微机原理和接口技术

第七讲 指令系统与汇编程序7

提 纲

13. 编程语言及汇编语言编程风格
14. 汇编程序设计中的伪指令
15. 汇编与调试过程
16. 汇编语言程序设计概述
17. 程序设计的结构化
18. 基本程序设计
19. 子程序设计概述
20. 子程序设计举例

提 纲

14. 汇编程序设计中的伪指令

汇编语言中的伪指令

- 伪指令又称汇编程序控制译码指令，在汇编时不产生机器指令代码，不影响程序的执行，仅指明在机器汇编源程序时，需要执行的一些特殊操作，如指定程序或数据存放的起始地址，给一些连续存放的数据确定单元以及指示汇编结束等。

1. 起始汇编 ORG（Origin）

格式： ORG nn

➢ **功能：**

- 给出程序存放的起始地址，nn是16位二进制数。
- ORG指令给程序起始地址或数据块起始地址赋值。
- 应放在每段源程序或数据块的开始。

汇编语言中的伪指令

1. 起始汇编 ORG

在一个源程序中可以多次使用，以规定不同程序段或数据块的起始位置，所规定的地址必须从小到大，并且不允许重叠。

```
例：      ORG   0000H
MAIN:  MOV   SP，#6FH
          . . .
          LCALL  SUB1
          . . .
          ORG   1000H
SUB1:   MOV  A，#74H
          . . .
           RET
```

主程序MAIN存放的起始地址是0000H；子程序SUB1的起始地址是1000H。

汇编语言中的伪指令

2. 赋值 EQU（Equal）

格式：字符名称 EQU 数据或表达式

➢ **功能：把数据或表达式赋值给字符名称**

例：

DATA1　EQU 22H

ADDR1　EQU 2000H

给标号DATA1和ADDR1赋值22H和2000H

➢ **注意**

- EQU语句给字符名称赋值后，在整个程序中该字符名称的值就固定不能更改了。
- EQU定义的字符必须先定义后使用。

3. 定义字节 DB（Define Byte）

格式：标号：**DB** 字节常数或字符串

- 功能：将常数或字符串存入从标号开始的连续存储单元中。

例：　　ORG 2000H

TABLE： DB 73H,04,100,32,00,-2,"ABC";

表示：TABLE 标号的起始地址为2000H；数据73H,04H,64H,20H,00H, FEH,41H,42H,43H依次存入 2000H开始的ROM中。

4. 定义字 DW（Define Word）

格式：标号： **DW** 字或字串

- 功能：把字或字串存入由标号开始的连续存储单元中，且把字的高字节数存入低地址单元，低字节数存入高地址单元。

例： ORG 1000H

LABLE： DW 100H，3456H，1357H，……

表示：从 LABLE 地址（1000H）开始，按顺序存入

01H,00H,34H,56H,

13H,57H,……

- 注意：

- DB和DW定义的数表，其个数不得超过80个，若数据的数目较多时,可以使用多个定义命令。
- 通常用DB来定义数据，用DW来定义地址。

5. 位地址赋值 BIT

格式：字符名称 **BIT** 位地址

位地址可以是绝对地址，也可以是符号地址。

➢ 功能：用于给字符名称赋予位地址。

例： LED1 BIT P3.1 ；LED1赋值为P3.1，在整个程序中，LED1即为P3.1。

SETB LED1 ；设置P3.1为高电平

6. 结束汇编 END

格式：**END**

➢ 功能：表示程序的结束

程序的结尾必须要有**END**语句，而且只能有一条。

7. 定义存储器空间 DS（Define Storage）

格式：标号： DS nn

- **功能：通知汇编程序，在目标代码中，从标号地址开始，保留出nn个存储单元。**

例： BASE: DS 100H

从标号BASE开始，保留出100H个存储单元，以备源程序另用。

- **注意：**

- 对于8051微控制器，DB、DW、DS等伪指令是应用于ROM，而不能用于RAM。

汇编语言中的伪指令

8. 伪指令应用举例

```
        ORG  2100H
BUF:    DS    10H
        DB    08H,42H;
        DW    100H,1ACH,122FH;
```

