程序设计基础

（下 册）

编 者 吴英泽

沈阳航空工业学院

一九八七年九月

目　　录

第五章　程序设计的基本方法

我们知道，任何一个可执行的程序中的每一条指令，都必须明确指出将要进行的操作，操作数以及与操作数有关的地址。用机器语言编写的程序虽然能直接描述这些内容，但这种程序的编制，修改、阅读都十分困难。汇编语言使机器语言符号化，用英文单词或简写来表示操作，用字符串等来表示地址，从而克服了机器语言编制程序的一些缺点，使程序的编制、修改、阅读和交流都比较容易。同时，汇编语言又保持了机器语言的特点，使得可以充分地利用计算机硬件资源，编制出质量较高的程序。因此，计算机系统中有许多软件都用汇编语言来编写，一些高级语言的编译程序所产生的目标程序也采用汇编语言的形式。鉴于汇编语言在计算机系统中占有的重要地位，学习汇编语言的程序设计是计算机专业人员必须接受的最重要的专业基础训练之一。

从本章开始，我们将介绍用汇编语言进行 程序设计的基本方法。

§5.1　软件设计方法概述

一个好的程序，必须保证是正确的，且有合理的结构和简明易读的特点。程序在某种范围内的正确是其使用的先决条件，而结构合理的程序在调试、修改、维护和扩充时比较容易。程序在阅读、推广和使用时则要求具有良好的可阅读性。在满足以上要求后，进而追求程序编写的技巧和运行效率。因而，为了编制一个高质量的程序，应该有正确的程序设计思想和风格，遵循一定的软件设计规范。这里我们把通用的软件设计步骤归纳为以下几个基本部分。

一、题目的任务说明

如果解答的问题是软件问题，必须准确地决定编成的软件系统将完成什么任务。规定软件系统所应有的工作性能叫任务说明。任务说明包括下述内容：

1、简要的题目说明。它描述了系统所要解决的问题。

2、硬件。了解哪些信号及设备是可以使用或需要使用的。

3、软件接口。设计程序时，经常要把它们放进系统。在系统中，设计的程序将会同其它程序共存或使用其它程序。这些都必须在功能说明书中加以明确。

4、说明系统最终所要具备的功能。这部分包括与用户的对话，所需的数据，输出结果，特殊功能，对错误的处理等。

一个好的功能说明是重要的。它是建立软件系统的基础。如果说明得不适当、不完

全，而在软件设计开始之后仍然在修改或增加一些功能，就要付出极大的劳动甚至重新进行设计。

二、程序模块化

完成了软件系统的任务说明后，可以把系统分成若干功能模块。模块是系统的组成部分，每个模块都能完成系统的某个特定功能，并且有简单的入口和出口。如图14所示。在系统运行的过程中，控制可以从一个模块转向另一个模块。进行一个复杂的课题前，要把程序结构设计成一系列前后连贯的模块，然后再进一步对每个程序模块进行设计。

划分和设计程序模块是把任务说明书变成程序的重要步骤。问题需要分成多少级功能模块，模块之间的控制和调用关系的确定，是和程序设计者对问题的分析和程序设计的经验分不开的。模块的划分具有一定的灵活性，以便于我们能较容易地把功能模块等级确定得适应题目的复杂程度。

通常我们是根据任务说明书来确定系统最初应分成什么样的功能模块。常见的功能模块如：输入，输出功能模块。系统的程序控制模块，程序功能模块（如数据传送模块、存贮器查找模块、运算模块等）以及主要的数据结构模块等。把这些功能模块连接起来构成了系统的程序结构框图。建立起系统的程序结构框图，应能确保已经把所有的输入、输出、数据转换，系统功能及故障状态都考虑在内了。如果有些功能只是在某些模块中部分地被定义，则还应在这些地方增加一些子模块。这些过程反复进行，直至确信程序结构框图定义的系统与任务说明书相符合。以后，就可以为实现各个程序功能模块而进行所需要的具体逻辑设计了。

