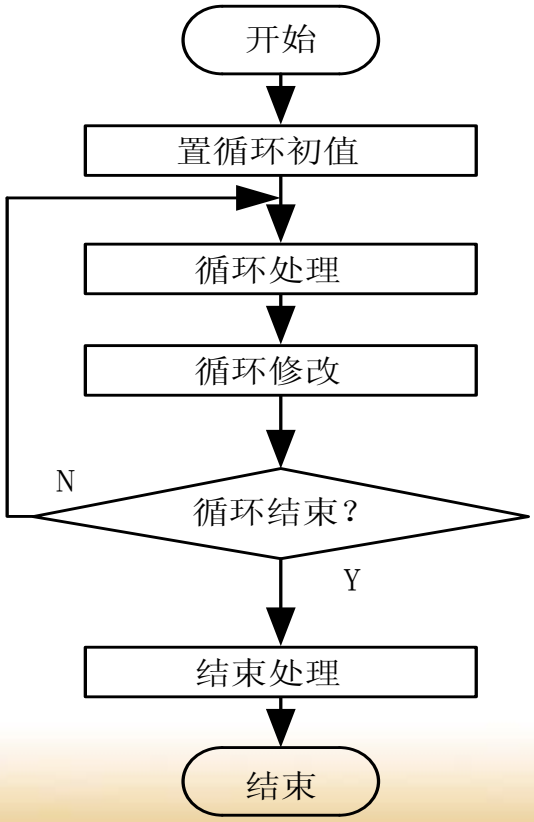
例　根据R7的内容转向相应的处理程序。

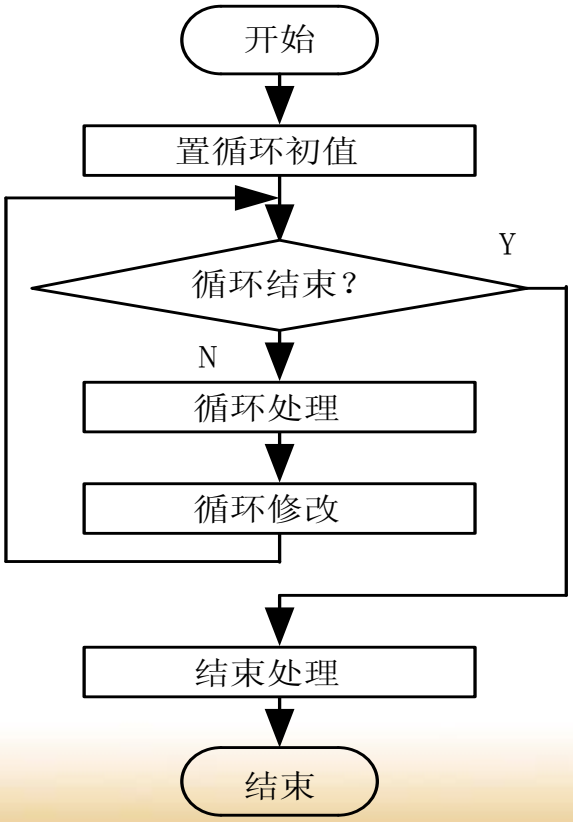
设R7的内容为0～N，对应的处理程序的入口地址分别为PP0～PPN。程序段如下：

```
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
```

4.3.3 循环程序

按某种控制规律重复执行的程序称为循环程序。循环程序有先执行后判断和先判断后执行两种基本结构：





一、先执行后判断

例 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。采用循环计数法实现延时，循环次数可以通过计算获得，并选择先执行后判断的循环结构。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

例　无符号数排序程序。在片内RAM中，起始地址为30H的8个单元中存放有8个无符号数。试对这些无符号数进行升序排序。

•数据排序常用的方法是冒泡排序法。执行时从前向后进行相邻数的比较，如数据的大小次序与要求的顺序不符就将这两个数互换，否则不互换。对于升序排序，通过这种相邻数的互换，使小数向前移动，大数向后移动。从前向后进行一次冒泡（相邻数的互换），就会把最大的数换到最后。再进行一次冒泡，就会把次大的数排在倒数第二的位置。