2100H	
2101H	
	… …
2110H	08
2111H	42
2112H	01
2113H	00
2114H	01
2115H	AC
2116H	12
2117H	2F

说明:

- 从2100H至210FH为预留的缓冲区空间
- (2110H)=08H　(2111H)=42H
- 2112H单元起依次存放01H、00H、01H、ACH、00H、12H、2FH

提 纲

15. 汇编与调试过程

汇编与调试过程

- 应用汇编语言开发应用程序，首先应用某个编辑软件（编辑器）完成源程序的编写，然后用汇编程序将源程序汇编成MCU可执行的机器码，通过调试确保程序逻辑正确，最终实现预定的功能。

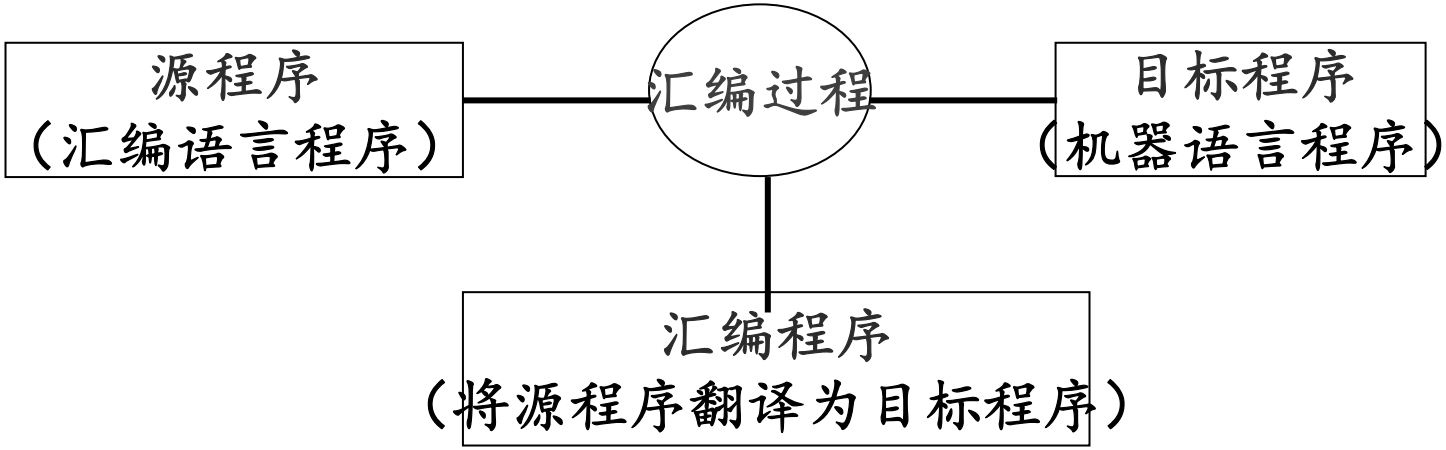
汇编与调试过程

1. 汇编程序的编辑

设计程序时，首先应用某个编辑软件完成源程序的编写，扩展名为.ASM。

2. 汇编程序的汇编

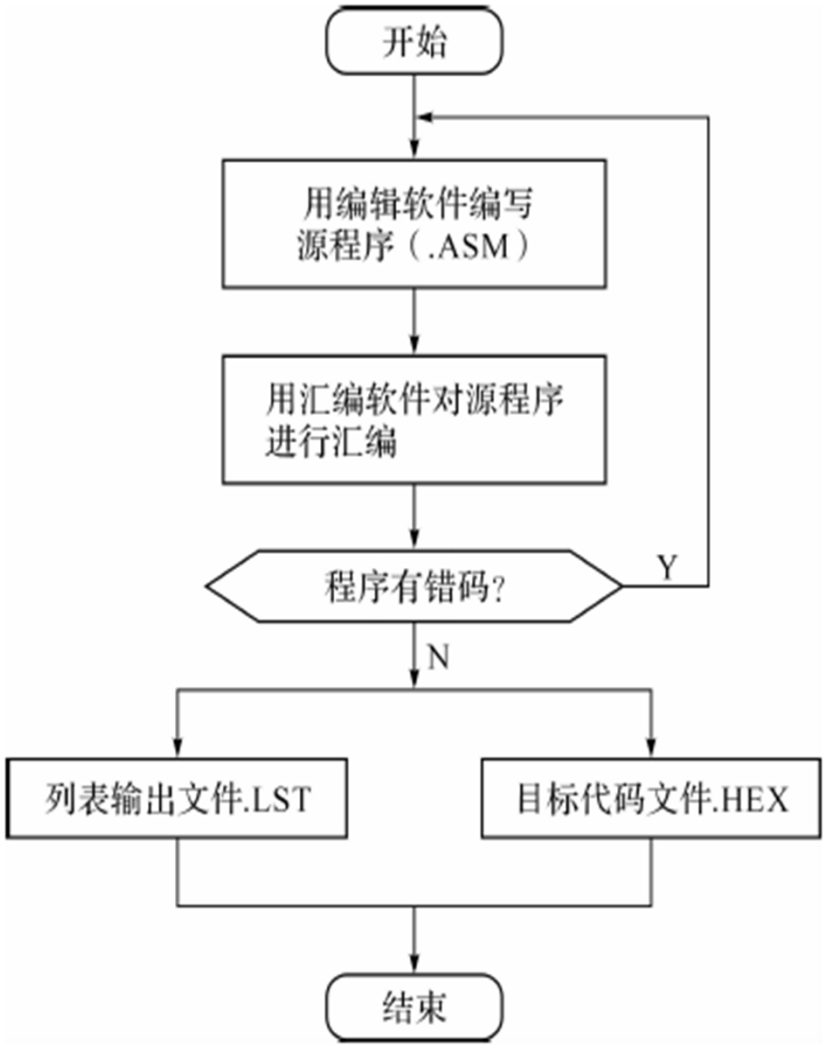
将汇编程序转换成机器程序（目标程序）的过程，称汇编。有人工汇编和机器汇编两种方式。



机器汇编过程

