第4章　汇编语言程序设计

（课时：12学时）

教学目的

- 了解汇编语言程序设计的特点。
- 掌握汇编语言程序的基本结构及简单应用的设计方法。
- 用循环程序设计一个控制信号灯的程序。
- 用循环程序设计一个控制步进电动机的程序。
- 用分支程序设计一个控制汽车信号灯的程序。
- 用分支程序设计一个控制水塔水位的程序。

学习重点和难点

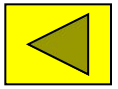
- 分支程序、循环程序的设计特点。
- 子程序的编写和应用。

第4章　汇编语言程序设计

- 4.1　程序设计概述
- 4.2　顺序程序设计
- 4.3　循环程序设计
- 4.4　分支程序设计
- 4.5　子程序设计
- 4.6　查表程序设计
- 本章小结
- 习题

4.1　程序设计概述

- 4.1.1　程序设计语言
- 4.1.2　汇编语言源程序的编辑与汇编
- 4.1.3　汇编语言程序的基本结构
- 4.1.4　程序设计方法和技巧



4.1.1　程序设计语言

1. 机器语言(Machine Language)

这是一种用二进制代码“0”和“1”表示指令和数据的程序设计语言。计算机只能识别二进制代码，这种语言是能被计算机直接识别和执行的机器级语言。

特点：机器语言能够被计算机立即识别并加以执行，具有执行速度快、占用内存少等优点。但对于使用者来说，用机器语言编写程序具有编写难、识别难、记忆难、查错难、交流难等缺点。

4.1.1　程序设计语言

2. 汇编语言(Assembly Language)

汇编语言是一种用助记符来表示的面向机器的程序设计语言。不同的机器所使用的汇编语言一般是不同的。但计算机的CPU不能直接识别汇编语言，所以计算机不能立即执行汇编语言程序。用汇编语言编写的源程序，在由计算机执行之前，必须将它翻译成机器语言程序。

特点：这种语言弥补了机器语言的不足，用汇编语言编写程序比用机器语言方便、直观、易懂、易用、易记。可以编写出结构紧凑、运行时间精确的程序。所以，这种语言非常适合于实时控制的需要。

4.1.1　程序设计语言

3. 高级语言(High-Level Language)

高级语言是面向过程并能独立于计算机硬件结构的通用程序设计语言，是一种接近人类语言和数学表达式的计算机语言。比如：**BASIC**、**FORTRAN**、**COBOL**、**PASCAL**、**C**语言等。高级语言不能被计算机直接识别和执行，需要用编译程序或解释程序将高级语言编写的源程序翻译为机器语言。

特点：它比汇编语言易学、易懂，具有通用性强、易于移植等优点。高级语言的语句功能强，它的一条语句往往相当于许多条指令，因而用于翻译的程序要占用较多的存储空间，而且执行时间长，且不易精确掌握，故在高速实时控制中一般是不适用的。

4.1.2　汇编语言源程序的编辑与汇编

基本概念

- 在目前单片机的开发应用中，经常采用C语言和汇编语言共同编写程序。要想很好地掌握和应用单片机首先要掌握汇编语言。
- 汇编语言是面向机器的程序设计语言，对于CPU不同的单片机，其汇编语言一般是不同的。用汇编语言编写的程序称为汇编语言源程序 。
- 汇编语言源程序是由汇编语言语句构成的。汇编语言语句可分为两大类：指令性语句和指示性语句。
- 指令性语句是由指令组成的由CPU执行的语句，
- 指示性语句是由伪指令组成的，它不被CPU执行，用来告诉汇编程序如何对程序进行汇编的指令；由于它不能生成机器语言，故又被称为伪指令语句。

4.1.2　汇编语言源程序的编辑与汇编

1. 指令性语句格式

[标号:] 操作码助记符　[目的操作数] [,源操作数] [;注释]

- 每条汇编语句一般由若干部分组成，每一部分称为一个字段。
- 每个字段之间应该严格地用分界符加以分隔。
- 分界符包括冒号、空格符、逗号、分号等。标号段与操作码之间要加冒号“:”；操作码与操作数之间要用空格相隔；各操作数之间要用逗号“，”相隔；操作数与注释段之间要加分号“；”相隔。

4.1.2　汇编语言源程序的编辑与汇编

2.伪指令的指示性语句格式

[标号:] 伪操作　操作数[,操作数,.....] [;注释]

- 伪指令不是真正的指令，是在汇编时供汇编程序识别的指令，又称为汇编指令。
- 它不属于指令系统，也无对应的机器码，只是用来对汇编过程进行某种控制。利用伪指令告诉汇编程序如何进行汇编，为编程提供方便。

4.1.2　汇编语言源程序的编辑与汇编

3. 汇编语言源程序的汇编

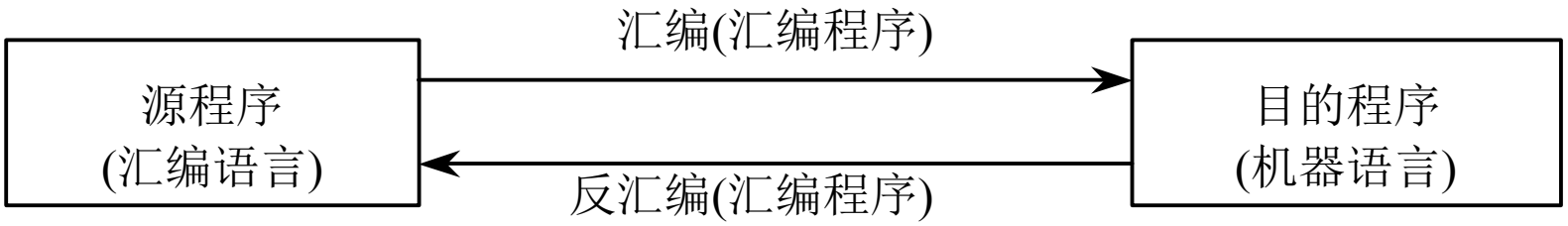
汇编语言源程序必须要转换为机器码(即目的程序)，计算机才能执行，这个转换过程称为汇编。

汇编语言源程序的汇编可分为手工汇编和机器汇编两类。

- 手工汇编是指用人脑通过查指令编码表(见附录中的指令表)把汇编语言源程序翻译成机器码的过程，又称为人工汇编。
- 机器汇编是用机器代替人脑并由专门的程序来进行的，这种程序称为汇编程序(不同的指令系统汇编程序不同)。机器汇编由计算机自动完成，汇编程序把用汇编语言编写的源程序翻译成由机器语言表示的目的程序。
- 反汇编是在分析程序存储器已有的程序时，将机器语言翻译成汇编语言的转换过程。

4.1.2　汇编语言源程序的编辑与汇编

源程序、汇编程序和目的程序之间的关系如下图所示



4.汇编语言源程序的编辑

汇编语言源程序一般在微机上借助编辑软件进行编写，可供使用的编辑工具有许多，如行编辑软件、屏幕编辑软件等。

常用伪指令简介（一）

1. ORG(Origin)汇编起始指令

ORG是程序汇编起始地址定位伪指令，

功能：是规定对汇编语言源程序进行汇编时，目的程序在程序存储器中存放的起始地址。

格式：[标号:] ORG　16位地址或标号

注意：在一个源程序中，可多次使用ORG指令，以规定不同程序段的起始位置，地址应从小到大顺序排列，不允许重叠。

例如：

```
ORG  1000H
MOV  A,#12H    ;该指令的机器码是74H、12H
ADD  A,#34H    ;该指令的机器码是24H、34H
```

在上述源程序中，第一条指令的首字节74H存放到程序存储器的1000H地址单元中，其他字节和后续指令的数据顺序存放到后面的存储单元中。

常用伪指令简介（二）

2. END(End)汇编结束指令

END是汇编语言程序结束伪指令。

功能：是表示程序已结束，汇编程序对**END**后面的指令不再汇编。

格式：**[标号:] END**

注意：在一个源程序中，只能有一条**END**指令，而且必须放在整个程序的末尾。

常用伪指令简介（三）

3. EQU(Equate)赋值指令

EQU是赋值(也称等值)伪指令 。

功能：把操作数段中的数据或地址赋值给标号字段中的字符名称 。

格式：字符名称　**EQU**　数值或汇编符号

注意：字符名称必须先赋值后使用，故**EQU**指令通常放在源程序的开头。 **EQU**可定义**8**位或**16**位的数据或地址，

例如：
```
ABC     EQU   30H    ;AB与30H等值
ACB     EQU   R3     ;AC与R3等值
MOV   A,ABC          ;把片内RAM30H单元中的数据送入A中
MOV   A,ACB          ;把R3中的数据送入累加器A中
```

