第三章　汇编语言程序设计

3-1 汇编程序的约定

汇编程序：能将汇编语言的源程序转换成机器语言的目标程序的一种系统软件

汇编：汇编语言程序到机器语言程序的转换过程

1.手工汇编：人工查指令表，查出程序中每条指令对应的机器代码。用于设计短小程序或调试程序的场合

2. 机器汇编：用计算机中的汇编程序对用户源程序进行汇编。用机器汇编要考虑汇编程序的约定

汇编的主要任务：

1)确定程序中每条汇编语言指令的指令机器码

2)确定每条指令在存储器中的存放地址

3)提供错误信息

4)提供目标执行文件(*.OBJ/*.HEX)和列表文件(*.LST)

一. 汇编语言指令类型

1. 机器指令：指令系统中的全部指令。每条机器指令都有对应的机器代码，可以被CPU执行

2. 伪指令： 汇编控制指令，没有指令代码，只用于汇编过程(预编译指令)，为汇编程序提供汇编信息

3. 宏指令

宏汇编功能：将需要反复多次执行的程序段定义成一个宏指令名（宏定义）。编程时，可在程序中使用宏指令名来替代被定义的程序段（宏调用）

宏定义过程：

```
宏指令名    MACRO   形式参数
              …                    ；被定义的程序段
            ENDM
```

宏调用过程：

```
            …
        宏指令名    实际参数
            …
        宏指令名    实际参数
```

用机器汇编要考虑汇编程序的约定

1）按指令格式和语法规则编写程序

常数表示法：

十进制数：　20，20D

十六进制数：87H，0F0H

二进制数：　01011001B

字符：　　‘H’

字符串：　“Hello”

2）使用伪指令提供汇编信息

二．伪指令 常用伪指令及功能：

1. ORG—起始地址指令。定义程序或数据的起始地址

指令地址	机器码		源程序
			ORG 2000H
2000H	78 30	MAIN：	MOV R0，#30H
2002H	E6		MOV A，@R0
			…
			ORG 3000H
3000H	23	TAB：	DB 23H，100，‘A’
3001H	64		
3002H	41		

2. DB —定义字节常数指令。输入程序中的常数

例： DB 23H，100，‘A’

DB “HELLO”

3．DW —定义字型常数指令

例： DW 1234H，5678H

4. EQU —等值指令。为标号或标识符赋值

```
      X1    EQU  2000H
      X2    EQU  0FH
            …
MAIN: MOV  DPTR，#X1
      ADD  A，#X2
```