3. 汇编语言程序的设计过程

- 编写好源程序后进行汇编操作。如果源程序有语法错误，汇编程序会列出出错个数及错误语句所在行。
- 再返回编辑状态、修改源程序，再进行汇编操作，如此往复，直到无错误为止，即汇编通过。
- 汇编通过后会形成两个文件，即列表程序文件(.LST)和目标程序文件(.HEX) 。列表程序为汇编后的程序清单；目标程序文件是可执行文件。



汇编与调试过程

4. 汇编程序的调试过程

对于汇编通过的程序即可进行调试，可以在通用计算机环境下的模拟调试或在仿真器、目标系统上，进行在线实时仿真调试。

➢模拟仿真调试：将目标程序文件在模拟调试软件环境中，模拟程序运行状态的调试。难以对外围电路进行调试。

➢实时目标仿真调试：通过串行口将汇编好的目标程序文件传送到实时在线仿真器中，，实时仿真器通过仿真头与目标系统相连。仿真器为目标系统提供了一个可单步、可设断点运行、可修改、可观察运行状态。

➢脱机运行：经实时目标仿真调试通过的系统程序，通过程序写入器写入到目标系统的ROM中，进行脱机试运行。如果运行良好，则系统程序设计调试完毕，否则需要重新修改源程序，反复进行以上操作，直到成功。