常用伪指令简介（四）

4. DATA(Data)数据地址赋值指令

DATA是数据地址赋值伪指令 。

功能：把操作数段中的表达式的值赋给标号字段中的字符名称。

格式：字符名称　DATA　表达式

注意：DATA指令功能与EQU指令类似，它们的主要区别如下：

- DATA定义的字符名称可以先使用后定义，DATA指令可以放在源程序的任何位置。
- DATA只能用来定义8位的数据或地址。
- EQU可以把汇编符号赋给字符名称，而DATA只能把数据赋给字符名称。
- DATA可以把表达式的值赋给字符名称，这个表达式是可以进行求值运算的。

常用伪指令简介（五）

5. XDATA数据地址赋值指令

XDATA是数据地址赋值伪指令 。

功能：把操作数段中的表达式的值赋给标号字段中的字符名称 。

格式：字符名称　**XDATA**　表达式

注意：**XDATA**指令功能与**DATA**指令类似，它们的主要区别是**XDATA**可定义**16**位的数据或地址。

常用伪指令简介（六）

6. BIT(Bit)位地址赋值指令

BIT是位地址赋值伪指令 。

功能：把位地址赋给字符名称 。

格式：字符名称　BIT　位地址

例如：
```
AB    BIT   30H    ;AB与30H等值
AC    BIT   P1.0   ;AC与P1.0等值
      MOV   C,AB   ;把位地址区30H单元中的数据送入
                   位累加器C中
      CLR   AC     ;把P1.0中的内容清零
```

常用伪指令简介（七）

7. DB(Define Byte)定义字节指令

DB是定义字节伪指令 。

功能：从程序存储器指定地址单元开始存放若干个字节的数值或ASCII码字符 。

格式：[标号:] DB　字节数据或ASCII码字符

注意：多个字节数据或ASCII码字符之间要用逗号相隔，DB指令常用于定义8位的数据常数表。

例如：
```
ORG  1000H
TAB:  DB     50H,60,'A'
      DB     01010111B,'6'
```

常用伪指令简介（八）

8. DW(Define Word)定义字指令

DW是定义字伪指令 。

功能：从程序存储器指定地址单元开始存放若干个字的数值 。

格式：[标号:] DW　字节数据或ASCII码字符

注意：多个字数据之间要用逗号相隔，DW指令常用于定义16位的地址表。

例如： ORG 1000H

TAB: DW 20H,50H,00H,60H

常用伪指令简介（九）

9. DS(Define Space)定义存储空间指令

DS是定义存储空间伪指令 。

功能：从程序存储器指定地址单元开始保留表达式的值所规定的存储单元 。

格式：[标号:] DS　表达式

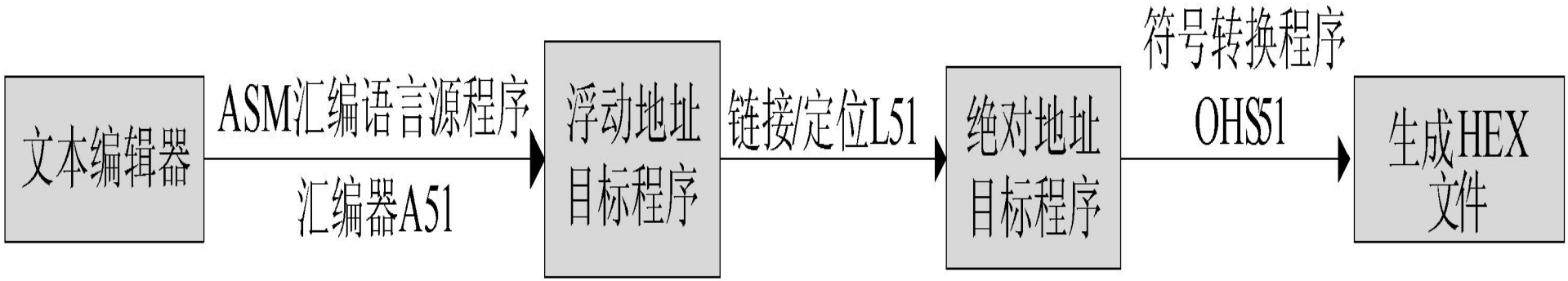
例如：

```
ORG  1000H
TAB:  DS  06H
      DB  25H,35H
```

在上述源程序中，程序存储器从1000H单元开始保留6个单元，1006H单元存放25H，1007H单元存放35H。

汇编语言程序的编辑与汇编

- 源程序的编写要以MCS-51单片机汇编语言指令和伪指令为基础，灵活运用指令完成确定的算法或解题思路。具体体工作过程如下图所示



源程序编辑和汇编过程

汇编语言源程序的格式

- 源程序基本上由主程序、子程序、中断服务子程序组成。编制汇编语言源程序根据MCS-51单片机ROM的出厂内部定义，一般按这样的主框架编制：

```
                              ; 程序变量定义区
1 SDA  BIT  P1.3              ; 定义SDA位变量
2 IO   EQU  P0                ; 定义I/O等值P0口
3 ByteCon  DATA  30H          ; 定义字节变量ByteCon
                              ; 程序主体部分
4 ORG       0000H                 ; 程序段从0000H单元开始存放
5 LJMP      MAIN                  ; 跳到主程MAIN
6 ORG       0003H                 ; 从0003H开始存放程序段
7 LJMP      INTERUPT1             ; 跳到外部中断0处理子程序
```

汇编语言源程序的格式

```
8  ORG 0030H       ；从0030H开始存放程序段
9  MAIN:                    ；主程序标号说明
10 MOV SP，#30H   ；设置堆栈指针，可以大于30H
11 LCALL      INITIATE   ；调用初始化子程
12     FCY:                 ；控制程序循环标号
   LCALL      SUB          ；调用功能子程序
13 LJMP       FCY          ；跳到FCY构成循环
14 ORG xxxx         ；以下功能程序的存放地址
15     INITIATE: …   …   ；初始化子程序标号
16 RET              ；子程序返回
17     SUB:   …   …         ；功能子程序标号
18 RET              ；子程序返回
19  INTERUPT1: …   …   ；外部中断0功能程序
20 RETI                     ；中断返回
21     TABLE:               ；表的标号
22 DB   00H，01H          ；表的数据
       END                  ；源程序结束，停止汇编。
```

汇编语言源程序的格式

- 第1～3行：把一些符号或变量定义成通俗的符号。
- 第4、6、8、14行：表示程序存储的开始地址。
- 第5行：使CPU在执行程序时，从0000H跳过各中断源的入口地址，主程序以跳转的目标地址作为起始地址开始编写。
- 第6行：中断服务程序的存储地址。
- 第9、12、15、17、19行：为程序语句标号。
- 第10行：设置堆栈指针一般最小设30H，栈区够用还可以增大。
- 第21、22行：为查表指令的表。

4.1.3　汇编语言程序的基本结构

汇编语言程序具有四种结构形式，即顺序结构、循环结构、分支结构和子程序结构。

1. 顺序程序

顺序程序是一种最简单、最基本的程序结构，又称为简单程序或直线程序。程序按顺序一条一条地执行指令，程序流向不变。

2. 循环程序

循环程序是把需要多次重复执行的某段程序，利用条件转移指令反复转向执行，可减小整个程序的长度，优化程序结构。

循环程序一般由循环初始化、循环处理、循环控制和循环结束四部分组成。

4.1.3　汇编语言程序的基本结构

3. 分支程序

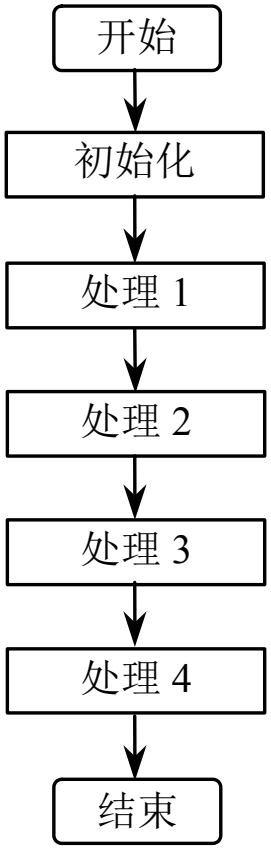
分支程序是根据条件进行判断决定程序的执行，满足条件则进行程序转移，不满足条件就顺序执行程序。判断是通过条件转移指令实现的。分支程序又分为单分支结构和多分支结构。