三、选择合适的算法

1、算法的描述

进行了系统的程序结构设计，实现了程序的模块化，我们就可以通过逻辑设计过渡到设计系统所需要的实际逻辑。在进行实现具体模块功能的设计时，首先要确定处理问题的方法和大体上的逻辑步骤及顺序，或者说先确定算法的思路。把解决问题的方法步骤具体化，就是通常所说的算法。算法是求得某一类问题的解的一个过程，这个过程是由一套明确的规则所规定并按一定顺序执行的一些有限的步骤所组成。一类问题可以同时存在几种算法，评价算法的标准主要应该看程序执行的时间，占用计算机的存贮空间，

设计该算法和编写程序的人力以及理解该算法是否简易清晰，扩充性，适应性等。

算法可以用自然语言描述，也可以用程序设计语言描述和流程图（框图）描述。自然语言描述算法比较容易理解和进行交流，但由于自然语言表达时有二义性、不明确性、所以在描述一些复杂情况的算法时，不易追随其中的逻辑流程，并且在编写程序时也较困难。因此这种表达方法只适用于对简单问题的描述。

用程序设计语言来描述算法比较简洁，而且可以直接上计算机进行处理。但这种方法表达时，对算法的逻辑结构不容易看的清楚。而且鉴于一种程序设计语言的局限性，使得用语言描述算法交流起来也不太方便。因此，现在通用的方法是半自然语言来描述算法。所谓半自然语言就是把程序设计语言中的某些语句和自然语言混合在一起使用。半自然语言也称为类程序设计语言。

流程图描述算法是一种传统上常用的方法。流程图是一种用特定的框状的图形符号加上简单的文字说明来表示数据处理过程的步骤。它指出了计算所执行操作的逻辑次序，而且表达得简明，清晰。使设计者可以从流程图上直接观察整个系统，了解各部分之间的关系，排除设计错误。

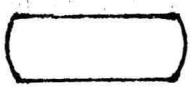
2、关于流程图

数据处理工业已有了一套标准的流程图符号，最经常使用的符号和它们的功能如图4.1.1所示。

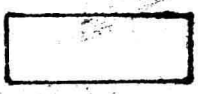
我们能画出足以描述任何复杂程度算法的流程图。最通用的二种流程图是逻辑流程图和程序流程图。逻辑流程图用通用操作术语来描述算法，而不涉及具体的计算机功能（如寄存器、存储器、标志位等），逻辑流程图也称作算法流程图。而程序流程图则根据某台具体的计算机所具备的功能来描述算法。开始时，可以用算法流程图来描述程序结构框图中的每一个功能模块，而后以它们作基础，用来编制与计算机有关的所需要的流程图。如果使用高级语言来编程，只需要用逻辑流程图来定义算法，而所有与计算机有关的细节都将由语言处理程序来实现。同样，逻辑流程图也用来描述在不同计算机上所执行的通用算法和解法，以作为在任何一台计算机上或应用任何一种语言来实现一种解法的基础。

对流程图的使用存在一些不同的看法。有人认为：流程图用来绘制图形，写说明等很不方便，而且修改流程图是很困难的事情，更不便于用计算机的处理功能来处理和保存。尽管如此，流程图描述算法的优点还是主要的特别是对于使用汇编语言的初学者，

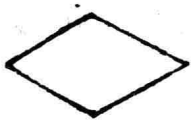
由于要同时学习编制程序，算法设计和计算机结构，使用流程图对他们进行这些学习活动是很有帮助的。为此，我们将使用通用逻辑流程图来介绍所用的算法的思路和结构。在一些例子中，还使用一些和计算机有关的程序流程图，并以此来编写汇编语言程序。