•设R7为比较次数计数器，初始值为07H，位地址00H为数据互换标志位。

```
START：CLR   00H          ；互换标志清0
       MOV  R7，#07H      ；各次冒泡比较次数
       MOV  R0，#30H      ；数据区首址
 LOOP：MOV  A，@R0        ；取前数
       MOV  2BH，A        ；暂存
       INC   R0
       MOV  2AH，@R0      ；取后数
       CLR   C
        SUBB  A，@R0      ；前数减后数
       JC    NEXT         ；  前数小于后数，不互换
       MOV   @R0，2BH
       DEC   R0
       MOV  @R0，2AH      ；两数交换
       INC   R0           ；准备下一次比较
       SETB  00H          ；置互换标志
 NEXT：DJNZ  R7，LOOP     ；进行下一次比较
      JB    00H，START    ；进行下一轮冒泡
      SJMP  $
```

二、先判断后执行

例　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。由于循环次数事先不知道，但循环条件可以测试到。所以，采用先判断后执行的结构比较适宜。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：… …
```

4.3.4 子程序及其调用

一、子程序的调用

在实际应用中，经常会遇到一些带有通用性的问题，例如：数值转换、数值计算等，在一个程序中可能要使用多次。这时可以将其设计成通用的子程序供随时调用。

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。

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。

二、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。

在执行完子程序，返回继续执行主程序前恢复其原内容，称为现场恢复。保护与恢复的方法有以下两种：

- 在主程序中实现；
- 在子程序中实现。

1、在主程序中实现

示例如下：

```
PUSH   PSW      ；保护现场
PUSH   ACC      ；
PUSH   B        ；
MOV   PSW，#10H ；换当前工作寄存器组
LCALL  addr16     ；子程序调用
POP    B        ；恢复现场
POP    ACC      ；
POP    PSW      ；
… …
```

其特点是结构灵活。

2、在子程序中实现

示例如下：

```
SUB1：PUSH  PSW     ；保护现场
      PUSH  ACC     ；
      PUSH  B       ；
      … …
      MOV  PSW，#10H ；换当前工作寄存器组
      … …
      POP   B       ；恢复现场
      POP   ACC     ；
      POP   PSW     ；
      RET
```

其特点是程序规范、清晰。

注意，无论哪种方法保护与恢复的顺序要对应。

三、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。在调用子程序时，主程序应通过某种方式把有关参数（即子程序的入口参数）传给子程序，当子程序执行完毕后，又需要通过某种方式把有关参数（即子程序的出口参数）传给主程序。在80C51单片机中，传递参数的方法有三种：

1、利用累加器或寄存器

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。

例　编写程序，实现c=a2+b2 。设a，b，c分别存于内部RAM的30H，31H，32H三个单元中。程序段如下：

```
START：MOV   A，30H     ；取a
       ACALL  SQR       ；调用查平方表
       MOV   R1，A      ；a2 暂存于R1中
       MOV   A，31H     ；取b
        ACALL  SQR      ；调用查平方表
       ADD    A，R1     ；a2+b2 存于A中
       MOV   32H，A     ；存结果
       SJMP   $
SQR：  MOV   DPTR，#TAB  ；子程序
       MOVC  A，@A+DPTR；
       RET
TAB：  DB   0，1，4，9，16，25，36，49，64，81
```

2、利用存储器

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。

例　将R0和R1指向的内部RAM 中两个3字节无符号整数相加，结果送到由R0指向的内部RAM中。入口时，R0和R1分别指向加数和被加数的低位字节；出口时，R0指向结果的高位字节。低字节在高地址，高字节在低地址。

实现程序：

```
NADD：MOV   R7，#3      ；三字节加法
       CLR    C          ；
NADD1：MOV   A，@R0     ；取加数低字节
       ADDC   A，@R1     ；被加数低字节加A
       MOV   @R0，A      ；
       DEC    R0
       DEC    R1
       DJNZ   R7，NADD1
       INC    R0
       RET
```

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，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，再用POP指令从堆栈中弹出数据。

例　把内部RAM中20H单元中的1个字节十六进制数转换为2位ASCII码，存放在R0指示的两个单元中。