5. END —结束汇编指令

```
例：START:      …
                …
            END   START
```

3-2 汇编语言程序设计步骤

一. 确定方案和计算方法

二. 了解应用系统的硬件配置、性能指标

三. 建立系统数学模型，确定控制算法和操作步骤

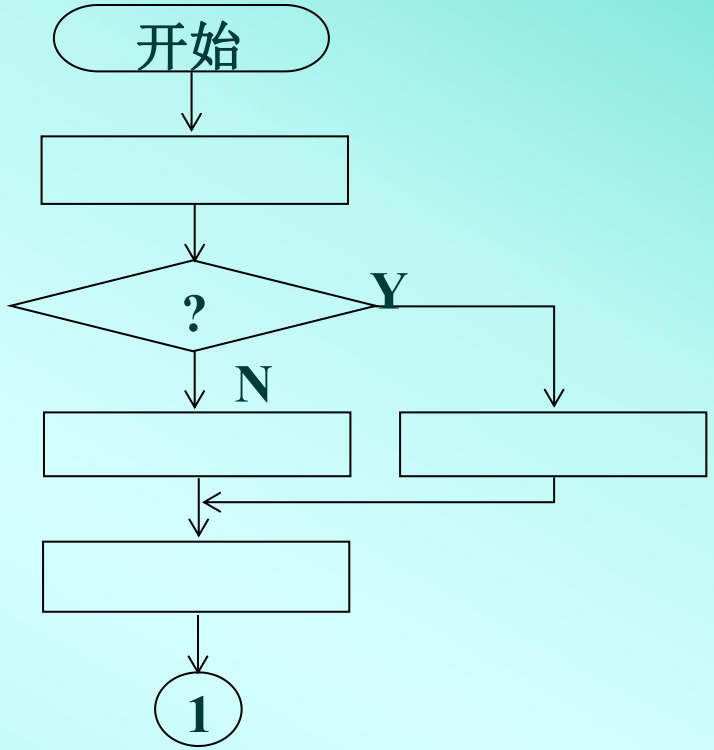
四. 合理分配存储器单元和了解I/O接口地址

五. 编制源程序

1.按功能设计程序，明确各程序之间的相互关系

2. 用流程图表示程序结构和功能

3.程序中用注释说明指令在程序中的作用，方便阅读、调试和修改



常用程序结构

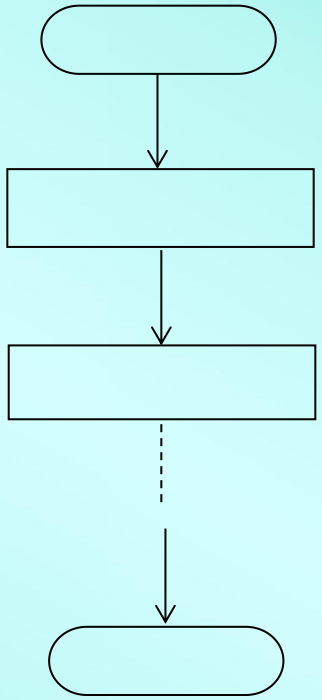
直线程序、分支程序、循环程序、子程序

3-3 直线程序

直线程序(简单程序)，程序走向只有一条路径

双字节变补程序(设数据在R4R5中)

```
MOV   A，R5 ；取低字节
CPL   A
ADD   A，#1 ；低字节变补
MOV   R5，A
MOV   A，R4 ；取高字节
CPL   A
ADDC  A，#0 ；高字节变补
MOV   R4，A
```



例3-3-4 压缩式BCD码分解成为单字节 BCD码(拆字)

```
MOV   R0，#40H；  设指针
MOV   A，@R0   ；  取一个字节
MOV   R2，A    ；  暂存
ANL   A，#0FH  ；  清0高半字节
INC   R0
MOV   @R0，A   ；  保存数据个位
MOV   A，R2
SWAP A         ；  十位换到低半字节
ANL   A，#0FH
INC   R0
MOV   @R0，A   ；  保存数据十位
```

片内RAM	
42H	0000十位
41H	0000个位
40H	十位 个位

3-4　分支程序

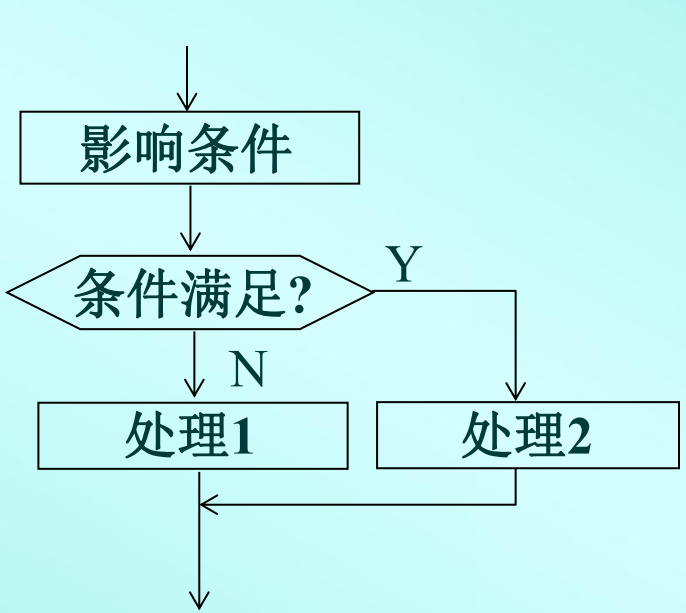
由条件转移指令构成程序判断框部分，形成分支结构

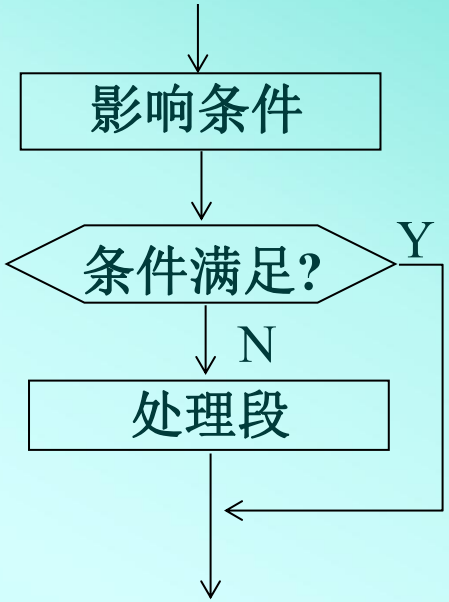
例　求8位补码的绝对值：
正数不变，负数变补

```
    MOV A，R2
    JNB  ACC.7，N；为正数？
    CPL  A        ；负数变补
    INC  A
    MOV  R2，A
N：  SJMP N        ；结束
```

3-4-2 单重分支程序

一个判断决策框，形成一个分支两条出路。两种分支结构：





每10kg为1个计价单位G已存入40H单元。（G值小，下列运算不溢出）

行李运费计算：当G≤5，M=G×3；当G＞5，M=G×3＋(G-5)×(5-3)

```
FRT:   MOV  A，40H          ；取行李重量计价单位G
       MOV  R3，A           ；暂存G
       MOV  B，#03H         ；运费M=G×3
       MUL  AB
       MOV  R2，A           ；暂存3G
       MOV  A，R3           ；取回G
       CJNE A，#05H，L1     ；G≤5？
       SJMP WETC
L1:    JC   WETC            ；G≤5转至WETC
       SUBB A，#05H         ；否则 M=3G+2(G-5)
       RLC  A
       ADD  A，R2
       MOV  R2，A
WETC:  MOV  41H，R2         ；存运费 M
       RET
```