终端框或中断框。表示程序开始或结束，也用于表示中断操作。



程序框。方框内表示的是要完成的功能。



判定框。程序的流向决定于框内所标注的条件。

予定义程序框。用来代表框内未详细说明的一个或一组操作。如调用的子程序。

程序的流向线。用来指向下一个要完成的操作。

连接符。用于连接不在同一张纸上或不在同一处的程序流。

图5.1.1　常用的流程图符号及其功能

3、怎样设计算法

设计一个好的算法是不容易的，算法的设计和具体的课题也有直接关系，在这里我们只介绍把程序结构模块变换成算法的一些通用方法。

首先明确程序模块的任务，用一二句话来描述所要进行的操作。确定被操作的数据在什么地方，是读入的，或是从别的功能模块中传送过来的，或是从表中查出的等。我们将用程序操作框输入所需数据。当取得数据之后，还应确定，在使用数据之前，是否还需要对数据作处理。例如：要不要求补？要不要循环移位？以及要不要截取数据的部分？要不要换算等等。若需要，还应在流程图中有一些数据变换的程序框。

进一步应该考虑如何来完成所需要的操作，这是算法设计的关键。在这里，可以把数据处理程序框用数据框及各种判定框结合起来，从而将数据从输入格式变换成输出格

式。这段过程在整个算法设计中要占据很大的部分。算法设计是一个反复的过程，获得正确的算法之前，通常要进行多次的尝试。编写算法时，纠正错误逻辑将会节省在系统调试阶段的时间。

有了一个可以使用的算法之后，应该用数据检验一下，看其是否能正常工作。要设想可能出现的每一种数据情况，并确保在每一种情况下算法都能正确处理。然后确定怎样处置结果数据，要不要进行专门的格式化？要不要保存？要不要返回主程序及输出？

一个好的算法的标准应该是结构清晰，流程简捷，符合逻辑、易于阅读的。实际上，在编制任何算法时，真正使用的只不过是几种简单的程序结构，如顺序结构和分支结构，循环结构以及栈和子程序结构。这些基本结构可以实现任何一种有明确定义的算法，严格地按照这几种结构的编程方法叫结构化程序设计方法，使用这些结构可以大大简化程序校验，修改，以及调试的工作。

四、程序设计和调试

在完成了题目的定义，程序结构框图和各模块的算法设计以后，就可以编制实现各算法所需要的程序了。通过前一阶段几步的工作，程序模块的划分和算法设计已经给出了系统结构和控制逻辑，编程时只是用已有的编程语言来实现这些功能。

利用汇编语言编制程序实际就是寻找一些机器指令的组合以实现所需要的操作。为了编制一个高质量的汇编语言程序，大体应该注意以下几点：

1、编程时对使用的指令应有选择，能用简单和常用指令的地方决不用复杂的和不常用的指令。因为使用后一类指令会增加调试和阅读程序的困难。

2、汇编语言的优点之一就是能充分利用计算机资源。编制程序时，应该利用这个优点，充分合理的使用寄存器，尽量使操作在寄存器之间进行，而较少访问内存单元，以便节省时间，提高效率。

3、程序中的数据单元和工作单元集中定义在数据段内，和代码段中的可执行指令分开。从而使得程序结构清楚，便于阅读和修改。

4、程序的可执行指令代码在程序执行过程中应保持不变。编制自身修改某些指令代码的变指令程序会给程序的阅读和调试带来很大的困难。

5、程序中应用简明而充分的注释。注释可以描述程序段和指令在程序中所起的作用。简明而充分的注释对程序的调试、修改、阅读和交流都有重要的影响。

汇编语言是面向计算机的语言，影响程序的因素较多，程序设计人员甚至必须安排

计算机动作的细节，所以用汇编语言进行程序设计比用高级语言进行程序设计要困难的多。在编制一个高质量的汇编语言程序时，要根据所描述的算法，先写出一些能表达算法结构的关键性语句，形成在逻辑上能反映算法结构的程序框架，然后再反复充实框架之间的具体内容。在充实内容的过程中，可及时修改和扩充算法中不完备甚至错误的地方。经过这样反复修改、求精，能使得到的算法较好地解决题目提出的任务，并使编出的程序具有较高的质量。这种编程的过程就是反复修改，逐步求精的过程。