4. 子程序

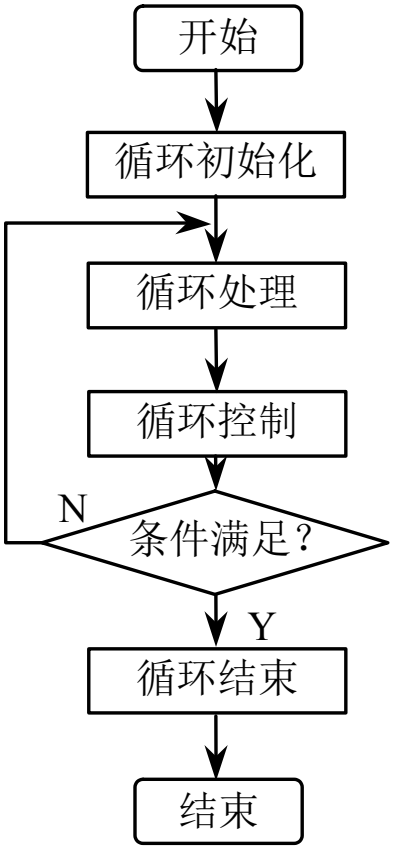
子程序是指完成某一确定任务并能被其他程序反复调用的程序段。

使用子程序可以减小整个程序的长度，实现模块化程序结构。

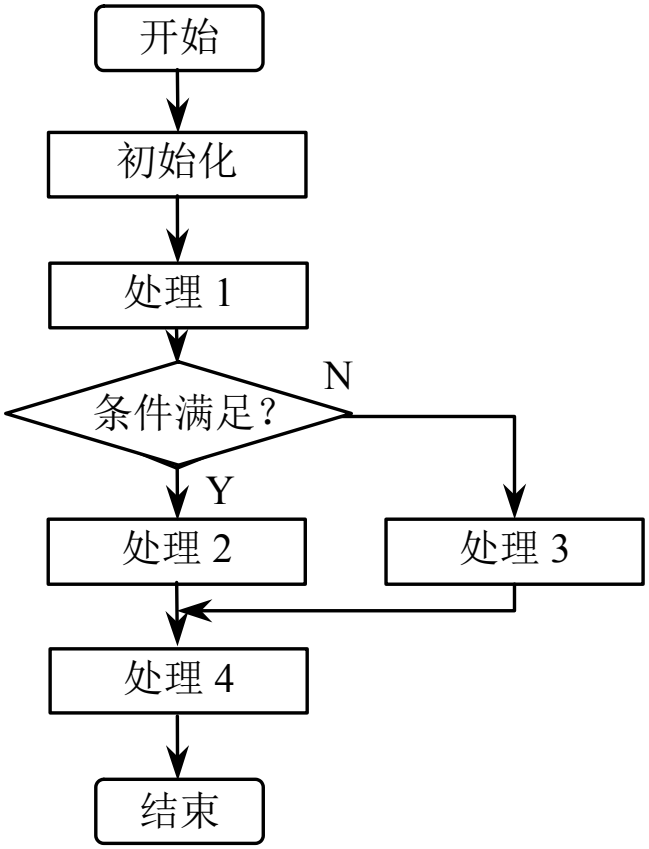
4.1.3 汇编语言程序的基本结构



顺序程序流程图



循环程序流程图



分支程序流程图

4.1.4　程序设计方法和技巧

1. 程序设计的一般步骤

(1) 分析工作任务，明确要达到的工作目的、技术指标等。

(2) 确定解决问题的算法。算法就是如何将实际问题转化成程序模块来处理，要对不同的算法进行分析、比较，找出最适宜的算法。

(3) 画程序流程图。其图形的符号规定均与高级语言流程图相同，如桶形框表示程序的开始或结束，矩形框表示需要进行的工作，菱形框表示需要判断的事情，指向线表示程序的流向等。

(4) 分配内存工作单元，确定程序与数据的存放地址。

(5) 编写源程序。

(6) 上机调试、修改源程序。

4.1.4　程序设计方法和技巧

2. 程序设计的一般原则

- 按照尽可能使程序简短和缩短运行时间两个原则编写程序。
- 应用程序一般都由一个主程序(包括若干个功能模块)和多个子程序构成，即采用模块化的程序设计方法。
- 每一功能模块或子程序都能完成一个明确的任务，实现某个具体功能，如检测输入信号、码制转换、输出控制信号、发送数据、接收数据、延时、显示、打印等。

4.1.4　程序设计方法和技巧

3. 模块化程序设计方法的特点

- 单个模块结构的程序功能单一，易于编写、调试和修改。
- 对程序的局部修改，可以使无关的部分保持不变。
- 程序可读性好，便于功能扩展和版本升级。
- 对于使用频繁的子程序可以建立子程序库，便于多个模块调用。
- 可实现多人同时进行程序的编写和调试工作，缩短程序编写时间。

4.1.4　程序设计方法和技巧

4. 划分模块应遵循的原则

- 高内聚性。每个模块应具有独立的功能，能产生一个明确的结果。
- 低耦合性。模块之间的控制耦合应尽量简单，数据耦合应尽量少。控制耦合是指模块进入和退出的条件及方式，数据耦合是指模块间的信息交换(传递)方式、交换量的多少及交换的频繁程度。
- 模块长度适中。模块语句的长度为20～100条的范围较合适。模块太长时，分析和调试比较困难，失去了模块化程序结构的优越性；过短则模块的连接太复杂，信息交换太频繁。

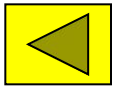
4.1.4　程序设计方法和技巧

5. 程序设计的一般技巧

- 尽量采用循环结构和子程序结构。这样可以使程序的总容量大大减少，提高程序的效率，节省内存。
- 尽量少用无条件转移指令。这样可以使程序条理更加清楚，从而减少错误。
- 对于通用的子程序，除了用于存放子程序入口参数的寄存器外，子程序中用到的其他寄存器的内容应压入堆栈，即保护现场。一般不必把标志寄存器压入堆栈。
- 在中断处理程序中，除了要保护中断处理程序中用到的寄存器外，还要保护标志寄存器。
- 用累加器传递入口参数或返回参数比较方便，在子程序中，一般不必把累加器内容压入堆栈。

4.2　顺序程序设计

- 4.2.1　顺序程序设计方法
- 4.2.2　顺序程序设计实例



4.2.1　顺序程序设计方法

- 顺序结构程序是最简单、最基本的程序。要设计出高质量的程序需要掌握一定的技巧，需要熟悉指令系统，正确地选择指令，掌握程序设计的基本方法和技巧，以达到提高程序执行效率、减少程序长度、最大限度地优化程序的目的。
- 顺序程序的特点和设计方法。
- 结构比较单一和简单，按程序编写的顺序依次执行，中间没有任何分支，程序流向不变。
- 数据传送指令使用得较多，没有控制转移类指令。
- 作为复杂程序的某个组成部分，如循环结构程序中需多次重复执行的某段程序(称为循环处理)。

4.2.2　顺序程序设计实例

【例1】 有两个**6**位**BCD**码分别存放在片内**RAM 30H**、**31H**、**32H**单元和**40H**、**41H**、**42H**单元内，求它们的和并将和存放到片内**RAM 50H**、**51H**、**52H**单元中。

解： 设定片内**RAM 30H**单元存放高位，片内**RAM 32H**单元存放低位，其他单元与之类同。**BCD**码加法运算后，要用**DA**指令进行调整。

参考程序：

地址	机器码	程序	注释
		ORG 0000H	
0000H	02 00 30	LJMP MAIN	
		ORG 0030H	
0030H	E5 32	MAIN:MOV A,32H	;低2位被加数送入累加器A
0032H	25 42	ADD A,42H	
0034H	D4	DA A	;BCD码调整
0035H	F5 52	MOV 52H,A	;存放和的低2位
0037H	E5 31	MOV A,31H	;中2位被加数送入累加器A
0039H	35 41	ADDC A,41H	;加上低位的进位
003BH	D4	DA A	;BCD码调整
003CH	F5 51	MOV 51H,A	;存放和的中2位
003EH	E5 30	MOV A,30H	;高2位被加数送入累加器A
0040H	35 40	ADDC A,40H	;加上中位的进位
0042H	D4	DA A	;BCD码调整
0043H	F5 50	MOV 50H,A	;存放和的高2位
0045H	80 FE	SJMP $	;暂停
		END	

4.2.2　顺序程序设计实例

【例2】有一个16位二进制负数的补码存放在片内RAM 30H、31H单元内，求它的原码的绝对值并将它存放到片内RAM 40H、41H单元。

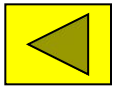
解：设定片内RAM 30H单元存放高位，片内RAM 31H单元存放低位，其他单元与之类同。补码取反后要加1，绝对值要去掉符号位。

参考程序：

地址	机器码	程序	注释
		ORG 0000H	
0000H	02 00 30	LJMP MAIN	
		ORG 0030H	
0030H	E5 31	MAIN: MOV A,31H	;低8位补码送入累加器A
0032H	F4	CPL A	;低8位取反
0033H	24 01	ADD A,#01H	;补码取反后要加1
0035H	F5 41	MOV 41H,A	;存放原码绝对值的低8位
0037H	E5 30	MOV A,30H	;高8位补码送入累加器A
0039H	F4	CPL A	;高8位取反
003AH	34 00	ADDC A,#00H	;加上低8位的进位
003CH	54 7F	ANL A,#7FH	;去掉最高位符号位
003EH	F5 40	MOV 40H,A	;存放原码绝对值的高7位
0040H	80 FE	SJMP $	;暂停
		END	

