第4章

汇编语言程序设计

4.1 汇编语言程序设计基础

单片机的汇编语言程序设计步骤如下：

(1)明确要解决的问题和要求。

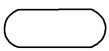
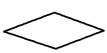
(2)根据要解决的问题，制定程序流程图。如程序较长，可以先画出粗框图，再根据要求进行细化。

(3)根据程序流程图，编写程序。如果程序较长，可按功能模块进行编写。

(4)对汇编语言程序进行调试，并进行优化处理。

程序流程图是用几何图形(方框和圆框)、直线及文字说明描述程序。它不但形象地描述程序执行的过程，而且清楚地表达程序结构的内在联系。利用流程图能方便地发现和分析程序结构存在的错误，便于掌握和进行交流，是程序设计的重要工具。程序流程图中常用的符号见表 4-1。

表 4-1　　程序流程图符号说明表

序号	图形	名称	含义
1		端点框	表示程序的开始或结束，在框内可填入相应的文字，如“开始”、“结束”或者程序名、起始地址等
2		流程线	表示程序执行的方向
3		处理框	表示一种处理功能或者过程，框内用文字简要说明
4		判断框	用于指示一个判定点，从这点产生分支。在框内应注明测试条件，而测试结果则注明在各分支流程线上
5		连接框	表示流程中止而并非流程结束。通常用来连接同一页上的流程，以避免流程线的交叉，使流程图阅读起来清晰；它也可用来连接不同页上的流程，注意在连接处连接框内的标识符要相同

单片机的汇编语言程序设计要特别注意如下几点：

(1)正确安排寄存器和数据存储单元。

(2)熟悉单片机的硬件结构，特别是特殊功能寄存器、并口、串口、定时器/计数器、中断等的应用。

(3)如果编写的程序较长，可以使用流程图帮助解决问题。

4.2 伪指令

汇编程序需要编译为机器码才能被单片机识别。在汇编程序中加入伪指令，可以使编译过程更容易实现。伪指令是用来指导汇编如何进行的，既不控制机器的操作也不能被汇编成机器代码。MCS-51 单片机常用的伪指令如下。

1. 起始地址伪指令 ORG

伪指令 ORG 用于规定目标程序段或数据块的起始地址。其格式为：

[标号:] ORG 16 位地址

其中，方括号内的标号是任选项，可有可无。

通常，在汇编语言程序的开始处均用 ORG 伪指令指定程序存放的起始地址。

2. 汇编结束伪指令 END

伪指令 END 表示源程序到此结束，在一个程序中，只允许出现一条 END 语句，而且必须安排在源程序的末尾。否则，汇编程序对 END 语句后面的所有语句都不进行汇编，其格式为：

[标号:] END

其中，方括号内的标号是任选项，可有可无。

例如：
```
ORG   0030H      ;从 ROM 的 0030H 单元开始存储程序
START:MOV   A,#34H
      MOV   B,A
      END                ;源程序到此结束
```

3. 赋值伪指令 EQU

伪指令 EQU 表示将伪指令右边的数值或地址赋给左边用户定义的符号。其格式为：

字符名称 EQU 数据或地址

由伪指令 EQU 赋值的字符名称在源程序中可以作为立即数，也可以作为数据地址、代码地址或位地址。由伪指令 EQU 所定义的符号必须先定义后使用，所以，伪指令 EQU 语句通常放在程序开头。

例如：
```
BR   EQU   08H     ;BR 的值等于 08H
BK   EQU   30H     ;将 30H 赋值给字符名称 BK
MOV  A,#BR         ;伪指令定义的符号作为立即数使用
MOV  B,BK          ;伪指令定义的符号作为地址使用
```

4. 字节定义伪指令 DB

伪指令 DB 表示从指定的地址单元开始，定义若干个字节存储单元的内容。其格式为：

[标号:] DB 8 位(二进制)数据表

例如：
```
ORG   0800H
FIRST:  DB  63H,01H,91H,07H
TABLE:  DB  96,40H,'C','7',1101B
```

经汇编后，ROM 的相关单元内容为：

(0800H) = 63H，(0801H) = 01H，(0802H) = 91H，(0803H) = 07H，(0804H) = 60H，(0805H) = 40H，(0806H) = 43H，(0807H) = 37H，(0808H) = 0DH。