```
MAIN：MOV   A，20H    ；
      SWAP   A
      PUSH   ACC      ；参数入栈
      ACALL  HEASC
      POP    ACC
      MOV   @R0，A  ；存高位十六进制数转换结果
      INC    R0       ；修改指针
      PUSH   20H      ；参数入栈
      ACALL  HEASC
      POP    ACC
      MOV   @R0，A  ；存低位十六进制数转换结果
      SJMP   $
```

```
HEASC：MOV   R1，SP    ；借用R1为堆栈指针
       DEC    R1
       DEC    R1              ；R1指向被转换数据
       XCH   A，@R1           ；取被转换数据
       ANL   A，#0FH          ；取一位十六进制数
       ADD   A，#2            ；所加值为MOVC与DB间字节数
       MOVC  A，@A+PC ；查表
       XCH    A，@R1          ；1字节指令，存结果于堆栈
       RET                    ；1字节指令
ASCTAB：DB  30H，31H，32H，33H，34H，35H，36H，37H
       DB   38H，39H，41H，42H，43H，44H，45H，46H
```

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。

4.4 常用程序举例

4.4.1 算术运算程序

一、多字节数的加、减运算

80C51单片机的指令系统提供的是字节运算指令，所以在处理多字节数的加减运算时，要合理地运用进位（借位）标志。

例 多字节无符号数的加法。

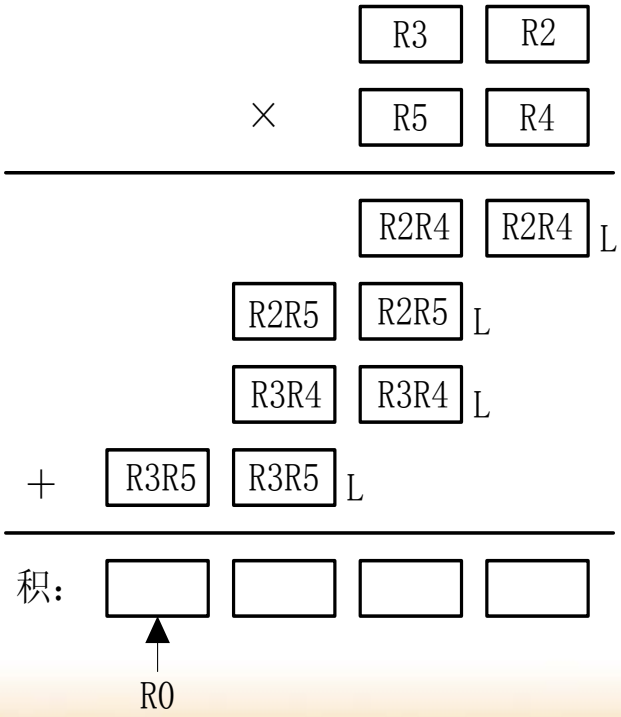
设两个N字节的无符号数分别存放在内部RAM中以DATA1和DATA2开始的单元中。相加后的结果要求存放在DATA2数据区。

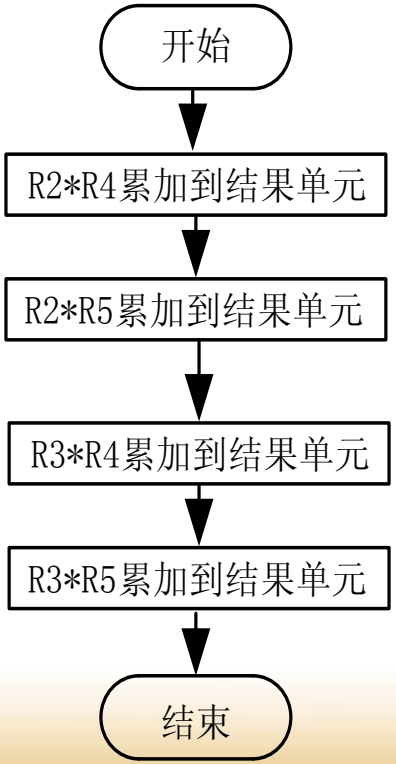
```
        MOV  R0，#DATA1；
        MOV  R1，#DATA2；
        MOV  R7，#N    ；  置字节数
        CLR  C              ；
LOOP：  MOV  A，@R0    ；
        ADDC A，@R1    ；求和
        MOV  @R1，A    ；存结果
        INC  R0             ；修改指针
        INC  R1         ；
        DJNZ R7，LOOP ；
```

二、多字节数乘法运算

例　双字节无符号数的乘法。

设双字节的无符号被乘数存放在R3、R2中，乘数存放在R5、R4中，R0指向积的高位。





例　多字节无符号数的减法。

设两个N字节的无符号数分别存放在内部RAM中以DATA1和DATA2开始的单元中。相减后的结果要求存放在DATA2数据区。