4. 汇编程序的调试过程

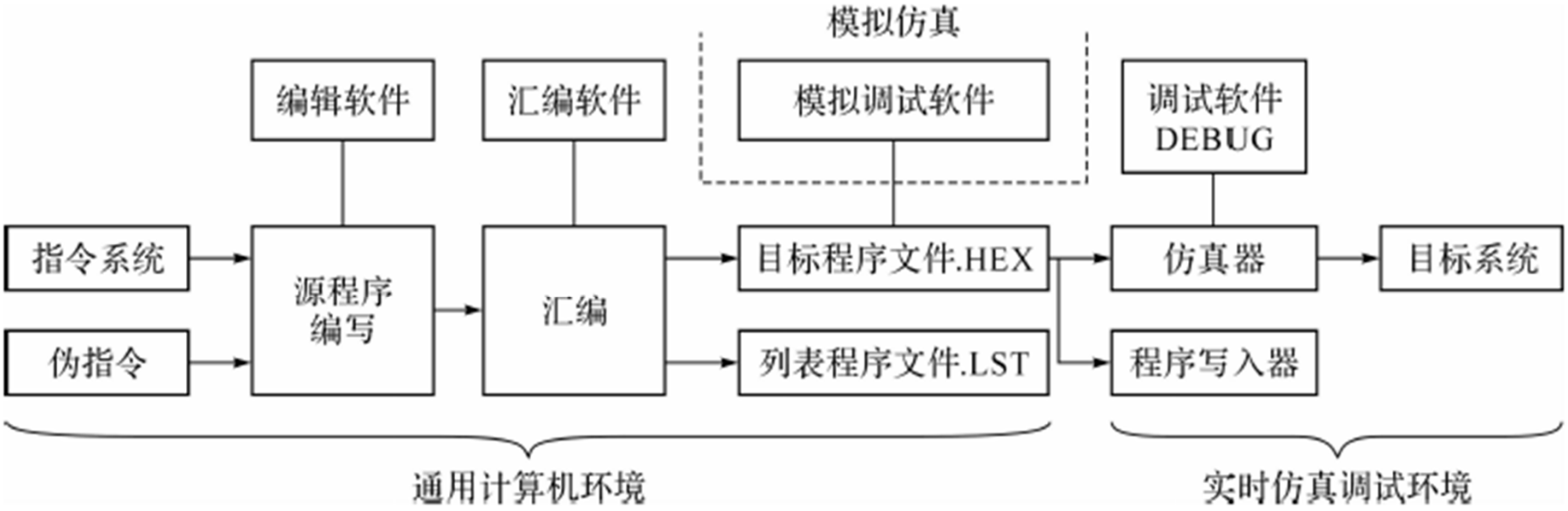


图 3-4　程序编辑、汇编运行调试的全过程

提 纲

16. 汇编语言程序设计概述

汇编语言程序设计概述

1.评价程序质量的标准

- 程序的执行时间，程序长度；
- 程序的逻辑性、可读性；
- 程序的兼容性、可扩展性；
- 程序的可靠性。

2.采用汇编语言的优点

- 占用的内存单元和CPU资源少；
- 程序简短，执行速度快；
- 可直接调动微控制器的硬件资源，有效地利用微控制器的专有特性；
- 能准确地掌握指令的执行时间，适用于实时控制系统。

3.汇编语言程序设计的步骤

- **模块划分：**根据设计系统的功能需求，进行功能模块的划分，把一个大而复杂的功能划分为若干个相对独立的功能模块。
- **模块功能分析：**尽可能将一个功能设计为一个子程序；仔细分析每个子程序的功能与具体实现方法，确定并画出子程序的流程图。
- **子程序功能和资源确定：**确定子程序名、调用条件、出入口参数等，以及程序中使用的工作寄存器、内存单元和其它硬件资源。
- **子程序编写调试：**按照各子程序流程图，分别编写源程序并进行汇编、调试和运行，直至实现各子程序的预期功能。
- **总程序构建：**有机整合各子程序构成系统总程序，并进行系统总体程序的分析调试，直至实现系统全部功能。