其中，单引号定义的是字符的 ASCII 码，例如：0806H 单元中的 43H 是字符'C'的 ASCII 码；0807H 单元中的 37H 是字符'7'的 ASCII 码。0804H 单元中的 60H 是十进制数 96 对应的十六进制数；0808H 单元中的 0D 是二进制数 1101B 对应的十六进制数。

5. 字定义伪指令 DW

伪指令 DW 表示从指定的地址单元开始定义若干个 16 位数据。其格式为：

[标号:]　DW　16 位(二进制)数据表

由于一个字长为 16 位，故要占据两个存储单元，其中高 8 位存入低地址单元，低 8 位存入高地址单元。

例如：
```
ORG   0800H
TABLE: DW 7654H,40H,12,'AB'
```

经汇编后，ROM 对应单元的内容为：

(0800H) = 76H，(0801H) = 54H，(0802H) = 00H，(0803H) = 40H，(0804H) = 00H，
(0805H) = 0CH，(0806H) = 41H，(0807H) = 42H。

6. 数据地址赋值伪指令 DATA

伪指令 DATA 表示将表达式的数据地址或代码地址赋与规定的字符。其格式为：

字符名称　DATA　表达式

伪指令 DATA 与伪指令 EQU 的功能相似，但伪指令 DATA 所定义的符号可先使用后定义，在程序中它常用来定义数据地址，该语句一般放在程序的开头或末尾。

7. 定义空间伪指令 DS

伪指令 DS 表示从指定的地址单元开始，保留由表达式指定的若干字节空间作为备用空间。其格式为：

[标号:] DS　表达式

例如：
```
ORG   0800H
      DS   0AH
      DB   71H,13H,11H
```

经汇编后从 0800H 单元开始，保留 10 个存储单元，从 080AH 单元开始连续存放立即数 71H，13H，11H。

注意：伪指令 DB、DW、DS 只能用于 ROM，而不能用于 RAM。

8. 位地址赋值伪指令 BIT

伪指令 BIT 表示将位地址赋与规定的字符，常用于位处理的程序中。其格式为：

字符名称　BIT　　位地址

例如：
```
FLAG   BIT   F0
       X   BIT   P1.2
```

经汇编后是将 F0 及 P1.2(位地址)赋给变量 FLAG 和 X。

4.3　汇编语言程序设计实例

汇编语言程序设计中，普遍采用结构化程序设计方法。结构化程序分顺序结构、分支

结构、循环结构三种。

1. 顺序结构程序设计

顺序结构程序是程序结构中最简单的一种，从第一条指令开始顺序执行程序，直到最后一条指令为止。

顺序结构程序虽然简单，但只有掌握一定的技巧才能设计出高质量的程序。需要熟悉指令系统，正确选择指令，减少程序长度，最大限度地优化程序。

【例 4-1】 用逻辑运算指令编程实现将片内 RAM 单元存放的压缩 BCD 码拆成低四位和高四位，分别存入其他单元。

方法一：利用逻辑运算指令，把压缩 BCD 码拆成单字节 BCD 码。程序流程如图 4-1 所示。

程序清单：

```
SOU  EQU  60H          ;定义原数据存放地址
DT   EQU  61H          ;存放目的数据地址
ORG  0030H
MOV  A,SOU             ;取数据
ANL  A,#0FH            ;屏蔽高四位
MOV  DT,A              ;存低位
MOV  A,SOU             ;重新取数
SWAP  A                ;高低四位互换
ANL  A,#0FH
MOV  DT+1,A            ;存高位
RET
```

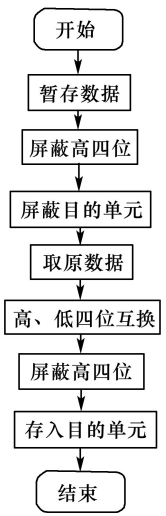


图 4-1 拆分单元数据流程图

进行高 4 位数转换时，也可使用指令“ANL A,#0F0H”先屏蔽低位，再利用高低四位互换指令，把高位数交换到低位。

方法二：把 BCD 数除以 16，商存在累加器 A 中，余数存在寄存器 B 中，则相当于把该数右移了 4 位，刚好把两个 BCD 码分别移到 A、B 的低 4 位。

程序清单：

```
SOU  EQU  60H
DT   EQU  61H
ORG  0030H
MOV  A,SOU             ;取数据
MOV  B,#16             ;除以 16
DIV  AB
MOV  DT,B              ;存低位
MOV  DT+1,A            ;存高位
RET
```