程序编制完成后，必须经过调试才能交付使用。程序的质量如何也只有经过调试之后才能鉴别。程序的调试是一个繁琐而又细致的过程。通常都是先调试各个可以独立运行的程序模块，各程序模块调通后，再进行各模块间的联调。调试时，除了根据题目的特点采用一般的调试方法外，还必须尽可能地使用系统提供的调试工具，从而提高调试的效率。调试过程中，程序和算法可能还要经过一次反复修改，逐步求精的加工处理。经过了调试阶段的程序，有了一定的质量水平，才可以投入实际运行。

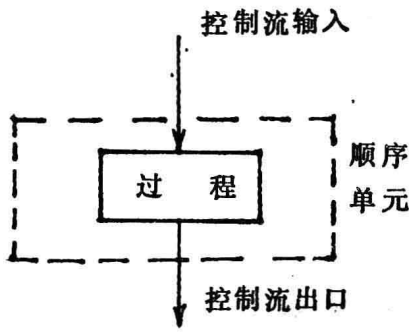
五、结构程序设计

按照结构程序设计的编程方法，所有的程序都严格地限制只利用少量的逻辑上完善的软件结构来构成。使用这一套在逻辑上完善的结构可以实现任何一种有明确定义的算法。在结构程序设计中使用的合法的基本软件结构都只具有单一入口和出口，各个基本软件结构之间通过专门设计的一些独立的控制软件结构联系在一起，而只有极少数软件结构间的转移，从而避免了不加限制地使用程序分支。用结构程序设计方式写出的程序便于阅读和理解、修改和维护也更加容易。很多高级语言中部分语句就是这样一些基本程序结构，如顺序程序结构（图5.1.2ⓐ），分支程序结构IF－THEN－ELSE和IF－THEN（图5.1.2ⓑⓒ），循环程序结构DO－WHILE和REPEAT－UNTIL（图5.1.2ⓓⓔ），栈和子程序结构等。这几种基本程序结构虽然为数不多，但了解其每种结构的功能是必要的，比较大型的而功能复杂的程序都可以由这些基本程序结构进行组合得到。结构程序设计的基本概念和基本程序结构是可以普遍适用的，在本章以下的几节中我们将介绍利用汇编语言实现这几个基本程序结构的方法。

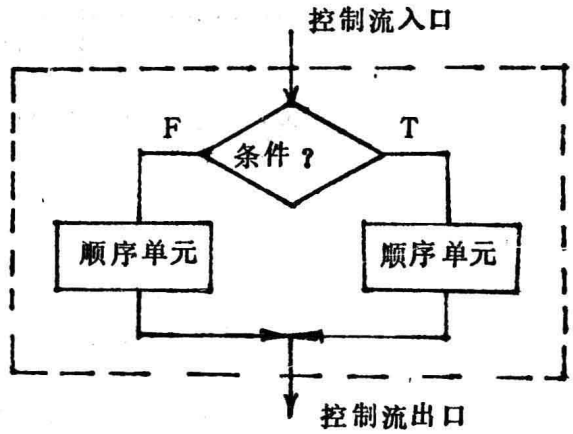
事实上，结构程序设计的思想和方法并不能为人们所普遍接受。有人认为使用的结构不必符合象基本的单入单出规则那样加以严格限制，另一些人则认为应该使程序的基本结构受到更严格的限制以提高程序的可靠性和易检验性。更有人认为结构程序设计限