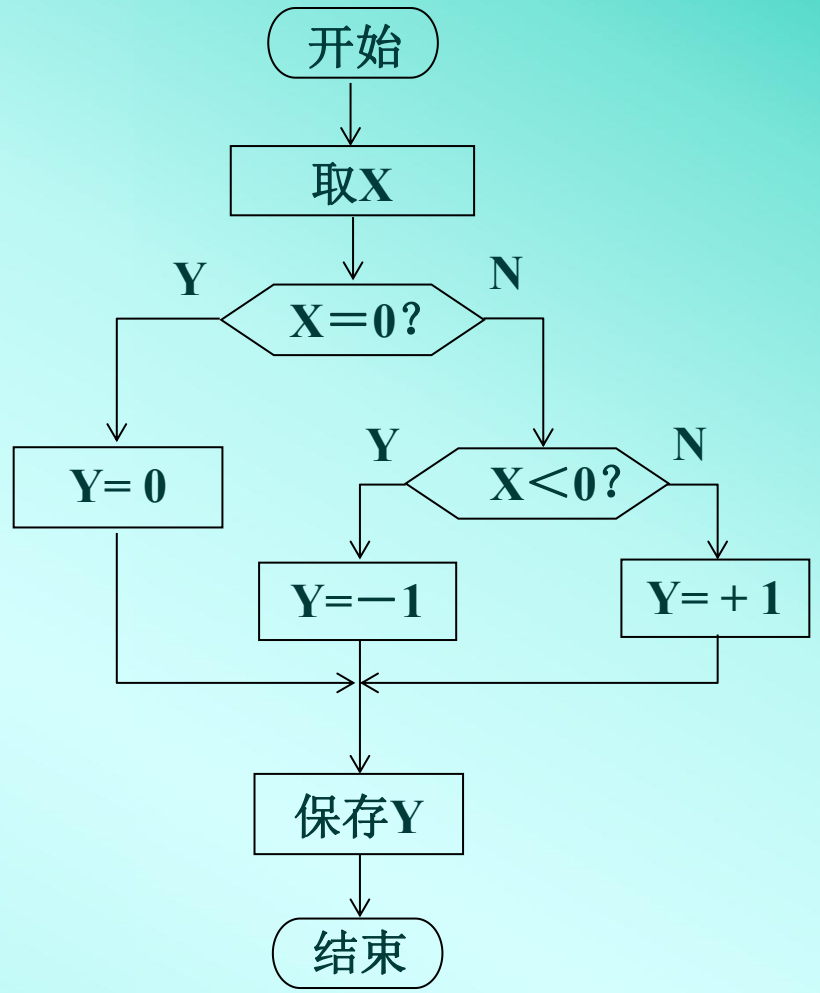
3-4-3　多重分支程序

一．多次使用条件转移指令，形成两个以上判断框

例　求符号函数 Y= SGN(X)

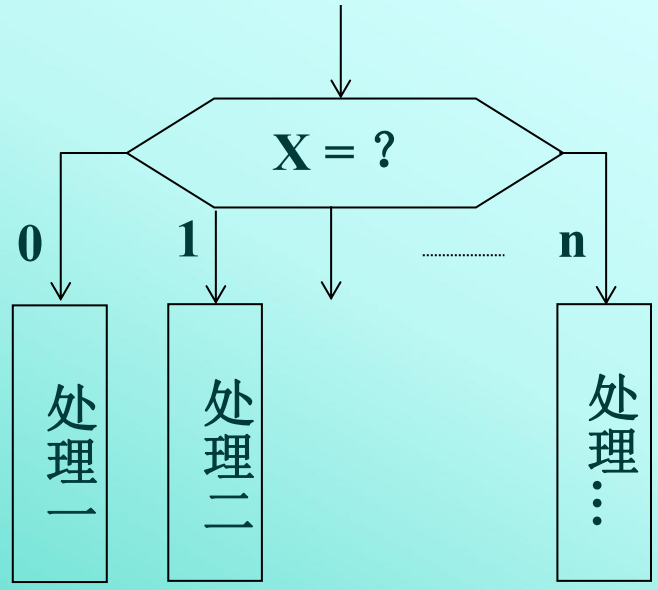
$$SGN(X)=\begin{cases}+1 & \text{当 } X>0\\ 0 & \text{当 } X=0\\ -1 & \text{当 } X<0\end{cases}$$

```
SYMB:   MOV  A，40H          ；取X
        JZ   STOR            ；X=0，Y=X
        JB   ACC.7，MINUS    ；X＜0
        MOV  A，#1           ；X＞0，Y=+1
        SJMP STOR
MINUS:  MOV  A，#0FFH        ；X＜0，Y= －1
STOR:   MOV  41H，A          ；保存Y
        RET
```