两种方法都可以达到目的，但通过两种方法比较可以看出，第二种方法占用容量小，但执行速度慢，方法一比较常用，当程序较长时需考虑这些细节。

2. 分支结构程序设计

如果在程序中需要根据不同条件采取不同处理方法，就应采用分支结构。分支程序

可以通过转移指令实现。编程中常会用到无条件转移、条件转移、散转移指令。

(1)无条件转移。

采用无条件转移时,程序执行方向是设计者事先安排的,与已执行程序的结果无关,使用时注意给出正确的转移目标地址即可。

(2)条件转移。

采用条件转移时,是根据已执行程序对标志位或累加器等结果的影响,决定程序走向,形成分支。编写时要注意选择正确的转移条件和转移目标地址。

需要注意的是在形成分支时,若不进行其他操作,两个分支不可转移到同一个目标地址。

【例 4-2】 设有一无符号数变量 X,编写计算下列函数式的程序,并将结果存入 Y 中。

$$Y=\begin{cases}(X+1)^2-1 & X<10\\(X+1)^2+8 & 10\leqslant X\leqslant 15\\(X15) & X>15\end{cases}$$

分析:本例为典型的分支程序。判断变量分支可以有多种方法,可以采用把无符号数分成小于 10,10 与 15 之间,大于 15 三段,首先判断该变量是否小于 10;否则判断它是否大于 15;否则该变量在 10 与 15 之间的方法。然后,编写 3 个分支程序分别进行计算。

注意:变量与 10 和 15 相等情况的判断。变量≥15 时,进位 C 都为 1,所以,分支 X>15 相当于判断 X≥16 情况。程序流程如图 4-2 所示。

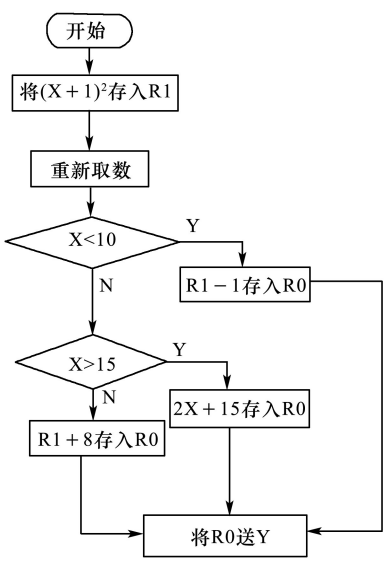


图 4-2　分支程序流程图

程序清单:

```
X   EQU   60H
Y   EQU   61H
ORG   0080H
MOV   A,X
```

```
        INC  A
        MOV  B,A
        MUL  AB              ;(X+1)²→A
        MOV  R1,A            ;暂存入 R1
        MOV  A,X             ;重新取数
        CJNE A,#10,shi1
shi1:   JC  shi3             ;X<10 转
        CJNE A,#16,shi2
shi2:   JNC  shi4            ;X>15 转
        MOV  A,R1
        ADD  A,#08H          ;(X+1)²+8→A
        MOV  R0,A
        SJMP  shi5
shi3:   MOV  A,R1
        CLR  C
        SUBB  A,#01          ;(X+1)²-1→A
        MOV  R0,A
        SJMP  shi5
shi4:   MOV  A,X             ;重新把 X 装入 A
        RL  A                ;2X→A
        MOV  R0,A
shi5:   MOV  Y,R0
        RET
```

(3)散转移。

散转移指令可实现根据某种已输入的结果,使程序转向各个处理程序。使用散转移指令的步骤为:

①建立一个转移指令表,表格首地址装入 DPTR 中;

②使分支指针指向 A,根据 A 值的不同,程序转向不同的分支。

【例 4-3】 假设键盘上有 4 个功能键,处理功能如下:

键盘	功能	键盘	功能
0 号键	减 1	1 号键	加 1
2 号键	右移	3 号键	左移

分析:将按键值存在片内 RAM 中,利用散转移指令,根据不同的按键值,对内部数据区某单元内容实现不同的处理。

程序设计的关键是建立一个转移指令表。可使用无条件转移指令 AJMP 指向处理程序入口,因它为双字节指令,各转移地址依次相差两个字节,所以累加器 A 中按键值应乘以 2,AJMP 指令的分支范围为 2KB。若使用 LJMP 作为处理程序入口,因为它是三字节指令,A 中按键值应乘以 3,其分支范围为 64KB。