4.3　循环程序设计

- 4.3.1　循环程序设计方法
- 4.3.2　循环程序设计实例



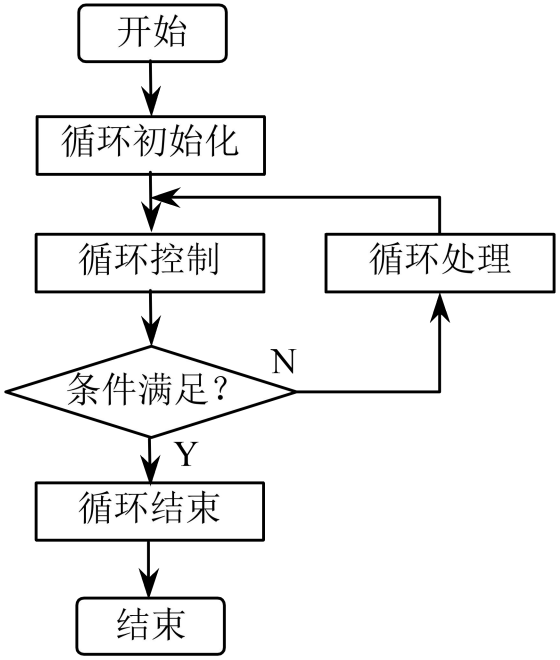
4.3.1　循环程序设计方法

- 循环程序的结构一般包括以下几部分。
- 循环初始化——是进入循环处理前必须要有的一个环节，用于完成循环前的准备工作。循环初始化包括给工作寄存器(或其他存储单元)设置计数初值、地址指针、数据块长度等。
- 循环处理——是需要多次重复执行的程序段。循环处理是循环程序的核心，用于完成主要的计算和操作任务。
- 循环控制——是用条件转移指令控制循环是否继续。每循环一次，根据循环结束条件进行一次判断；当满足条件时，停止循环，继续执行其他程序；否则，再作循环。
- 循环结束——用于存放循环程序的执行结果，同时恢复相关工作单元的初值。

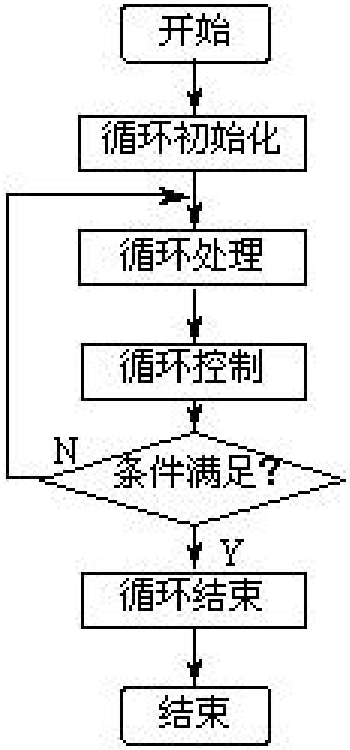
4.3.1　循环程序设计方法

- 循环程序一般有两种编写方法。
- 先循环处理后循环控制(即先处理后判断)，其流程如下图所示。
- 先循环控制后循环处理(即先判断后处理)，其流程如下图所示。
- 循环处理和循环控制构成循环体，若循环程序的循环体内不再包含其他循环程序，则称为单重循环程序。若循环程序的循环体内包含有其他循环程序，则称为多重循环程序，又称为循环嵌套。
- 多重循环程序中的各重循环不能有交叉，不能从外循环跳入内循环，只能外循环内嵌套内循环。两重循环程序流程如下图所示。

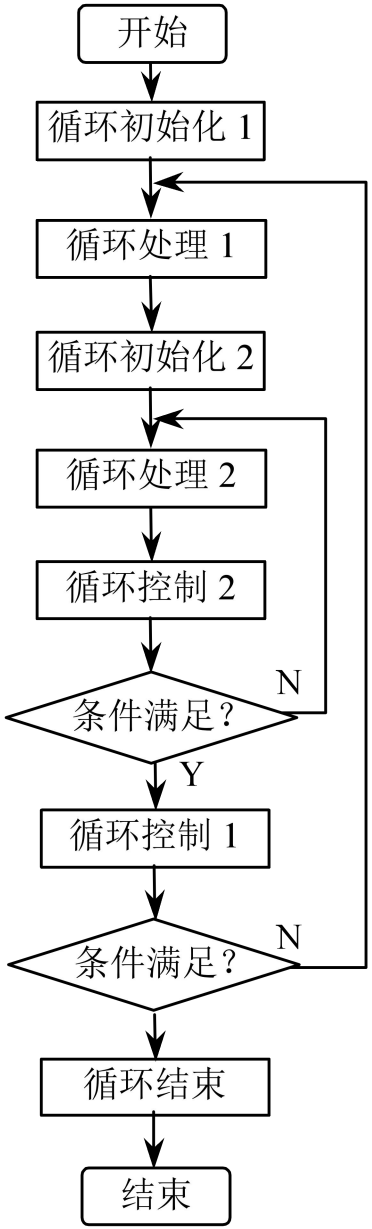
先判断后处理
循环程序流程图



先处理后判断
循环程序流程图



两重循环
程序流程图



4.3.1　循环程序设计方法

- 循环程序的特点和设计方法。
- 程序结构紧凑，占用存储单元较少，程序中间有分支，循环程序本质上是分支程序的一种特殊形式。
- **DJNZ**指令使用得较多，凡是分支程序中可以使用的控制转移类指令，循环程序一般都可以使用。
- 循环控制的形式有多种。计数循环是最常用的一种，它先预置计数初值，再用 **DJNZ**指令控制循环次数；条件循环也是较常用的一种，它先预置结束循环的条件，再用**CJNE**指令、**JZ**指令或**JB**指令控制循环的结束。

4.3.2　循环程序设计实例

【例3】片内RAM中存放有10个数据，首地址为30H，编程将数据块传送到片外RAM以1000H为首地址的存储单元中。

解：该程序是单重循环程序，片内RAM首地址30H、片外RAM首地址1000H和数据块长度10都是循环初始化的内容。

循环控制是对数据块长度进行判断，每传送一个数据，存放数据块长度的寄存器减1；10个数据传送完，存放数据块长度的寄存器内容正好为零，退出循环。

4.3.2 循环程序设计实例

地址	机器码	程序	注释
		ORG 0000H	
0000H	02 01 00	LJMP MAIN	
		ORG 0100H	
0100H	79 30	MAIN: MOV R1,#30H	;置片内RAM地址指针30H
0102H	90 10 00	MOV DPTR,#1000H	;置片外RAM地址指针1000H
0105H	7A 0A	MOV R2,#10	;数据块的长度
0107H	E7	LOOP: MOV A,@R1	;从片内RAM取数据
0108H	F0	MOVX @DPTR,A	;数据传送到片外RAM
0109H	09	INC R1	;修改片内RAM地址指针
010AH	A3	INC DPTR	;修改片外RAM地址指针
010BH	DA FA	DJNZ R2,LOOP	;循环次数未到10次,转移
010DH	80 FE	SJMP $	
		END	

4.3.2　循环程序设计实例

【例4】 P1口做输出口，控制8盏灯(P1口输出低电平时灯被点亮)，编程使灯按以下规律显示：同一时间只有两盏灯点亮，从P1.7、P1.6控制的灯开始，每盏灯闪烁5次，再移向下两盏灯，同样闪烁5次，循环往复，延时时间1s。晶振频率6MHz。

解：主程序是双重循环程序，循环移位是外循环，灯闪烁5次是内循环，内循环程序不能与外循环程序交叉。

延时1S采用三重循环程序。晶振频率为6MHz时，机器周期为2μs，延时程序的延时时间计算方法如下：

{1+[1+(1+(1+1+2)×125+2)×200+2]×5+2}×2μs

=1006036μs=1.006036s

4.3.2 循环程序设计实例

地址	机器码	程序		注释
			ORG 0000H	
0000H	02 00 30		LJMP MAIN	
			ORG 0030H	
0030H	74 5F	MAIN:	MOV A,#5FH	;灯点亮初始状态
0032H	79 05	LP1:	MOV R1,#5	;循环闪烁次数
0034H	F5 90	LP2:	MOV P1,A	
0036H	12 01 00		LCALL DELAY	;延时1s
0039H	75 90 FF		MOV P1,#0FFH	
003CH	12 01 00		LCALL DELAY	;延时1s
003FH	D9 F3		DJNZ R1,LP2	;循环闪烁次数不够5次,继续
0041H	03		RR A	;右移一位
0042H	03		RR A	;再右移一位
0043H	80 ED		SJMP LP1	

地址	机器码	程序		注释
			ORG 0100H	
0100H	7B 05	DELAY:	MOV R3,#5	;延时1s的循环次数
0102H	7C C8	DEL3:	MOV R4,#200	;延时200ms的循环次数
0104H	7D 7D	DEL2:	MOV R5,#125	;延时1ms的循环次数
0106H	00	DEL1:	NOP	
0107H	00		NOP	
0108H	DD FE		DJNZ R5,DEL1	
010AH	DC F9		DJNZ R4,DEL2	
010CH	DB F5		DJNZ R3,DEL3	
010EH	22		RET	;子程序返回
			END	