二．按分支号X转移： 当 X=0，程序转移到 ADDR0 处； 当 X=1，程序转移到 ADDR1处； …

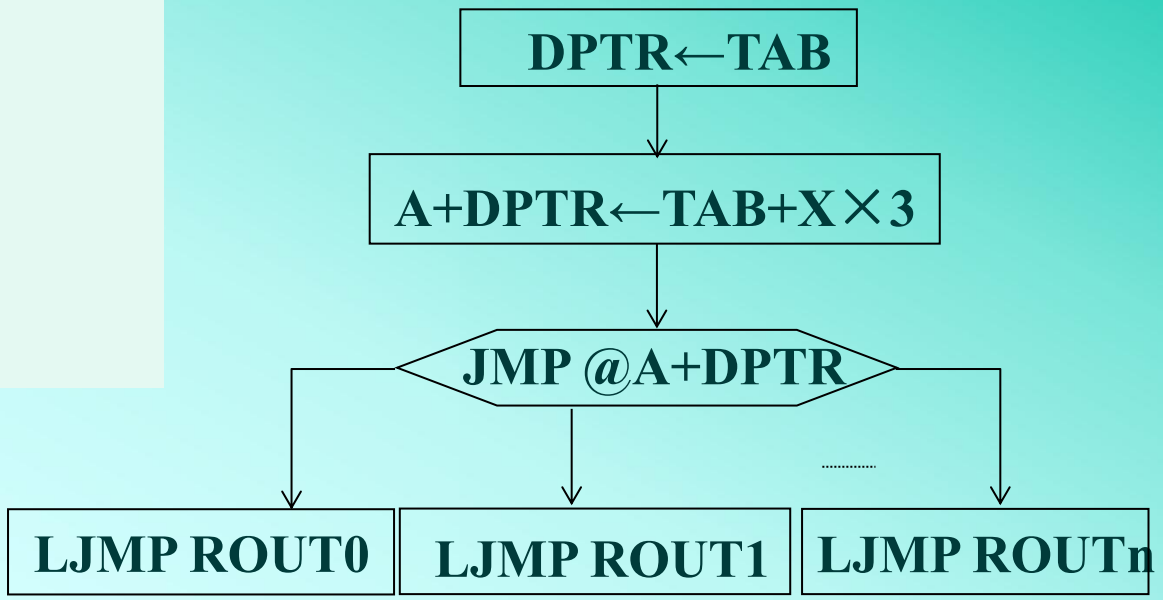
(1)用地址表法： 设分支号 X 已存入 A累加器




```
MTJS：  MOV   DPTR，#TAB     ；取表首地址
        ADD    A，A          ；X×2
        MOV   R2，A
        MOVC A，@A+DPTR     ；取分支地址
        PUSH   ACC           ；保存分支地址低8位
        MOV    A，R2
        INC    A
        MOVC  A，@A+DPTR    ；取分支地址
        PUSH   ACC           ；保存分支地址高8位
        RET                  ；分支地址→PC，转移
TAB：   DW     ADDR0         ；分支地址表
        DW     ADDR1
               …
ADDR0：        …             ；程序段 0
               …
ADDR1：        …             ；程序段 1
```

(2)转移表法 分支转移指令JMP @A+DPTR　　设R7R6 = X (分支号)

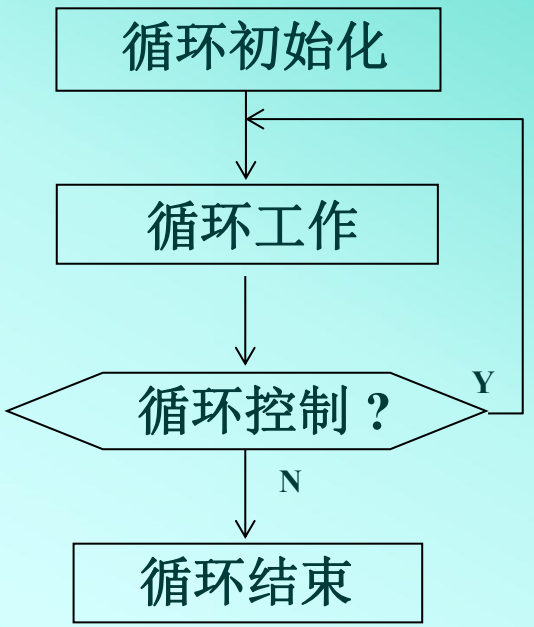
```
MTJS:  MOV DPTR，#TAB ；指向表首
       MOV A，R7   ；X 高字节×3
       MOV B，#03H
       MUL AB      ；DPH←乘积+DPH
       ADD A，DPH
       MOV DPH，A
       MOV A，R6   ；X 低字节×3
       MOV B，#03H
       MUL AB
       XCH A，B
       ADD A，DPH  ；DPH←DPH+((R7、R6)×3)高字节
       MOV DPH，A
       XCH A，B    ；A←((R7、R6)×3)低字节
       JMP @A+DPTR ；实现多分支转移
TAB:   LJMP ADDR0     ；转移表
       LJMP ADDR1
            …
       LJMP ADDRN
ADDR0:      …          ；程序段0 …
```