程序清单:

```
        KEY  EQU  60H        ;定义按键存储单元
        DIST  EQU  80H       ;定义处理单元
```

```
        ORG   0030H
LOOP:   MOV   A,KEY             ;取按键值
        MOV   DPTR,#TAB         ;建立转移指令表
RL   A  ;修正转移地址,采用 AJMP 指令
        JMP   @A+DPTR           ;利用散转移指令实现程序处理
TAB:    AJMP  JIAN              ;程序处理表格
        AJMP  JIA
        AJMP  RIGHT
        AJMP  LEFT
        ……
JIAN:   DEC   DIST
        RET
JIA:    INC   DIST
        RET
RIGHT:  MOV   A,DIST
        RR    A
        MOV   DIST,A
        RET
LEFT:   MOV   A,DIST
        RL    A
        MOV   DIST,A
        RET
```

3. 循环结构程序设计

(1)循环程序结构。

在处理实际问题时,有时要求某程序段多次重复执行,就可利用循环结构实现。可使程序简练,节省存储单元。典型循环结构如图 4-3 所示。

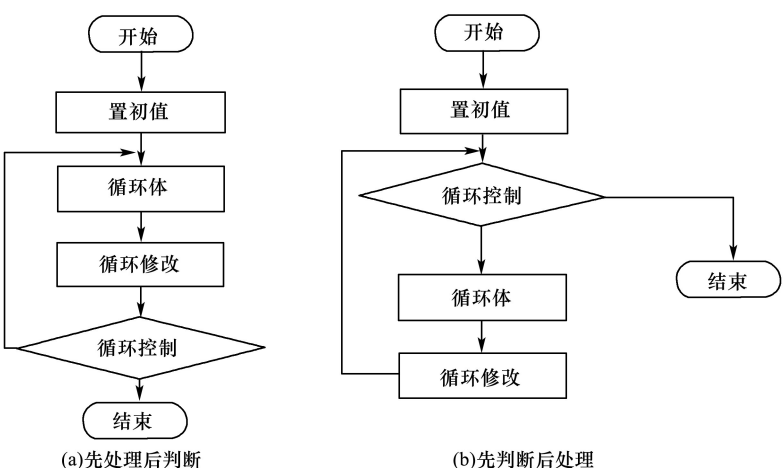


图 4-3　循环结构

循环程序一般包括置循环初值、循环体、循环控制部分和循环修改四部分。

• 置循环初值部分用来设置循环初始状态,如建立地址指针、设置循环计数器初值

等；

• 循环体是重复执行的数据处理程序段；

• 循环修改包括修改地址指针、修改循环变量等，每循环一次，都要对其修改；

• 循环控制用来控制循环继续与否，当不满足终值条件，重复执行循环，当满足终值条件，则退出循环。

循环程序分单循环和多重循环。根据循环结束条件不同，也可以分成循环次数是已知的和未知的循环程序。编写循环程序的关键是如何控制循环的继续与否，若循环次数是已知的，则可用循环次数计数器控制循环继续与否；若循环次数是未知的，可按问题的条件控制循环的结束与否。

(2)单循环结构程序设计。

单循环结构是指循环程序的循环体中不再包含循环程序。例如，在工程中，典型数据的采集通常采用求多个采集数据和，再取平均值的方法。

【例 4-4】 设采集的数据存放在片内 RAM 80H 开始的 10 个单元中，计算该 10 个 BCD 数的累加和。

分析：本例为已知循环次数，可先进行循环体处理，再判断循环是否结束。10 个 BCD 数累加，其结果可能变为双字节数，所以将累加结果存放在 R3、R4 中(R3 存高位)。