提 纲

17. 程序设计的结构化

程序设计的结构化

结构化程序设计方法是程序设计的常用方法。结构化程序设计是对用到的控制结构类程序作适当的限制，特别是限制转向语句的使用，从而控制程序的复杂性，使程序易读易理解，减少逻辑错误。

根据结构化程序设计的观点，任何复杂的程序都可由顺序结构、分支结构和循环结构三种基本结构组合而成。

程序设计的结构化

- **结构化程序设计的优点：**控制程序的复杂性，使程序易读易理解，减少逻辑错误。
- **结构化程序设计的特点：**程序结构简单清晰、易读/写、调试方便、生成周期短及可靠高。
- **程序设计的结构类型：**顺序结构、分支结构、循环结构

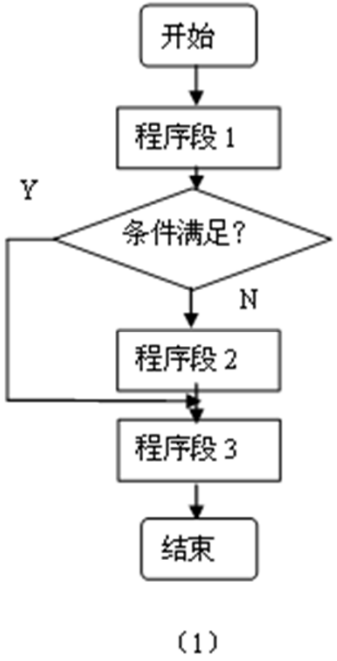
1.顺序结构

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。**具有一定功能的顺序程序是构成复杂程序的基础。**

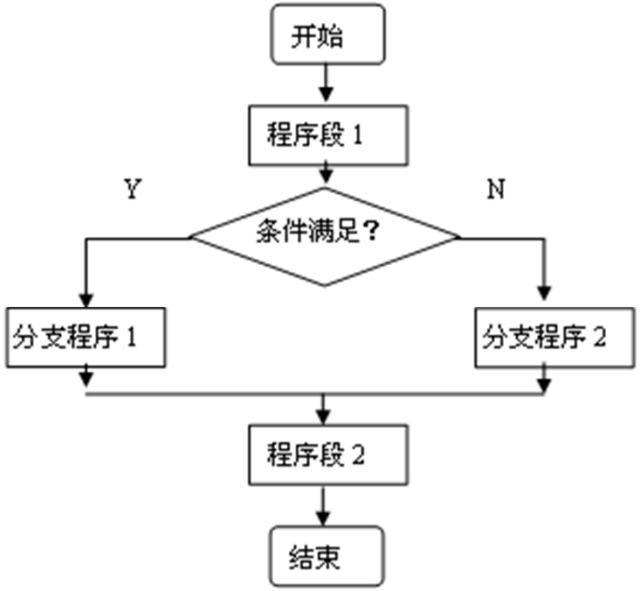
2.分支结构

包含条件判断指令，程序执行流程中需要做出逻辑判断，并根据判断结果选择合适的执行路径。

分支结构有**单分支结构、多分支结构**。

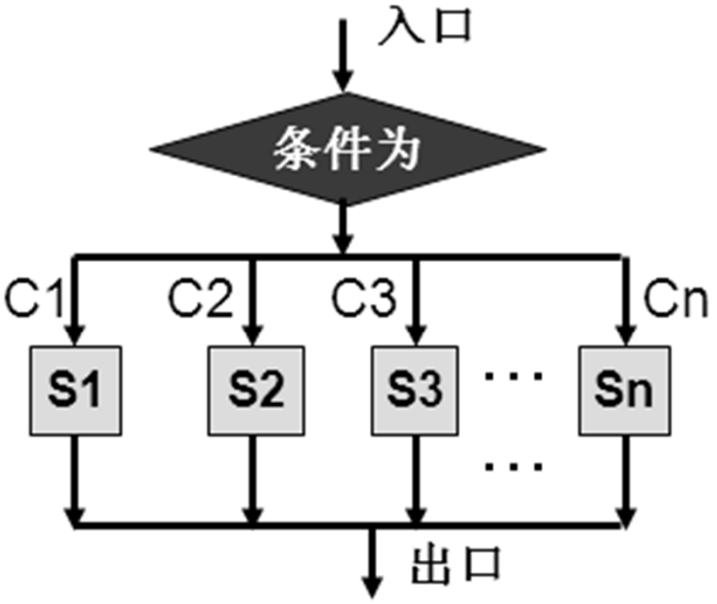


(1)



(2)

单分支结构



多分支结构

2.分支结构