制了提高程序的效率，增加了程序的规模，而且不能十分有效地使用现成的计算机资源和语言资源，在一定程度上限制了程序设计人员的创造性。

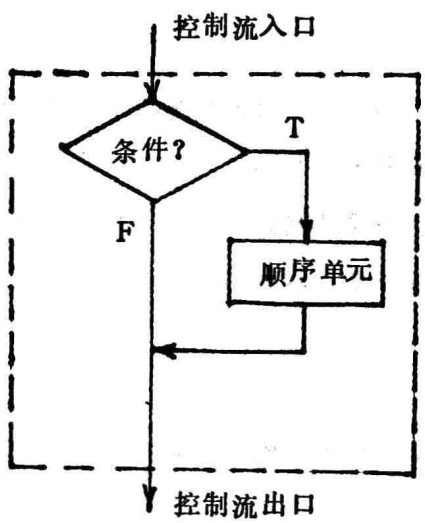
虽然人们在程序结构的标准化方面做了大量的尝试，取得了很大的成绩。但编制程序毕竟还是属于个人的活动。我们应当学会并使用简单的程序结构，努力使得程序的操作更加简单、明白、方便，而且可以不考虑是否利用了结构程序设计技术。许多情况下，采用非标准化的结构和启发式程序设计方法可能会更好的解决问题。在今后对程序设计更深入的学习中我们能进一步对此予以体会。



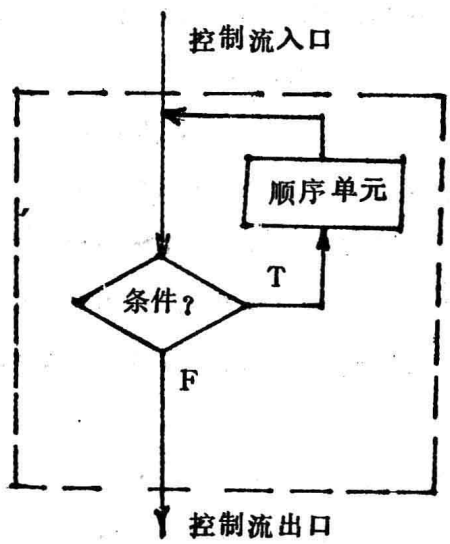
Ⓐ 顺序结构



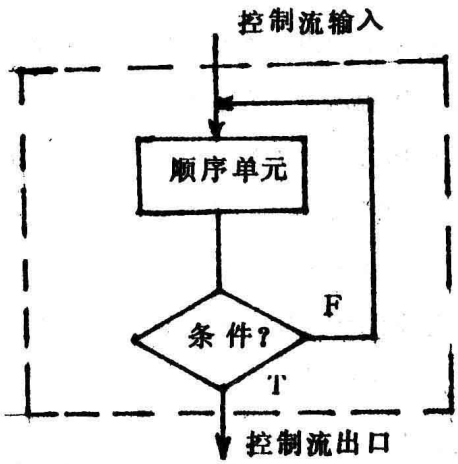
Ⓑ IF-THEN-ELSE结构



Ⓒ IF-THEN结构



Ⓓ DO-WHILE结构



⑤ REPEAT-UNT IL结构

图5.1.2 基本程序结构

§5.2 顺序程序设计

程序的基本结构中，顺序结构最为简单。在流程图中，一个接着一个的处理框就是描述顺序结构的算法。在高级语言的程序中，这种结构主要用赋值语句和过程语句组成。而在汇编语言程序中，顺序结构主要用数据传送和算术及逻辑运算指令组合形成。

一、表达式程序设计

任何复杂的运算问题，要在计算机上实现，最终都是要把问题转化为算术运算和逻辑运算来完成。算术表达式和逻辑表达式的计算是最基本的计算。高级语言中算术表达式和逻辑表达式可以用一个赋值语句来实现，而赋值语句的功能在汇编语言中可以是由一个指令序列来完成的。我们把这个指令序列称为表达式程序。