设计中要注意：①需要先把累加和设为“0”；②要注意 BCD 码的调整。程序流程如图 4-4 所示。

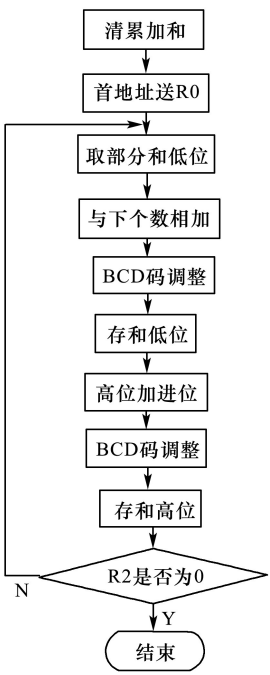


图 4-4 累计求和流程图

程序清单：

```
        ORG   0030H
MAIN:   MOV   R3,#00H       ;清累加和高位
        MOV   R4,#00H       ;清累加和低位
        MOV   R2,#10        ;循环次数
        MOV   R0,#80H       ;源数据首地址
LOOP:   MOV   A,R4          ;取部分和低位
        ADD   A,@R0         ;部分和低位与下个数相加
        DA    A             ;BCD 数调整
        MOV   R4,A          ;存部分和低位
        MOV   A,R3          ;取部分和高位
        ADDC  A,#00H        ;加进位
        DA    A
        MOV   R3,A          ;存部分和高位
        INC   R0            ;修改源数据单元指针
        DJNZ  R2,LOOP       ;判断数据是否累加完?
        RET
```

【例 4-5】 有以“*”作为结束符的一串字符(长度不超过 100),存放在片内 RAM 80H 开始的单元中,要求统计字符串的长度,并存入寄存器 R1 中。

分析:本例属于未知循环次数的单循环程序,程序运行中应先判断字符串是否结束,再进行程序处理,其流程如图 4-5 所示。

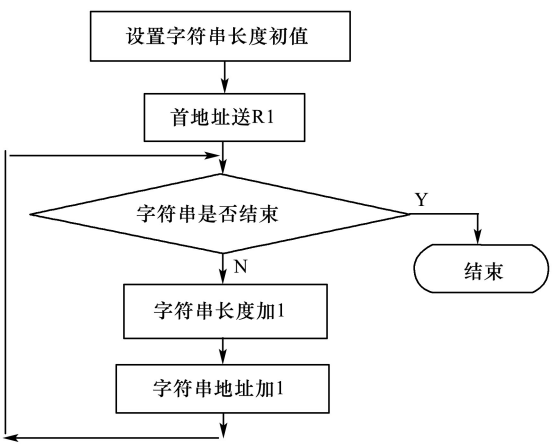


图 4-5　统计字符串长度流程图

程序清单：

```
        ORG   0030H
        MOV   R1,#00H               ;字符串长度初置
        MOV   R2,#80H               ;字符串首地址
LOOP:   CJNE  @R2, #'*',LOOP1       ;比较字符串是否为结束符
        RET
LOOP1:  INC   R1                    ;字符串长度加 1
```

```
        INC   R2                    ;修改字符串首地址
        AJMP  LOOP
        END
```

工程实践中,若采集到的临时数据很多,片内 RAM 存储单元容量有限时,可把数据移到片外 RAM 存储单元。

(3)多重循环设计。

一个循环程序的循环体中还包含一个或多个循环的结构,称为双重循环或多重循环。在设计多重循环程序时,各循环层次要分明,不能出现层次交叉的情况,否则将引起程序运行的混乱。

【例 4-6】 设计延迟 10 ms 的子程序,单片机晶振是 6MHz(机器周期 T 为 2 μs)。

分析:①执行每条指令都需要一定的时间,通过反复执行几条指令,就可实现延时;

②单片机晶振是 6 MHz,则机器周期 T 为 2 μs。

程序清单:

```
          ORG   0800H
DEALY:    MOV   R6,#10              ;置外循环计数 10 次
DEALY1:   MOV   R7,#250             ;内循环计数 250 次
DEALY2:   DJNZ  R7,DEALY2           ;该指令为双周期指令
          DJNZ  R6,DEALY1           ;外循环 10 次
          RET
```

循环程序的延时时间为:(2 μs×2×250)×10=10000 μs=10 ms,若要精确计算时间,应该考虑其他指令执行时间因素。若需要更长延时,可采用多重循环。

4. 查表程序设计

在复杂的函数运算程序设计中,有时先把其运算结果按一定规律编制成表格,存放在计算机的 ROM 中,当程序中用到这些函数值时,直接在表格中寻找其对应的值即可,该方法称为查表法程序设计。单片机中有两条功能相同的专用查表指令:MOVC A,@A+DPTR 与 MOVC A,@A+PC。