4.3.2 循环程序设计实例

【例5】 P1口做为输出口控制步进电动机的四相绕组，编写程序，控制步进电动机每2s正向转动一步。晶振频率6MHz。

解：步距角：$\theta_b=360/mZ$ (°)

电机转速：$n=60f/mZ$ (r/min)

上式中：f 为脉冲频率，单位：Hz或步/s。

m 为拍数，本例中为4。

Z 为转子齿数，本例中取5。

拍数m=4，若使用的步进电动机转子齿数Z为5，则步距角θ_b=18°。题目要求步进电动机每2s正向转动一步，即T=2s，则f=0.5Hz，电机转速n=1.5r/min。

用三重循环设计2s的循环程序。晶振频率为6MHz时，机器周期为2μs，延时程序的延时时间计算方法如下：

{1+[1+(1+1+2×123+2)×200+2]×20+2}×2μs

=2000126μs=2.000126s

P1.0～P1.3口作为输出口分别控制步进电动机的四相绕组，步进电动机的控制状态与P1口的控制码的对应关系如下表所示。

控制状态	P1口 控制码	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
						D相	C相	B相	A相
A相、B相绕组通电	03H	0	0	0	0	0	0	1	1
B相、C相绕组通电	06H	0	0	0	0	0	1	1	0
C相、D相绕组通电	0CH	0	0	0	0	1	1	0	0
D相、A相绕组通电	09H	0	0	0	0	1	0	0	1

主程序：

地址	机器码	程序	注释
		ORG 0000H	
0000H	02 00 30	LJMP MAIN	
		ORG 0030H	
0030H	75 90 03	MAIN: MOV P1,#03H	;AB相通电
0033H	12 01 00	LCALL DELAY	
0036H	75 90 06	MOV P1, #06H	;BC相通电
0039H	12 01 00	LCALL DELAY	
003CH	75 90 0C	MOV P1, #0CH	;CD相通电
003FH	12 01 00	LCALL DELAY	
0042H	75 90 09	MOV P1, #09H	;DA相通电
0045H	12 01 00	LCALL DELAY	
0048H	80 E6	SJMP MAIN	;重复循环

延时子程序：

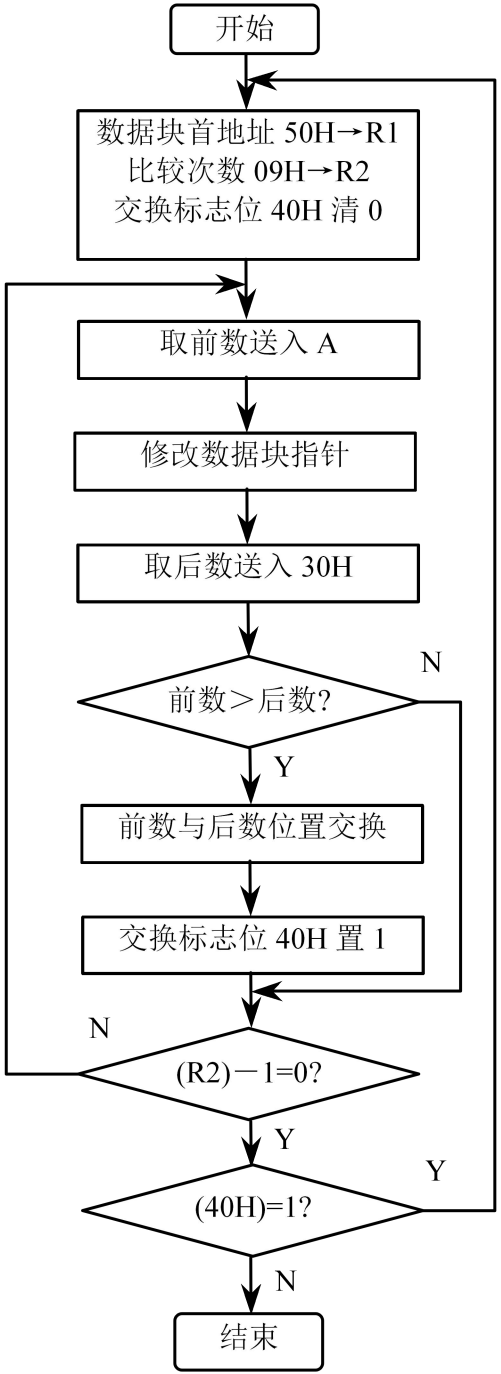
地址	机器码	程序		注释
			ORG 0100H	
0100H	7B 14	DELAY:	MOV R3,#20	;延时2s的循环次数
0102H	7F C8	LP1:	MOV R7,#200	;延时100ms的循环次数
0104H	7E 7B	LP2:	MOV R6,#123	;延时0.5ms的循环次数
0106H	00		NOP	
0107H	DE FE	LP3:	DJNZ R6,LP3	
0109H	DF F9		DJNZ R7,LP2	
010BH	DB F5		DJNZ R3,LP1	
010DH	22		RET	
			END	;程序结束

4.3.2　循环程序设计实例

【例6】片内RAM从50H单元开始存放了10个无符号数，编程将它们按由小到大的顺序排列。

解：数据排序的方法有很多，本例采用常用的冒泡排序法，又称为两两比较法。

想象把10个数纵向排列，自上而下将存储单元相邻的两个数进行比较，若前数大于后数，则存储单元中的两个数互换位置；若前数小于后数，则存储单元中的两个数保持原来位置。按同样的原则依次比较后面的数据，直到该组数据全部比较完，经过第1轮的比较，最大的数据就像冒泡一样排在了存储单元最末的位置上。经过9轮冒泡，便可完成10个数据的排序。

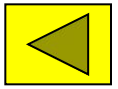


在实际排序中，10个数不一定要经过9轮排序冒泡，可能只要几次就可以了。为了减少不必要的冒泡次数，可以设计一个交换标志，每一轮冒泡的开始将交换标志位清0，在该轮数据比较中若有数据位置互换，则将交换标志位置1；每轮冒泡结束时，若交换标志位仍为0，则表明数据排序已完成，可以提前结束排序。

```
                            ORG   0000H
0000H    02 01 00           LJMP   MAIN
                            ORG   0100H
0100H    79 50      MAIN:   MOV   R1,#50H      ;置数据块首地址
0102H    7A 09              MOV   R2,#09H      ;置每次冒泡比较次数
0104H    C2 40              CLR   40H          ;交换标志位清0
0106H    E7         LOOP1:  MOV   A,@R1        ;取前数
0107H    09                 INC   R1
0108H    87 30              MOV   30H,@R1      ;取后数
010AH    B5 30 00           CJNE  A,30H,LOOP2  ;比较前数与后数的大小
010DH    40 07      LOOP2:  JC    LOOP3        ;若前数＜后数则转移,不互换
010FH    F7                 MOV   @R1,A        ;大数存放到后数的位置
0110H    19                 DEC   R1
0111H    A7 30              MOV   @R1,30H      ;小数存放到前数的位置
0113H    09                 INC   R1           ;恢复数据指针,准备下一次比较
0114H    D2 40              SETB   40H         ;有互换,标志位置1
0116H    DA EE      LOOP3:  DJNZ   R2,LOOP1    ;若一次冒泡未完,继续进行比较
0118H    20 40 E5           JB    40H,MAIN     ;若有交换,继续进行下一轮冒泡
011BH    80 FE              SJMP   $
                            END
```

4.4　分支程序设计

- 4.4.1　分支程序设计方法
- 4.4.2　分支程序设计实例



4.4.1　分支程序设计方法

- 分支程序是根据程序的要求改变程序的执行顺序，并根据条件对程序的流向进行判断的程序结构。
- 分支程序一般有两个或两个以上的出口。
- 分支程序又分为单分支和多分支结构。