3-5 循环程序

包含多次重复执行的程序段，循环结构使程序紧凑

3-5-1 循环程序的构成



各个环节任务：

一．初始化部分：循环准备工作

如：清结果单元、设数据指针、设循环控制变量的初值等

二．循环体

循环工作部分：需多次重复处理的工作

循环控制部分：

1. 修改数据指针，修改循环控制变量

2. 检测循环条件：满足循环条件，继续循环，否则退出循环

三. 结束部分

处理和保存循环结果

循环工作至少执行一次的循环结构

允许 0 次循环的结构：在循环工作之前检测循环条件

3-5-2　单重循环

简单循环结构：循环体中不套循环

例：求n个单字节数据的累加，设数据串已在43H起始单元，数据串长度在42H单元，且累加和不超过2个字节

```
SUM:  MOV R0，#42H   ；设数据指针 R0
      MOV A，@R0
      MOV R2，A      ；设循环计数器 R2←n
      CLR A          ；结果单元清0
      MOV R3，A      ；结果高位清0
ADD1: INC R0         ；修改数据指针
      ADD A，@R0     ；累加
      JNC NEXT       ；处理进位
      INC R3         ；有进位，高字节加1
NEXT: DJNZ R2，ADD1  ；循环控制：数据全部加完？
      MOV 40H，A     ；循环结束：保存结果
      MOV 41H，R3
      RET
```

片内 RAM	
…	…
	Xn
…	…
43H	X1
42H	n
41H	SUM_H
40H	SUM_L

循环控制方法：计数控制、特征控制

一. 计数控制

设循环计数器，控制循环次数，有正计数和倒计数两种方式

例：为一串7位ASCII码加奇校验(D_7=0)，设ASCII码数据串已存放在片外RAM的2101H起始单元，数据长度在2100H单元

```
        MOV  DPTR，#2100H；设指针
        MOVX A，@DPTR
        MOV  R2，A        ；设计数器
NEXT：  INC  DPTR         ；修改指针
        MOVX A，@DPTR；   取ASCII码
        ORL  A，#80H      ；加奇校验位
        JNB  P，PASS      ；不符合奇校验？转移
        MOVX @DPTR，A；   修改数据
PASS：  DJNZ R2，NEXT ；  已处理全部数据？
DONE：  SJMP DONE
```

片外RAM	
…	…
2102H	01101000
2101H	00101101
2100H	n

二. 特征控制：设定循环结束标志实现循环控制

例：找正数表中的最小值，设正数表已存放在片外RAM中以LIST为起始单元，用-1作为结束标志

```
START：  MOV DPTR，#LIST      ；数据表首地址
         MOV  B，#127         ；预置最小值（最大正数）
NEXT：   MOVX A，@DPTR        ；取数
         INC      DPTR        ；修改指针
         CJNE A，#-1，NEXT1   ；是否为结束标志？
         SJMP DONE            ；循环结束
NEXT1：  CJNE A，B，NEXT2     ；找较小值
NEXT2：  JNC  NEXT
         MOV  B，A            ；保存较小值
         SJMP NEXT            ；循环
DONE：   SJMP DONE
```