```
        MOV   R0，#DATA1 ；
        MOV   R1，#DATA2 ；
        MOV   R7，#N     ；置字节数
        CLR   C          ；
LOOP：  MOV   A，@R0     ；
        SUBB  A，@R1     ；求差
        MOV   @R1，A     ；存结果
        INC   R0         ；修改指针
        INC   R1         ；
        DJNZ  R7，LOOP   ；
```

主程序段如下：

```
MULTB：MOV  R7，#04    ；结果单元清0
  LOOP：MOV   @R0，#00H  ；
        DJNZ   R7，LOOP   ；
        DEC    R0
        ACALL  BMUL      ；
        SJMP   $
```

另有2段子程序：

BMUL

RADD（在BMUL中被调用）

先看子程序段：

```
RADD：ADD   A，@R0    ；
      MOV   @R0，A    ；
      MOV   A，B      ；
      INC   R0        ；
      ADDC  A，@R0    ；
      MOV   @R0，A    ；
      INC   R0        ；
      MOV   A，@R0    ；
      ADDC  A，#00H   ；加进位
      MOV   @R0，A    ；
      RET
```

```
BMUL：MOV   A，R2      ；
      MOV   B，R4      ；
      MUL    AB        ；低位乘
      ACALL  RADD      ；
      MOV   A，R2      ；
      MOV   B，R5      ；
      MUL    AB        ；交叉乘
      DEC    R0        ；
      ACALL  RADD      ；
      MOV   A，R4      ；
      MOV   B，R3      ；
      MUL   AB         ；交叉乘
      DEC    R0        ；
      DEC    R0        ；
      ACALL  RADD      ；
      MOV   A，R5      ；
      MOV   B，R3      ；
      MUL   AB         ；高字节乘
      DEC    R0        ；
      ACALL  RADD      ；
      DEC    R0
      RET
```

4.4.2 码型转换

一、十六进制数与ASCII码间的转换

十六进制数与ASCII码的对应关系如表所示。当十六进制数在0～9之间时，其对应的ASCII码值为该十六进制数加30H；当十六进制数在A～F之间时，其对应的ASCII码值为该十六进制数加37H。

十六进制数与 ASCII 码的关系表

16 进制数	ASCII 码	16 进制数	ASCII 码	16 进制数	ASCII 码	16 进制数	ASCII 码
0	30H	4	34H	8	38H	C	43H
1	31H	5	35H	9	39H	D	44H
2	32H	6	36H	A	41H	E	45H
3	33H	7	37H	B	42H	F	46H

例　将1位十六进制数转换成相应的ASCII码。

设十六进制数存放在R0中，转换后的ASCII 码存放于R2中。实现程序如下：

```
HASC：MOV  A，R0      ；取4位二进制数
      ANL  A，#0FH    ；屏蔽掉高4位
      PUSH ACC        ；4位二进制数入栈
      CLR  C          ；清进（借）位位
      SUBB A，#0AH    ；用借位位的状态判断该数在0～9还是A～F之间
      POP  ACC        ；弹出原4位二进制数
      JC   LOOP       ；借位位为1，跳转至LOOP
      ADD  A，#07H    ；借位位为0，该数在A～F之间，加37H
LOOP：ADD  A，#30H    ；该数在0～9之间，加30H
      MOV  R2，A      ；ASCII码存于R2
      RET
```

例　将多位十六进制数转换成ASCII码。

设地址指针R0指向十六进制数低位，R2中存放字节数，转换后地址指针R0指向十六进制数的高位。R1指向要存放的ASCII码的高位地址。实现程序如下：