例5.2.1 设计计算算术表达式

$$(3x+4y)/2$$

值的表达式程序。式中 x，y 均是已知十六进制正整数。假设计算过程不产生溢出。

程序设计过程一般应包括以下几个步骤：

1、在分析表达式运算的先后次序的基础上形成计算过程的大致流程。并画出如图

5.2.1ⓐ所表示的逻辑流程图。图5.2.1ⓐ中 v、u、w、z 表示在计算算术表达式过程中产生的中间结果值。

2、有了求解问题的逻辑流程图，还不能直接写出程序。结合计算机具体情况，在编写程序前，要对存贮器的使用情况和寄存器的使用情况进行安排和考虑。即为指令、原始数据、中间结果和最终结果等分配必要的内存空间和寄存器。在 IBM PC宏汇编语言的程序中，通常把数据和指令程序分别安排在二个独立的段，原始数据 x、y 在数据段中用符号名ADRX，ADRY 表示的变量中，表达式结果存贮在数据段中用符号名ADRZ表示的变量中。中间结果 v、u、w、z 用寄存器来暂存。结合存贮分配和寄存器的使用，把逻辑流程图5.2.1ⓐ进一步改造成与 IBM PC宏汇编语言有关的程序流程图5.2.1ⓑ。在程序流程图中方括号〔〕表示直接或间接从内存中取得操作数。

程序流程图5.2.1ⓑ是对逻辑流程图5.2.1ⓐ中每一框内的操作要求进一步细化成能用一条或几条指令来实现的操作而得到的。这个过程也叫做工程化，所以图5.2.1ⓑ也称作工程流程图。有了程序流程图，就能容易地用符号指令来实现流程图中要求，即编制出相应的程序。

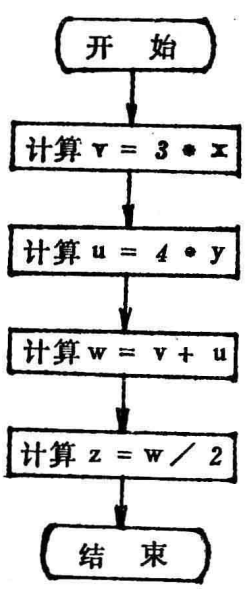


图 5.2.1 逻辑流程图

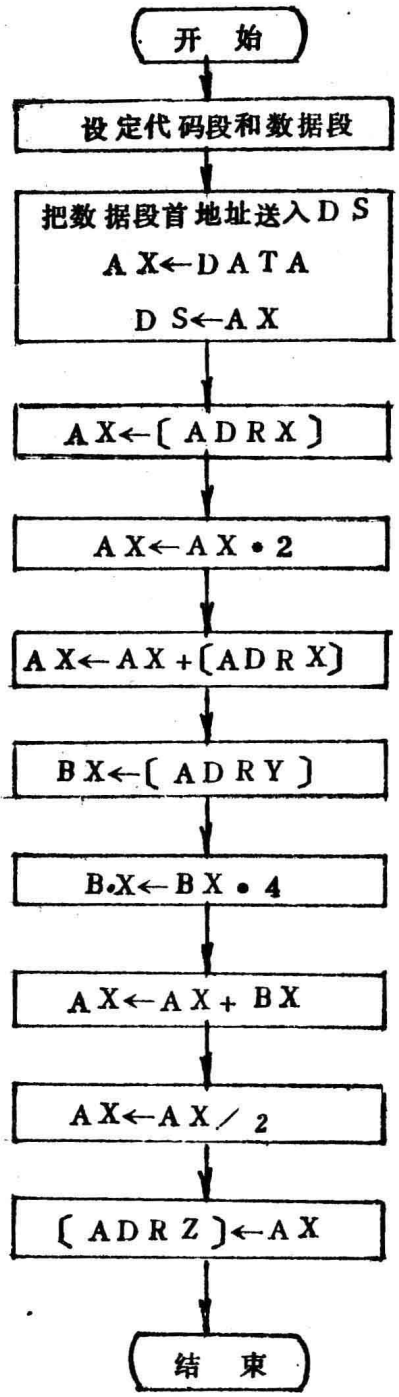

开 始
设定代码段和数据段
把数据段首地址送入DS
AX←DATA
DS←AX
AX←〔ADRX〕
AX←AX•2
AX←AX+〔ADRX〕
BX←〔ADRY〕
BX←BX•4
AX←AX+BX
AX←AX/2
〔ADRZ〕←AX
结 束