3-5-3　多重循环

循环体中套循环结构。以双重循环使用较多

例 将内存中单字节无符号数串升序排序

双重循环，每次取相邻两个单元的数据比较，决定是否需要交换数据位置

冒泡排序法步骤：

第一次内循环，比较N-1次，取数表中最大值

第二次内循环，比较N-2次，取到次大值

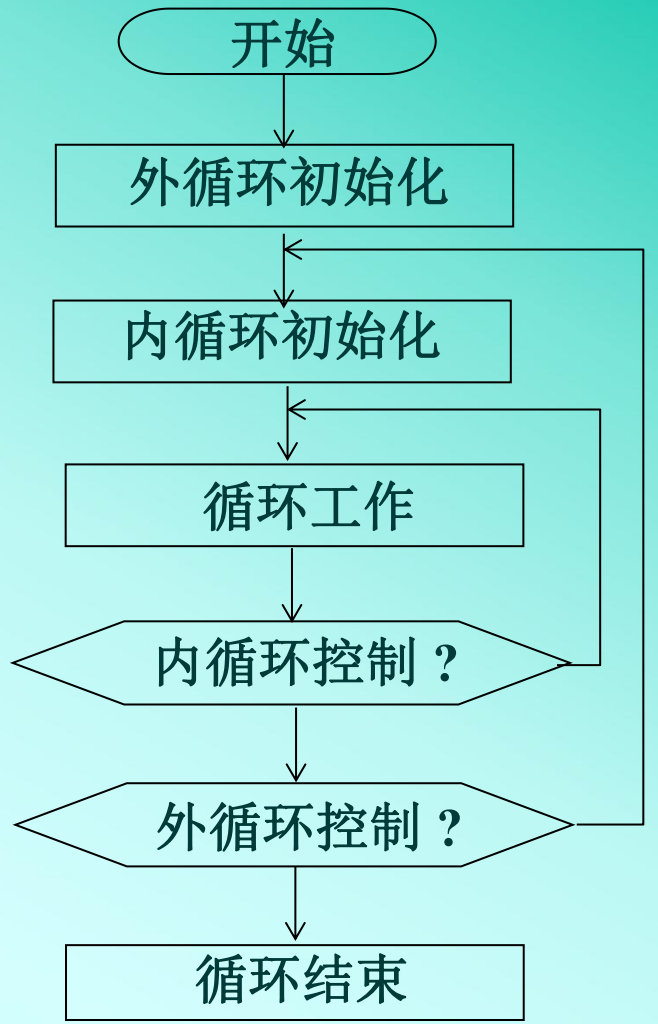
…

第N-1次内循环：比较一次

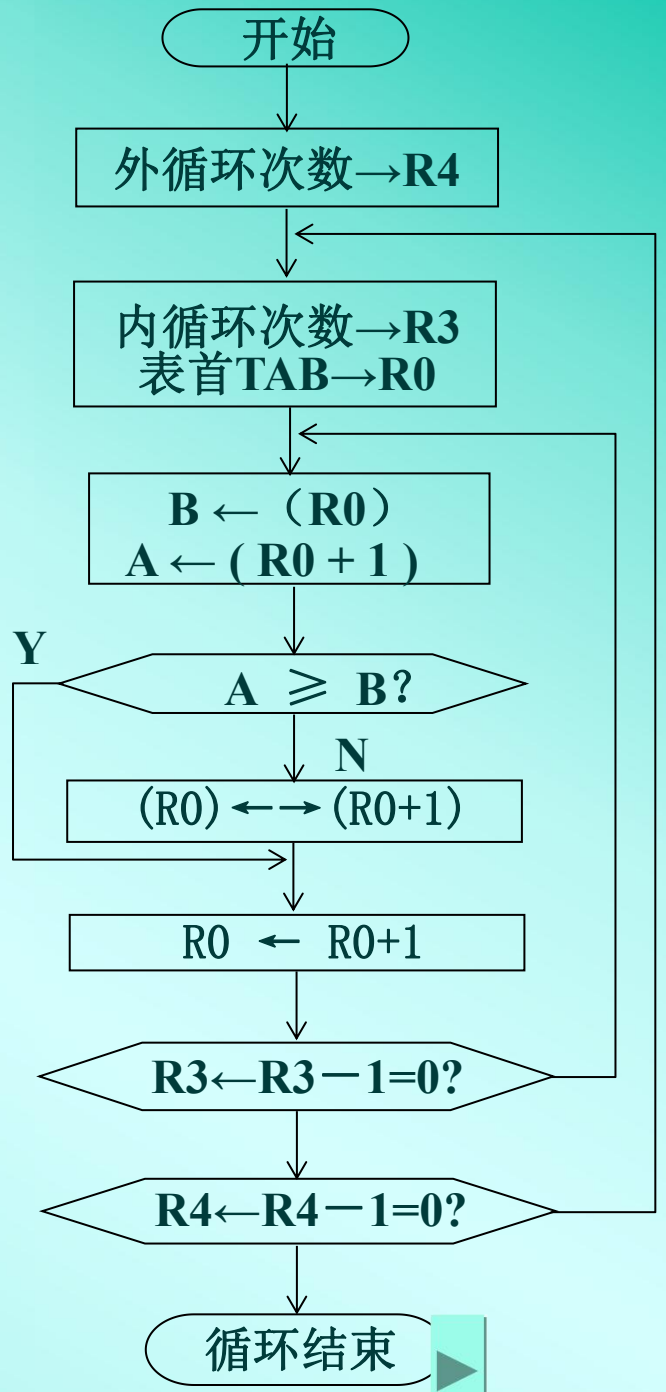
排序结束

由外循环计数器决定内循环次数；内循环计数器决定比较次数

片内 RAM	
MAX	
	…
	5
	2
	4
	1
TAB→	3



```
SORT:  MOV  A, #N-1    ; N个数据排序
       MOV  R4, A      ; 外循环次数
LOOP1: MOV  A, R4
       MOV  R3, A      ; 内循环次数
       MOV R0, #TAB    ; 设数据指针
LOOP2: MOV B, @R0      ; 取相邻单元数据
       INC  R0
       MOV  A, @R0
       CJNE A, B, L1   ; 比较
    L1: JNC  UNEX      ; A≥B，不交换
       DEC  R0         ; 相邻单元交换
       XCH  A , @R0
       INC  R0
       MOV  @R0, A
UNEX:  DJNZ R3, LOOP2; 内循环结束?
       DJNZ R4, LOOP1; 外循环结束?
       RET
```