```
HTASC：MOV  A，@R0    ；取低4位二进制数
      ANL   A，#0FH    ；
      ADD   A，#15     ；偏移量修正
      MOVC  A，@A+PC   ；查表
      MOV   @R1，A     ；存ASCII码
      INC   R1         ；
      MOV   A，@R0     ；取十六进制高4位
      SWAP  A
      ANL   A，#0FH    ；
      ADD   A，#06H    ；偏移值修正
      MOVC  A，@A+PC   ；
      MOV   @R1，A
      INC   R0         ；指向下一单元
      INC   R1         ；
      DJNZ  R2，HTASC  ；字节数存于R2
      RET
ASCTAB：DB  30H，31H，32H，33H，34H，35H，36H，37H
      DB  38H，39H，41H，42H，43H，44H，45H，46H
```

二、BCD码与二进制数之间的转换

在计算机中，十进制数要用BCD码来表示。通常，用四位二进制数表示一位BCD码，用1个字节表示2位BCD码（称为压缩型BCD码）。

例　双字节二进制数转换成BCD码。

设（R2R3）为双字节二进制数，（R4R5R6）为转换完的压缩型BCD码。

十进制数B与一个8位的二进制数的关系可以表示为：

只要依十进制运算法则，将 bi（i＝7，6，… …，1，0）按权相加，就可以得到对应的十进制数B。（逐次得到：b7×20；b7×21＋b6×20；b7×22＋b6×21＋b5×20；…）。

```
DCDTH：CLR   A       ；
       MOV  R4，A    ；R4清0
       MOV  R5，A    ；R5清0
       MOV  R6，A    ；R6清0
       MOV  R7，#16  ；计数初值
LOOP：CLR   C       ；
       MOV   A，R3   ；
       RLC   A       ；
       MOV   R3，A   ；R3左移一位并送回
       MOV   A，R2   ；
       RLC   A       ；
       MOV   R2，A   ；R2左移一位并送回
       MOV   A，R6   ；
       ADDC  A，R6   ；
       DA    A       ；
       MOV   R6，A   ；（R6）乘2并调整后送回
       MOV   A，R5   ；
       ADDC  A，R5   ；
       DA    A       ；
       MOV   R5，A   ；（R5）乘2并调整后送回
       MOV   A，R4   ；
       ADDC  A，R4   ；
       DA    A       ；
       MOV   R4，A   ；（R4）乘2并调整后送回
       DJNZ  R7，LOOP；
```

思考题及习题

1、80C51单片机汇编语言有何特点？

2、利用80C51单片机汇编语言进行程序设计的步骤如何？

3、常用的程序结构有哪几种？特点如何？

4、子程序调用时，参数的传递方法有哪几种？

5、什么是伪指令？常用的伪指令功能如何？

6、设被加数存放在内部RAM的20H、21H单元，加数存放在22H、23H单元，若要求和存放在24H、25H中，试编写出16位数相加的程序。

7、编写一段程序，把外部RAM中1000H~1030H的内容传送到内部RAM的30H~60H中。

8、编写程序，实现双字节无符号数加法运算，要求（R1R0）+（R7R6）→（61H60H）。

9、若80C51的晶振频率为6MHz，试计算延时子程序的延时时间。。

```
DELAY：MOV  R7，#0F6H
  LP：MOV  R6，#0FAH
      DJNZ  R6，$
      DJNZ  R7，LP
      RET
```

10、在内部RAM 的21H单元开始存有一组单字节不带符号数，数据长度为30H，要求找出最大数存入BIG单元。

11、编写程序，把累加器A中的二进制数变换成3位BCD码，并将百、十、个位数分别存放在内部RAM的50H、51H、52H中。

12、编写子程序，将R1中的2个十六进制数转换为ASCII码后存放在R3和R4中。

13、编写程序，求内部RAM中50H~59H十个单元内容的平均值，并存放在5AH单元。