(1)单分支结构

通常用条件转移指令来实现程序的分支。相关指令有:

- **位条件转移指令**,如：**JC、JNC、JB、JNB**和**JBC**等;
- **条件转移指令**,如：**JZ、JNZ、DJNZ**等。

(2)多分支结构

当程序通过判别，有两个以上的不同走向时，称为多分支结构。 对于8051微控制器，可实现多分支选择的相关指令有:

- **散转指令： JMP @A＋DPTR**

根据A的内容选择对应的分支程序,可达256个分支。

- **比较转移指令：如 CJNE 指令4条**

比较两个数的大小，必然存在大于、等于、小于三种情况，因此可实现三个程序分支。

3.循环结构

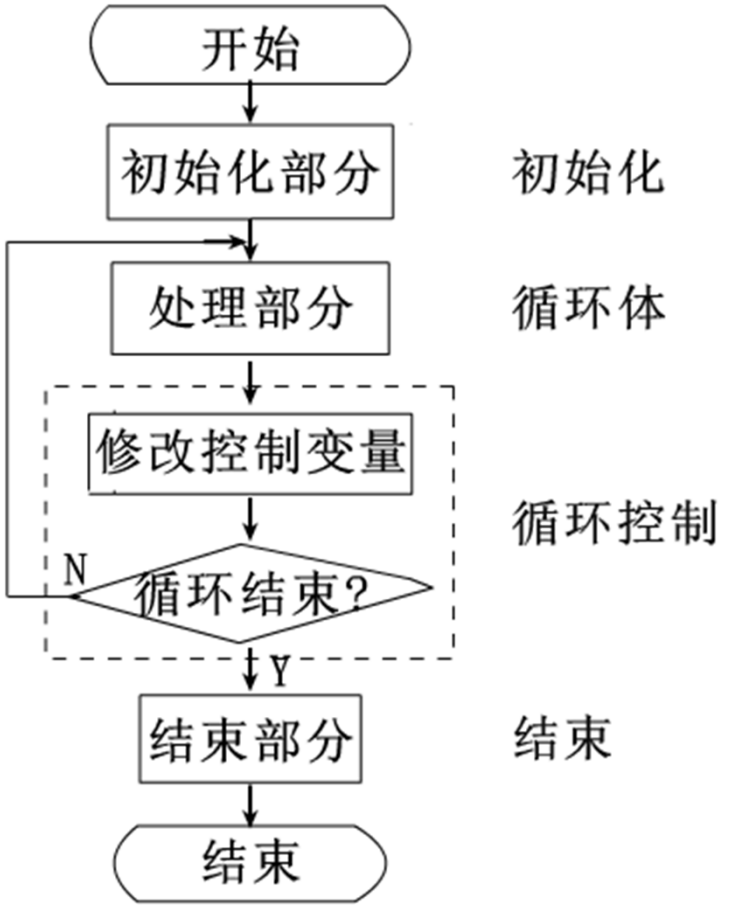
(1)循环结构由初始化、循环体、循环控制和结束四部分组成

- **初始化部分：**程序在进入循环之前，应对循环过程的工作单元，如循环次数、起始地址等变量设置初值，为循环做准备。有些情况下还要保护现场。
- **循环体：**是循环结构程序核心部分，完成实际的处理工作，需反复循环执行。
- **循环控制：**循环程序的控制部分，通过循环变量和结束条件进行控制。在循环体执行过程中，要不断修改循环变量和地址指针等有关参数，当符合结束条件时，结束程序的执行。
- **循环结束：**对循环程序执行的结果进行分析、处理和保存。如初始化部分进行了保护现场，则在这里需恢复现场。