• 远程查表指令:MOVC A,@A+DPTR。

该指令查表范围为 64K。一般情况下,在程序中需用立即寻址的方式设置表格首地址 DPTR。

• 近程查表指令:MOVC A,@A+PC。

该指令查表范围为 2K。需修正偏移量,使用起来较繁琐。

(1)简单查表法。

简单查表法是指变量与查表得到的函数值是一一对应的关系。比如:使用数码管显示数字,其代码就可以通过查表方式获得。

【例 4-7】 假设片内 RAM 80H 单元中存有一个压缩 BCD 码,用查表的方法,将其共阴极 LED 显示的代码存入片内 RAM 81H、82H 单元中(81H 单元存放高位 BCD 码对应的 LED 显示码)。

分析:首先应建立一个供查表使用的共阴极 LED 数字显示代码表,再把压缩 BCD 码拆成单字节,并查找其对应代码。

程序清单：

```
LOOP:MOV  DPTR,#TAB        ;取表格首地址
     MOV  A,80H            ;取压缩 BCD 码
     ANL  A,#0FH           ;屏蔽高位
     MOVC  A,@A+DPTR       ;查表取 LED 代码
     MOV  82H,A            ;存低位的 LED 代码
     MOV  A,80H            ;重新取压缩 BCD 码
     ANL  A,#0F0H          ;屏蔽低位
     SWAP  A
     MOVC  A,@A+DPTR
     MOV  81H,A
TAB: DB  3FH,06H,5BH,4FH,66H,6DH,7DH,07H
     DB  7FH,6FH,77H,7CH,39H,5EH,79H,71H
     RET
```

(2)多字节查表法。

上例中，查表得到的函数值为单字节，实际应用中，函数值多数是多字节，但是单片机的函数运算指令很少，因此，函数值可以通过查表法获得。

【例 4-8】 用查表法求 $Y=X^4$，X＝0，1，2，3……9

分析：①$9^4=6561=19A1H$，函数值要用 2 字节表示。首先建立每两个单元存放 0，1，2，3……9 的 4 次方函数值的表格。

②设变量 X 值存放在片内 RAM 80H 单元，函数值 Y 存放在片内 RAM81H、82H 单元。

程序清单：

```
        ORG  0030H
        X  EQU  80H
START:  MOV  DPTR,#TAB          ;取表格首地址
        MOV  A,X                ;取 X
        RL  A                   ;X×2
        MOV  R3,A               ;暂存
        MOVC  A,@A+DPTR         ;查表,取低位
        MOV  81H,A              ;存低位
        MOV  A,R3               ;重新取数
        INC  A                  ;指向下一个单元
        MOVC  A,@A+DPTR         ;查表,取高位
        MOV  82H,A              ;存高位
        RET
TAB:    DW  0000,0001,0016,0081,0256
```

5. 子程序设计

(1)子程序结构。

在同一个程序中，经常会遇到许多地方需要执行同样的一项任务，而该任务又并非规

则情况，不能用循环程序来实现，这时，可以对这项任务独立进行编写，形成一个子程序。在主程序中需要执行该任务时，调用子程序，执行完该任务后，又返回主程序，继续以后的操作。这样简化了程序的逻辑结构，节省程序空间，更便于调试。

子程序的特点是在功能上具有通用性，结构上具有独立性。子程序功能是执行完子程序后通过堆栈内的断点地址弹出至 PC 返回到主程序，如图 4-6 所示。子程序与一般程序的主要区别是子程序末尾有一条返回指令(RET)。

(2)子程序设计的注意事项。

• 子程序应有唯一的名称，以便主程序正确地调用。

• 子程序以 RET 指令结束，返回主程序，不可以用长转移或短转移指令返回主程序。

• 调用子程序时，子程序如何从主程序处获得有关参数(入口参数)；返回主程序时，主程序如何从子程序得到需要的结果(出口参数)，这就是所谓的“参数传递”。

主程序与子程序之间的参数传递一般有三种方法：利用寄存器传递参数；利用寄存器间址传递参数；利用堆栈传递参数。

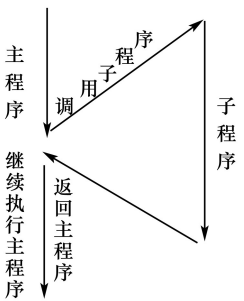


图 4-6 子程序调用流程

• 子程序使用中要注意保护和恢复现场。即进入子程序时注意对正在使用的寄存器的保护，返回主程序时应将它们恢复原来的状态。

(3)子程序的调用和返回。

• 主程序调用子程序指令有 ACALL 和 LCALL，其功能是将断点(调用指令的下一条指令地址称为断点)压入堆栈，实现断点的保护；然后将子程序首地址送入 PC，使程序转入子程序执行。