软件延时程序

用循环程序将指令序列重复多次执行，实现软件延时

例：试计算下面软件延时程序的执行时间

源程序	指令周期(M)	指令执行次数
DELAY：MOV R6，#64H	1	1
I1：MOV R7，#0FFH	1	100
I2：DJNZ R7，I2	2	100×255
DJNZ R6，I1	2	100
RET	2	1

延时时间计算：（设时钟频率fosc=12MHz，则机器周期 M=1μs）

t =(1×1 + 1×100 + 2×100×255 + 2×100+2×1)×M

= 1+100+51000+200+2 = 51303 μs = 51.3 ms

设计软件延时程序时，延时时间主要决定于内循环中指令序列的循环次数

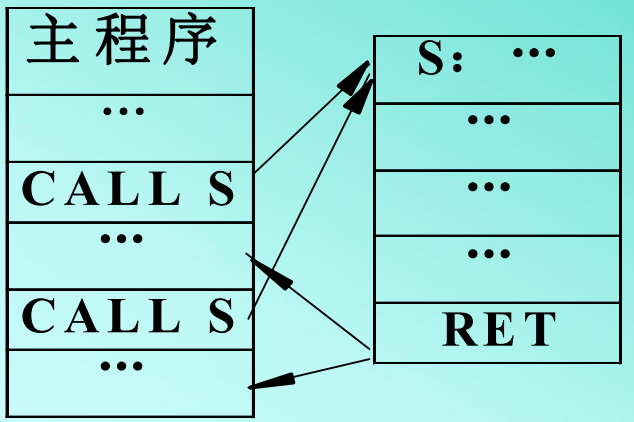
3-6　子程序

能完成某项特定功能的独立程序段，可被反复调用执行

子程序调用指令：　LCALL　子程序名

或：　ACALL　子程序名

子程序返回指令：　RET



3-6-1　子程序设计

一．子程序入口用标号作为子程序名

二．调用子程序之前设置好堆栈

三．用返回指令RET结束子程序，并保证堆栈栈顶为调用程序的返回地址

四．子程序嵌套要考虑堆栈容量

五．提供足够的调用信息：

子程序名、子程序功能、入口参数和出口参数、子程序占用的硬件资源、子程序中调用的其他子程序名…

3-6-4　子程序的类型

按子程序与调用程序之间传递参数的方式分类

入口参数：调用子程序之前，调用程序传给子程序的参数

出口参数：子程序送回调用程序的结果参数

设计子程序应满足通用性的要求，不能针对常数编程

例：1. 子程序功能为求单字节数的立方：

B、A←A^3，入口参数：A，出口参数：A、B

2. 子程序功能为求单字节数的n次方

(41H)(42H)←$(40H)^A$

入口参数：(40H)和 A，出口参数：(42H)(41H)

几种参数传递方式：

1. 寄存器传送参数
2. 存储器传送参数
3. 堆栈传送参数

例 比较两个数据串是否完全相等，若完全相等，A=0；否则 A=FFH

程序

入口参数：

A、R0、R1

A---长度

R0、R1 -首地址

出口参数：A

```
PMT：  MOV  R2，A        ；取数串长度
CHC：  MOV  A，@R0       ；分别取两个数串中的数据
       MOV  42H，@R1
       CJNE  A，42H，N；相等？不相等转移
       INC   R0              ；相等，修改指针
       INC   R1
       DJNZ R2，CHC    ；全部比较完？
       MOV  A，#0          ；设完全相等标志：A=0
       SJMP  P
  N：  MOV A，#0FFH    ；设不完全相等标志：A=FFH
   P：  RET
```

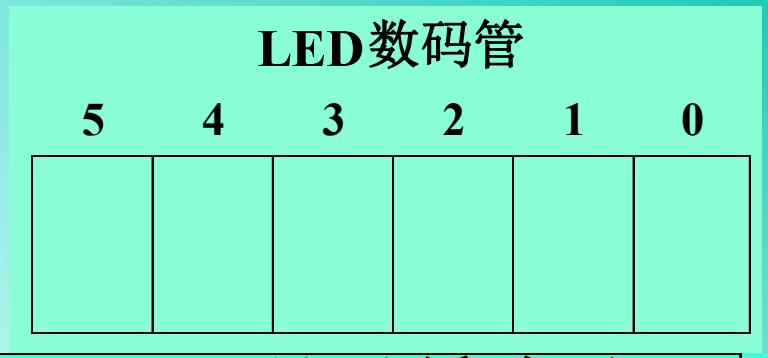
例：将R4R5R6中三个字节数据对半分解，变成6个字节，存入片内RAM的显示缓冲区(DISMEM0～DISMEM5)

```
UFOR1：MOV  @R0，#0
       XCHD  A，@R0；保存低半字节
       INC   R0          ；修改指针
       MOV  @R0，#0
       SWAP A
       XCHD A，@R0；保存高半字节
       RET
```