3.循环结构

(2)循环控制方式

循环控制的实现方法主要有**计数控制法**和**条件控制法**。即通过修改循环变量或判断循环条件，实现对循环的判断和控制。



➢**计数循环结构**：循环次数已知(初始化中已设定其初值)，由循环次数决定循环体的执行次数。常用DJNZ的2条指令进行控制。计数循环结构一般采用先处理后判断的流程。

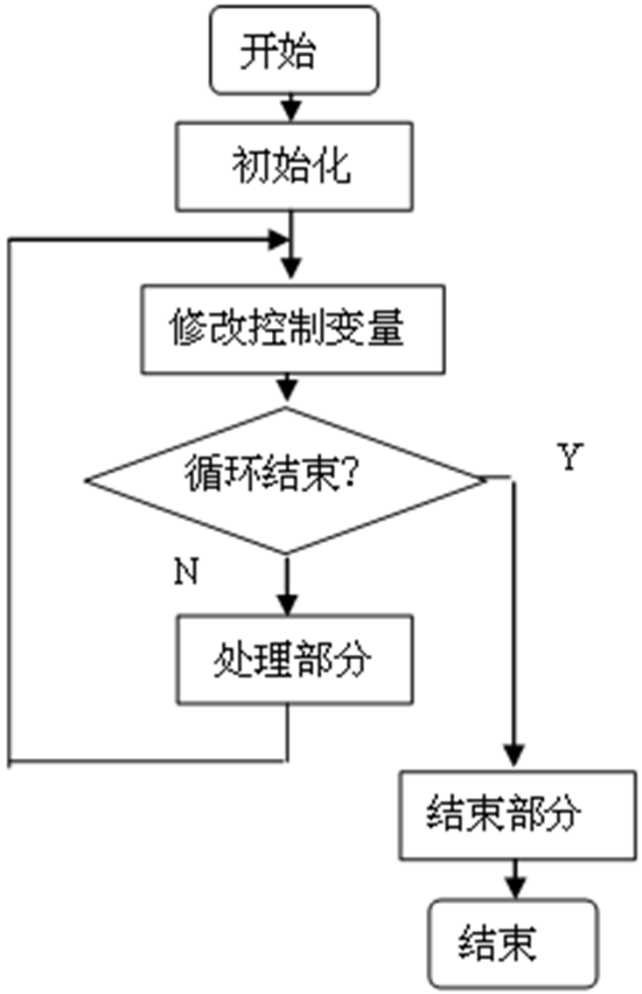
3.循环结构

(2)循环控制方式

➢ **条件循环结构：**根据循环结束的条件，决定是否继续循环程序的执行。

结束条件可以是搜索到某个参数（比如空格的ASCII码20H），也可以是发生某种状态（如电平变化）等，什么时侯结束循环是不可预知的。

常用比较转移指令或条件转移指令，来进行条件控制。一般采用先判断后处理的流程。



3.循环结构

(3)循环程序设计的注意点

- 在进入程序之前，应合理设置循环初始变量。
- 循环体只能执行有限次，如果无限执行的话，则会造成**死循环**，应避免这种情况的发生。
- 不能破坏或修改循环体，不能从循环体外直接跳转到循环体内。
- 在多重循环结构中，要求嵌套是从外层向内层一层层进入，从内层向外层一层层退出，不能在外层循环中用跳转指令直接转到内层循环体内。

Thank you!