4.4.1　分支程序设计方法

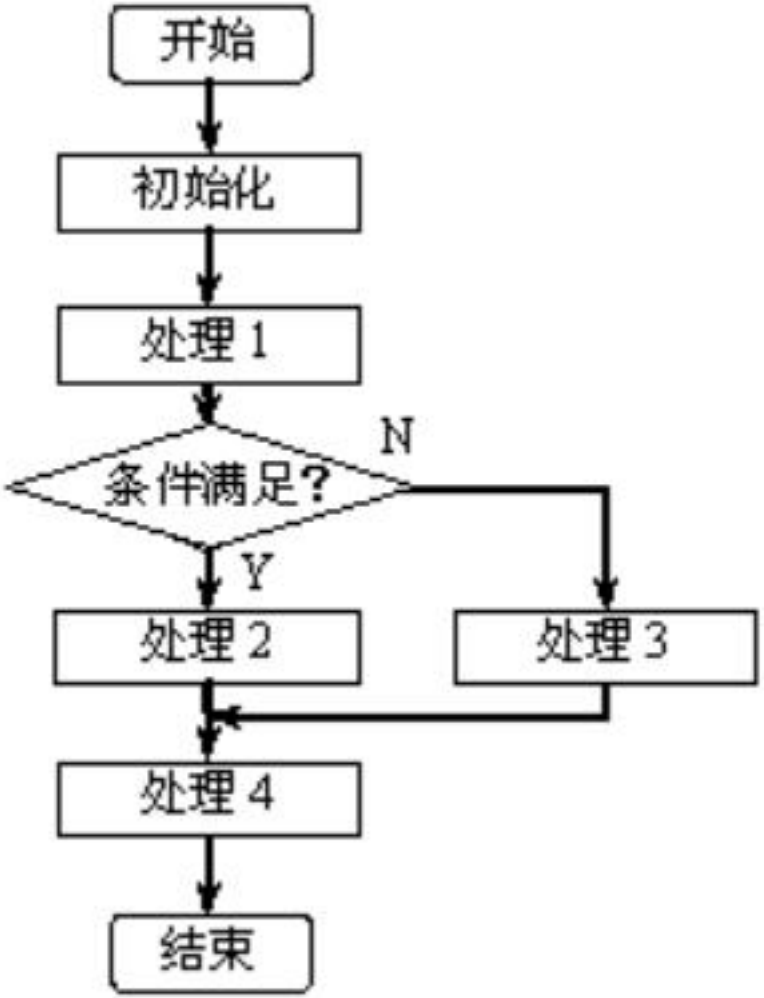
- 分支程序的特点和设计方法。
- 程序中有转移指令包括无条件转移、条件转移和散转指令。
- 分支的出口有两个以上时，形成散转程序，一般用散转指令来实现，设计方法有**4**种。分别是转移指令表法、地址偏移量表法、转向地址表法和利用**RET**指令法。
- 单分支程序一般有一个入口、两个出口，一般用无条件转移和条件转移指令来实现，结构形式有两种。

  一种是当条件满足时，执行处理程序**2**，否则执行处理程序**3**。分支程序流程图如下图**(a)**所示。

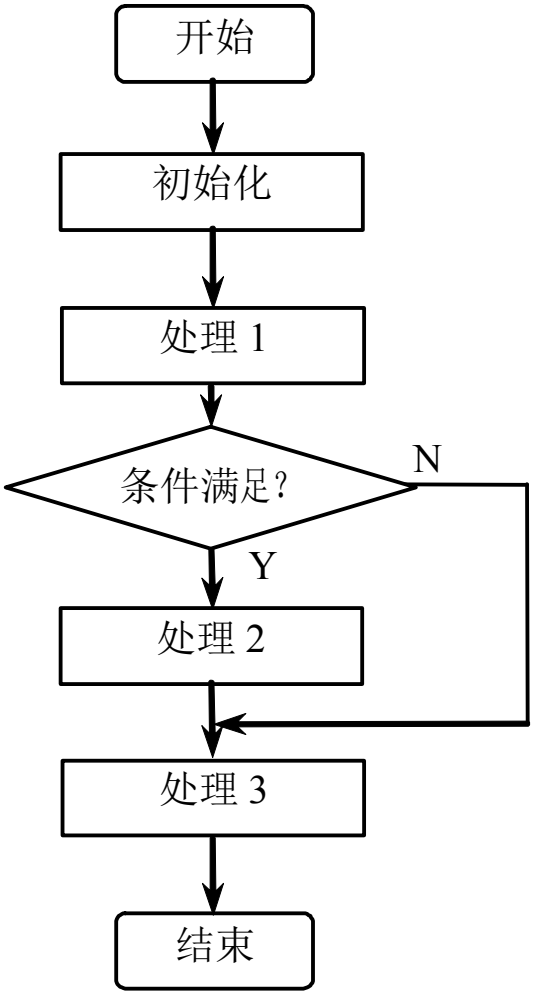
  另一种是当条件满足时，跳过处理程序**2**，直接执行处理程序**3**，否则顺序执行处理程序**2**和处理程序**3**。分支程序流程图如下图**(b)**所示。