• 子程序返回指令是 RET，其功能是将堆栈中存放的返回地址(即断点)弹出堆栈，送回到 PC 去，使程序继续从断点处执行。

一个主程序可多次地调用同一个子程序，也可以调用多个子程序。子程序也可调用其他子程序，称为子程序嵌套。

主程序在调用子程序时，要注意以下设置。

• 在主程序中要安排相应的指令，提供子程序的入口条件即入口数据。

• 在主程序中要安排相应的指令，处理子程序提供的出口数据。

• 在需要保护现场的程序中，正确设置堆栈参数。

【例 4-9】 用调用子程序的方法求函数式 $W=x^2+y^2+z^2$ 的值。其中，x、y、z 是小于 9 的自然数。

分析：①因为，x、y、z 是小于 9 的自然数，其平方和为单字节数，且函数式的值不大于 243，也为单字节数，所以，变量 x、y、z、W 可以各占用一个单元。

②本例中三次用到平方值，所以可将求平方的程序段作为子程序。

主程序清单：

```
x       EQU  80H
y       EQU  81H
z       EQU  82H
```

```
W       EQU  83H
        ORG  0030H
MAIN:   MOV  A,x            ;取 x 值
        LCALL  SQR          ;求 x 平方
        MOV  B,A            ;暂存
        MOV  A,y            ;取 y 值
        LCALL  SQR          ;求 y 平方
        ADD  A,B            ;x² + y²
        MOV  W,A            ;存入 W
        MOV  A,z            ;取 z 值
        LCALL  SQR          ;求 z 平方
        ADD  A,W            ;x² + y² + z²
        MOV  W,A            ;存入 W
        SJMP  $
                            ;子程序:入口参数是累加器 A 的数值(小于 9 的自然数),
                            ;出口参数也是累加器 A(平方数)。
SQR:    INC  A              ;修正偏移量
        MOVC  A,@A + PC     ;查平方表
        RET
TAB:    DB  0,1,4,9,16,25,36,49,64,81
```

4.4　本章小结

本章通过单片机汇编语言程序设计实例,详细介绍了单片机程序结构与设计方法,程序设计流程图的绘制,概要介绍了单片机的伪指令,如图 4-7 所示。

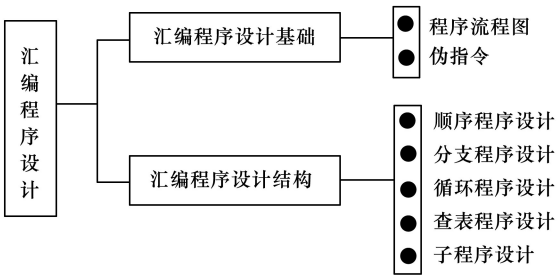


图 4-7　第 4 章小结

4.5　强化练习

1. 思考题

(1)简述汇编语言程序设计步骤。

(2)简述采用程序流程图有什么好处。

(3)简述“伪指令”概念、伪指令与指令的区别。

(4)循环程序由哪几部分构成？若要优化循环程序,首先该优化哪一部分？为什么？

(5)简述子程序设计的注意事项。

2. 设计题

(1)编程实现:

$$Y=\begin{cases}X & X<10\\ X^2 & 10\leqslant X\leqslant 15\\ 2X & X\geqslant 15\end{cases}$$

(2)根据图 4-8 所示流程图,编写程序,说明程序功能。

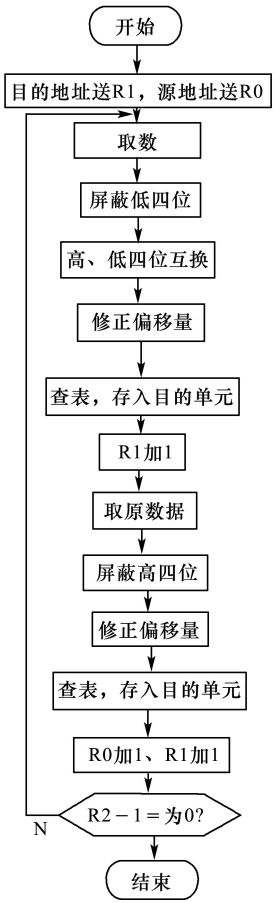


图 4-8 习题 2(2)图