1）UFOR1子程序的功能：

将A累加器中单字节数据，对半分解成两个字节，存入R0指向的相邻两个单元

2）入口参数：将待分解的内容送 A，片内RAM的存放地址送R0



RAM显示缓冲区	
地址	数据
DISMEM5	0 $R6_H$
DISMEM4	0 $R6_L$
DISMEM3	0 $R5_H$
DISMEM2	0 $R5_L$
DISMEM1	0 $R4_H$
DISMEM0	0 $R4_L$

	RAM	
R0 + 1 →	0	A_H
R0 →	0	A_L

例：查表求出数据的ASCII码，以字符形式输出

1）子程序HEXASC功能：取出堆栈中数据，查表将低半字节转换成ASCII码送累加器 A

2）分别将待转换数据入栈，然后调用子程序 HEXASC

```
        MOV  SP，#30H
        PUSH 40H              ；入口参数入栈
        LCALL HEXASC
        POP  A
          …
HEXASC：DEC  SP               ；跳过返回地址
        DEC  SP
        POP  A                ；取入口参数
          …                   ；查表求ASCII码
        PUSH A                ；保存出口参数
        INC  SP               ；指向返回地址
        INC  SP
        RET                   ；返回主程序
  TAB：DB   ‘0’，‘1’，…；ASCII码表
```

RAM	
…	
41H	23
40H	01

…	
33H	PC_H
32H	PC_L
31H	01
30H	×

3-7　算术运算程序

3-7-1　多字节加减运算程序

1. 多字节加法子程序，Z=X+Y

```
ADDS:  CLR  C
LOOP:  MOV  A，@R0
       ADDC A，@R1      ；加一字节
       SETB  RS0        ；选寄存器1区
       MOV  @R0，A      ；存和一字节
       INC   R0         ；修改指针
       CLR  RS0         ；选寄存器0区
       INC   R0         ；修改指针
       INC   R1
       DJNZ  R2，LOOP   ；全部字节加完?
       RET
```

	Y_H
	…
R1→	Y_L
	X_H
	…
R0→	X_L

3-7-2 多字节无符号数乘除运算

移位相加计算多字节乘法程序，步骤：

1. 部分积清零
2. 从右至左逐位检测乘数：

   为1则部分积对位加被乘数，否则加0
3. 对位方法：被乘数左移或部分积右移

```
      110
  ×   101
  ------
      110
     000
+   110
  ------
    11110
```

例：双字节数相乘： **R2R3×R6R7®R4R5R6R7**

解：部分积清零，右移并检测乘数的移出位，决定部分积是否加被乘数，然后部分积右移对位。循环16次

初值0：　0　0 … 0

循环16次

Cy　R4R5　右移R6R7 并检测Cy

+　R2R3　Cy=1，加被乘数；Cy=0则不加

Cy　R4R5　得部分积并右移对位

乘积最后右移一次

相减求商，计算多字节除法程序，步骤：

1. 对齐左端，用被除数高位试减除数
2. 若够减商上1，不够减商上0且恢复余数
3. 余数左移或除数右移对位
4. 循环重复前3步，直至取够相应位数的商

```
      10111
0101/ 01110110
      0101
       0100
       0101
        1001
        0101
         1001
         0101
          1000
          0101
           011
```

例 R2R3R4R5÷R6R7®R4R5（余数®R2R3）

解 1. 判断R2R3＜R6R7？保证商不大于16位

2. 被除数左移1位，试减除数

3. 若够减，执行商加1并保留余数的操作

循环16次

Cy←R2R3←R4R5←0　除数和商同时左移1位

－ R6R7 ＋ 1 ←够减商加1，否则商为0

R2R3

3-7-3 代码转换程序

(一)十六进制数转换为ASCII码

0～9的ASCII码：30～39H，A～F的ASCII码：41～46H

```
HASC：  CJNE A，#0AH，N
   N：    JNC   N1
          ADD   A，#30H
          SJMP  SE
  N1：    ADD   A，#37H
  SE：    RET
-----------------
    ACALL   HASC
    MOV @R0,A
    INC  R0
    INC  R1
    DJNZ R3,LOOP
    SJMP  $        END
```

(二)调用此程序，将30H开始的18个数转换

```
ORG  1000H
MAIN：  MOV  R1，30H
        MOV  R0，60H
        MOV R3,#12H
LOOP: MOV  A,@R1
        ANL  A,#0FH
        ACALL   HASC
        MOV @R0,A
        INC  R0
        MOV  A,@R1
        SWAP  A
        ANL  A,#0FH
```