4.4.1　分支程序设计方法

(a)　分支程序流程图

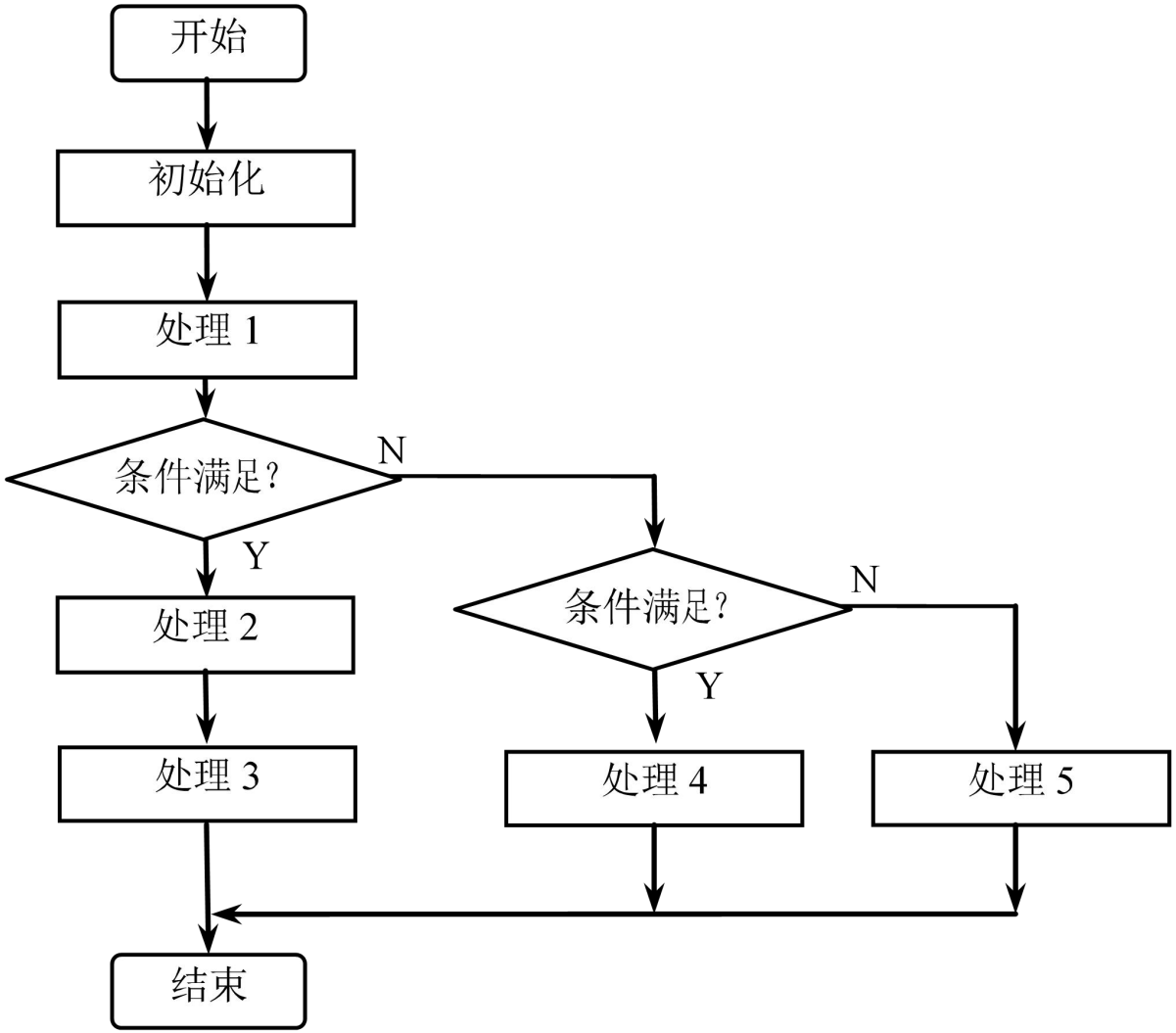


(b)　分支程序流程图



4.4.1　分支程序设计方法

- 分支程序允许嵌套，即一个分支接一个分支，形成树状多分支结构。多分支程序流程图如右图所示。



多分支程序流程图

4.4.2　分支程序设计实例

【例7】 设计一个水塔水位控制系统，晶振频率6MHz。设计要求如下：

(1) 在水塔内三个不同的高度分别安装了一根固定不动的金属棒，正常情况下，塔内水位应保持在虚线之内，水位控制原理如下图所示。

(2) A棒处于水位上限，B棒处于水位下限。当水位低于水位下限时，自动启动水泵电机给水塔供水；直到塔内水位达到水位上限，自动停止水泵电机动转。

(3) 塔内水位从水位上限下降到水位下限的过程中，水泵电机不会自动启动。

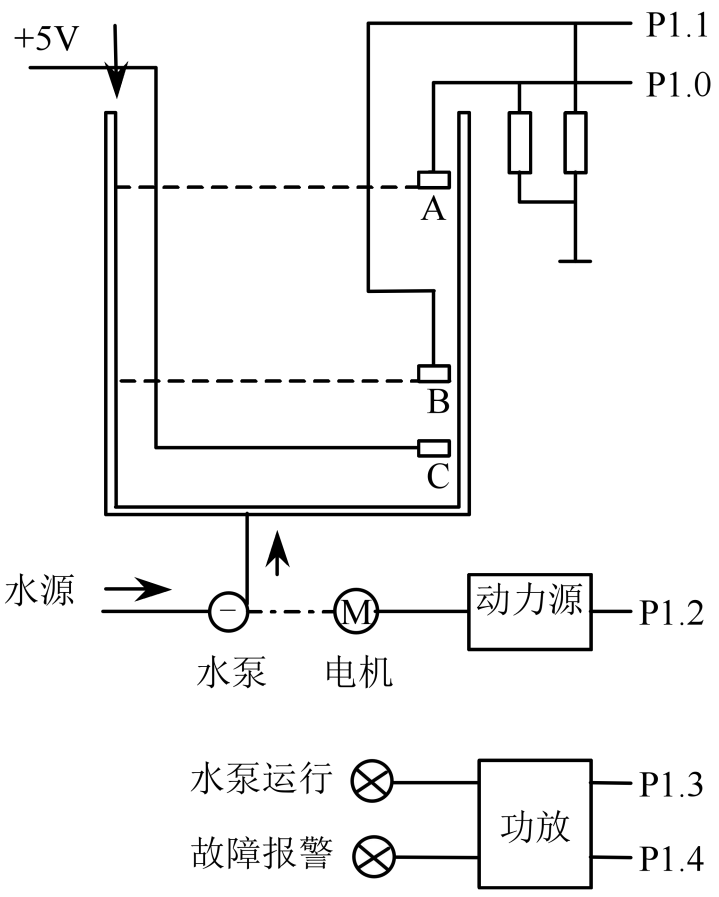
(4) 水塔进水时，要有信号灯指示；水位检测发生故障时，要有故障灯指示并使水塔水位控制系统停止工作。

4.4.2　分支程序设计实例

解：当塔内水位处于水位下限以下时，A、B棒通过电阻接地。

由于水的导电作用，当塔内水位达到水位下限时，B棒接通+5V；当塔内水位达到水位上限时，A棒也接通+5V。

水位上限信号输入至P1.0，水位下限信号输入至P1.1，P1.2输出控制信号以控制水泵电机的启动(P1.2=0)和停止(P1.2=1)，P1.3输出显示信号以指示水泵电机的运行状态(P1.3=0时点亮)，P1.4输出故障信号以指示水位检测系统故障状态(P1.4=0时点亮)。



4.4.2　分支程序设计实例

水位控制信号与水泵电机控制状态的对应关系

P1.1	P1.0	控制状态	P1.2	P1.3	P1.4
0	0	水泵电机启动	0	0	1
0	1	故障报警	1	1	0
1	0	维持原来状态			1
1	1	水泵电机停止	1	1	1

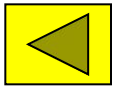
为了防止电机频繁启停，在启动或停止电机后最少要维持这一状态20s，这可以采用延时程序来实现。

```
                        ORG   0000H
0000H     02 00 30      LJMP  MAIN
                        ORG   0030H
0030H     43 90 03  MAIN: ORL   P1,#03H       ;水位信号输入端做读入准备
0033H     F5 90         MOV   A,P1            ;读入水位检测信号
0035H     30 E1 08      JNB   ACC.1,QDZB      ;P1.1=0,转移至启动准备
0038H     20 E0 16      JB    ACC.0,TZDJ      ;P1.0=1,转移至停止电机
003BH     12 01 00    YS: LCALL DELAY         ;延时20s
003EH     80 F0         SJMP  MAIN
0040H     30 E0 08  QDZB: JNB   ACC.0,QDDJ    ;P1.0=0,转移至启动电机
0043H     D2 92         SETB  P1.2            ;停止电机
0045H     D2 93         SETB  P1.3            ;关闭电机运行指示
0047H     C2 94         CLR   P1.4            ;打开水位检测故障指示
0049H     80 FE         SJMP  $
004BH     C2 92     QDDJ: CLR   P1.2          ;启动电机
004DH     C2 93         CLR   P1.3            ;打开电机运行指示
004FH     80 EA         SJMP  YS
0051H     D2 92     TZDJ: SETB  P1.2          ;停止电机
0053H     D2 93         SETB  P1.3
0055H     80 E4         SJMP  YS
```

地址	机器码	程序	注释
		ORG 0100H	
0100H	7B C8	DELAY: MOV R3,#200	;延时20s的循环次数
0102H	7F C8	LP1: MOV R7,#200	;延时100ms的循环次数
0104H	7E 7B	LP2: MOV R6,#123	;延时0.5ms的循环次数
0106H	00	NOP	
0107H	DE FE	LP3: DJNZ R6,LP3	
0109H	DF F9	DJNZ R7,LP2	
010BH	DB F5	DJNZ R3,LP1	
010DH	22	RET	
		END	;程序结束

4.5　子程序设计

- 4.5.1　子程序设计方法
- 4.5.2　子程序设计实例



4.5.1　子程序设计方法

- 子程序是指完成某一专门任务并能被其他程序反复调用的程序段。调用子程序的程序称为主程序或调用程序。使用子程序的过程称为调用子程序。子程序执行完后返回主程序的过程称为子程序返回。

- 主程序和子程序是相对的，同一程序既可以作为另一程序的子程序，也可以有自己的子程序。也就是说，子程序是允许嵌套的，嵌套深度和堆栈区的大小有关。

- 采用子程序能使整个程序结构简单，缩短程序设计时间，减少对存储空间的占用。

4.5.1　子程序设计方法

- 子程序的特点和设计方法
- 子程序具有通用性和独立性，以满足所有调用程序实现资源共享。
- 子程序的第一条指令的地址称为子程序的入口地址，该指令前应有标号。
- 合理地确定子程序的参数传递方式：入口参数是子程序需要的原始参数，由主程序通过相关的工作寄存器、特殊功能寄存器、片内RAM或堆栈等传送给子程序；出口参数是根据入口参数执行子程序后获得的结果，由子程序通过相关的工作寄存器、特殊功能寄存器、片内RAM或堆栈等传递给主程序。
- 在主程序中可以用调用指令调用子程序，在子程序末尾用RET返回指令从子程序返回主程序。

4.5.1　子程序设计方法

- 根据需要保护现场和恢复现场。在子程序的开始，使用压栈指令把需要保护的内容压入堆栈；在返回主程序前，使用弹出指令把堆栈中保护的内容送回原来的存储单元中。

  子程序中有可能要使用累加器**A**或工作寄存器，在子程序使用它们之前，把它们中可能存有的主程序的中间结果保存起来，这一过程称为保护现场。在子程序执行完并将返回主程序之前，再将这些中间结果取出，送回到累加器**A**或原来的工作寄存器中，这一过程称为恢复现场。
- 子程序中应尽量使用相对转移指令而不使用其他转移指令，以便子程序放在内存的任何区域都能被主程序调用。
- 要正确地设置堆栈指针，以避免堆栈区与工作寄存器或其他存储单元发生冲突。

4.5.1　子程序设计方法

- 传送子程序参数的方法。
- 利用寄存器或片内**RAM**传送参数。可以把入口参数存放到寄存器或片内**RAM**中传送给子程序，也可以把出口参数存放到寄存器或片内**RAM**中传送给主程序。
- 利用寄存器传送参数的地址。把存放入口参数的地址通过寄存器传送给子程序，子程序根据寄存器中存放入口参数的地址便可找到入口参数并对它们进行相应操作；出口参数的地址也可通过寄存器传送给主程序。
- 利用堆栈传送参数。可以用压栈指令**PUSH**把入口参数压入堆栈传送给子程序，也可以使用压栈指令**PUSH**把出口参数压入堆栈传送给主程序。

4.5.2　子程序设计实例

【例10】将片内RAM区20H～24H单元中的一位十六进制数转换成ASCII码，并分别存放到片内RAM区30H～34H单元中。

解：ASCII码是有一定规律的编码，如十六进制数的0～9的ASCII码为该数值加上30H，分别为30H～39H；十六进制数的A～F的ASCII码为该数值加上37H，分别为41H～46H。

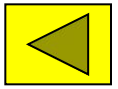
4.5.2 子程序设计实例

地址	机器码		程序		注释
			ORG 0000H		
0000H	02 01 00		LJMP MAIN		
			ORG 0100H		
0100H	7C 05	MAIN:	MOV R4,#05H		;数据块的长度
0102H	78 20		MOV R0,#20H		;存放十六进制数首地址
0104H	79 30		MOV R1,#30H		;存放ASCII码首地址
0106H	E6	LP1:	MOV A,@R0		;取十六进制数
0107H	12 01 50		LCALL HAC		;调用代码转换程序
010AH	F7		MOV @R1,A		;存放ASCII码
010BH	08		INC R0		
010CH	09		INC R1		
010DH	DC F7		DJNZ R4,LP1		
010FH	80 FE		SJMP $		

地址	机器码		程序	注释
			;程序名：HAC	
			;功能：十六进制数转换成ASCII码	
			;入口参数:A存放要转换的十六进制数	
			;出口参数:A存放转换后的ASCII码	
			;占用资源:PSW	
			ORG 0150H	;子程序从地址0150H开始存放
0150H	C0 D0	HAC:	PUSH PSW	;保护现场
0152H	54 0F		ANL A,#0FH	;屏蔽掉高4位
0154H	C0 E0		PUSH ACC	;
0156H	C3		CLR C	;
0157H	94 0A		SUBB A,#0AH	;比较A中内容的大小
0159H	D0 E0		POP ACC	
015BH	40 02		JC LP2	;(A)＜10时,转移
015DH	24 07		ADD A,#07H	
015FH	24 30	LP2:	ADD A,#30H	
0161H	D0 D0		POP PSW	;恢复现场
0163H	22		RET	;子程序返回
			END	

4.6　查表程序设计

- 4.6.1　查表程序设计方法
- 4.6.2　查表程序设计实例



4.6.1　查表程序设计方法

- 在单片机的实际应用中，经常要对一些数据进行函数运算，例如求平方、正弦函数等，为了提高单片机执行程序的速度，一般将某函数的全部函数值按一定的规律编成表格存放到程序存储器中。
- 查表程序就是根据某数据的函数运算要求，按索引号从程序存储器中查找与之相对应的函数值的程序结构。
- 设计查表程序时，主要通过两条查表指令实现查表功能。

4.6.1　查表程序设计方法

- 查表程序的特点和设计方法。
- 查表指令“**MOVC A，@A+DPTR**”的查表过程比较简单。查表时首先需要把数据表格起始地址存入**DPTR**，然后把所查数据的索引值送入累加器**A**中，最后使用查表指令“**MOVC A，@A+DPTR**”完成查表。
- 查表指令“**MOVC A，@A+PC**”的查表过程相对复杂一些。查表时首先使用传送指令把所查数据的索引值送入累加器**A**，然后用“**ADD A，#data**”指令对累加器**A**进行修正。**data**值由该式确定：**PC+data=**数据表格的首地址。其中，**PC**是“**MOVC A，@A+PC**”的下一条指令的地址。因此，**data**值实际等于查表指令和数据表格之间的字节数。最后使用查表指令“**MOVC A，@A+PC**”完成查表。

4.6.2　查表程序设计实例

【例14】 变量a、b均为小于10的正整数，编程计算$c=a^2+b^2$，其中变量a、b分别存放在片内RAM的51H和52H单元中，计算结果c存放到片内RAM的53H单元。

解法一：首先把平方表格起始地址0150H存入DPTR，然后把要查数据a或b的索引值送入累加器A中，最后使用查表指令“MOVC A，@A+DPTR”完成查表。

4.6.2 查表程序设计实例

```
                              ORG   0000H
0000H     02 01 00            LJMP   MAIN
                              ORG   0100H
0100H     90 01 50    MAIN:   MOV   DPTR,#0150H    ;平方表格首地址
0103H     E5 51               MOV   A,51H          ;取数a送到A中作为索引值
0105H     93                  MOVC   A,@A+DPTR     ;查数a的平方
0106H     F8                  MOV   R0,A           ;数a的平方送到R0中暂存
0107H     E5 52               MOV   A,52H          ;取数b送到A中作索引值
0109H     93                  MOVC   A,@A+DPTR     ;查数b的平方
010AH     28                  ADD   A,R0           ;求平方和
010BH     F5 53               MOV   53H,A
010DH     80 FE               SJMP   $
                              ORG   0150H
0150H     00 01 04 09 TABLE:  DB   0,1,4,9         ;0～9平方表
0154H     10 19 24            DB   16,25,36
0157H     31 40 51            DB   49,64,81
                              END
```

4.6.2　查表程序设计实例

解法二：如果采用查表指令“**MOVC A，@A+PC**”，首先把要查数据**a**或**b**的索引值送入累加器**A**中，然后用“**ADD A，#data**”指令对累加器**A**进行修正，最后使用查表指令“**MOVC A，@A+PC**”完成查表。

数**a**的索引值的修正值=平方表格首址-下一条指令的**PC**值

=0150H-0105H=4BH

数**b**的索引值的修正值=平方表格首址-下一条指令的**PC**值

=0150H-010BH=45H

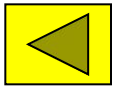
```
                                ORG   0000H
0000H    02 01 00               LJMP   MAIN
                                ORG   0100H
0100H    E5 51          MAIN:   MOV   A,51H        ;取数a送到A中作为索引值
0102H    24 4B                  ADD   A,#4BH       ;对索引值进行修正
0104H    83                     MOVC   A,@A+PC     ;查数a的平方
0105H    F8                    MOV   R0,A          ;数a的平方送到R0中暂存
0106H    E5 52                  MOV   A,52H        ;取数b送到A中作为索引值
0108H    24 45                  ADD   A,#45H       ;对索引值进行修正
010AH    83                     MOVC   A,@A+PC     ;查数b的平方
010BH    28                     ADD   A,R0         ;求平方和
010CH    F5 53                  MOV   53H,A
010EH    80 FE                  SJMP   $
                                ORG   0150H
0150H    00 01 04 09    TABLE:   DB   0,1,4,9      ;0～9平方表
0154H    10 19 24               DB   16,25,36
0157H    31 40 51               DB   49,64,81
                                END
```

本章小结

- 程序设计语言按语言结构可分为三大类，即机器语言、汇编语言和高级语言。在目前单片机的开发应用中，经常采用**C**语言和汇编语言共同编写程序。
- 汇编语言是面向机器的程序设计语言，对于**CPU**不同的单片机，其汇编语言一般是不同的。在进行汇编语言源程序设计时，必须严格遵循汇编语言的格式和语法规则。
- 汇编语言源程序是由汇编语言语句构成的。汇编语言语句可分为两大类：指令性语句和指示性语句。指令性语句一般由标号、操作码、操作数和注释四个字段组成，指示性语句也包括标号、操作码、操作数和注释四个字段。
- 汇编语言源程序的汇编可分为手工汇编和机器汇编两类。机器汇编可以使用汇编程序进行汇编或反汇编。

本章小结

- 汇编语言程序具有顺序结构、循环结构、分支结构和子程序结构四种结构形式。实际的应用程序一般都由一个主程序和多个子程序构成，即采用模块化的程序设计方法。
- 程序设计的原则是尽可能使程序简短和缩短运行时间，设计的关键首先是根据实际问题和所选用的单片机的特点来合理地确定解决问题的算法，然后是将工作任务细分成易于理解和实现的小模块。
- 在程序设计时，要注意顺序程序、循环程序、分支程序、查表程序和子程序的特点和设计方法。要设计出高质量的程序还需要掌握一定的技巧，通过多读、多看一些实用程序可以积累一定的设计经验。

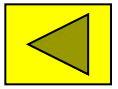


习　题

1. 程序设计语言有哪几类？各有什么特点？
2. 汇编语言有哪两类语句？各有什么特点？
3. 汇编语言源程序有哪两类汇编？各采用什么方法来实现？
4. 汇编语言程序设计一般分哪几个步骤？
5. 有两个4位十六进制数分别存放在片内RAM 20H、21H单元和30H、31H单元内，请编程求它们的和，并将和存放到片内RAM 40H、41H单元。
6. 有一个16位二进制负数的原码存放在片内RAM 60H、61H单元内，请编程求它的补码，并将它存放到片内RAM 70H、71H单元。
7. 片内RAM中存放有20个数据，首地址为40H，请编程将数据块传送到片外RAM以50H为首地址的存储单元中。
8. 片外RAM中存放有20个数据，首地址为40H，请编程将数据块传送到片外RAM以5000H为首地址的存储单元中，同时将片外RAM以40H为首地址的20个存储单元中的内容全部清零。
9. 片内RAM区30H～34H单元中存放有5个十六进制数，请编程计算这5个数的算术平均值，结果存放到片内RAM区35H单元中。
10. 请分别编写延时1 min、1 h的子程序，晶振频率12MHz。

习　题

11. 自变量X为一无符号数，存放在片内RAM区30H单元，函数Y存放在31H单元。请编写满足如下关系的程序：X≤95时，Y=1；95＜X＜105时，Y=2；X≥105时，Y=3。
12. 在片内RAM区从30H单元开始存放了50个数据，请编程找出某一关键值并将该值在片内RAM区存储单元的地址存放到片内RAM区70H单元中。
13. 编写双字节无符号数加法子程序。
14. 编写双字节无符号数减法子程序。



Q & A?

Thanks!