图 5.2.1 ⑤ 程 序 流 程 图

< 程序5．2．1 >

```
DATA  SEGMENT
ADRX  DW  2BH          ; x=2BH
ADRY  DW  3ACH         ; y=3ACH
ADRZ  DW  ?            ; 存结果
DATA  ENDS
CODE  SEGMENT
      ASSUME  CS:CODE, DS:DATA
START:MOV  AX, DATA    ;  取数据段首地址
      MOV  DS, AX      ;  DS赋值
      MOV  AX, ADRX    ;  AX←x
      SAL  AX, 1       ;  AX←x*2
      ADD  AX, ADRX    ;  AX←x*3
      MOV  BX, ADRY    ;  BX←y
      MOV  CL, 2       ;  CL←2
      SAL  BX, CL      ;  BX←y*4
      ADD  AX, BX      ;  AX←3x+4y
      SAR  AX, 1       ;  AX←(3x+4y)/2
      MOV  ADRZ, AX    ;  [ADRZ]←AX
      HLT              ;  停机状态
      CODE ENDS
           END  START
```

二、逻辑运算程序设计

因为逻辑值有真假之分，我们只要用一位二进制数字便可以表示一个逻辑值。通常规定用“0”表示假，“1”表示真。但在计算机中，算术逻辑运算总是对一个字节或字整体进行处理的，为此，我们把一个字节作为一个逻辑字。当该单元8位为“全1”时，就说该逻辑值为真；而当该单元8位为“全0”时，就说该逻辑值为假。这种用一个单元作为一个逻辑字的表示方法与用一位二进制数字表示一个逻辑字的方法并没有什

么本质的区别，只是形式上方便一些。

例5．2．2　假设A、B、C、D为已知的逻辑值，编制计算逻辑表达式

$$(\overline{A} \vee B) \wedge (C \vee \overline{D})$$

的值的程序

分析该表达式中的所有逻辑运算都可以用逻辑运算指令语句实现，我们可以直接写出程序流程图5．2．2。

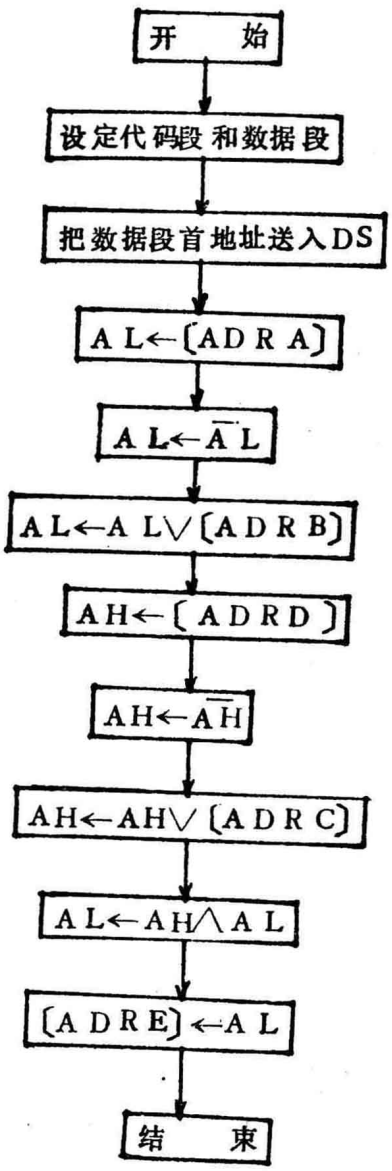


图 5．2．2　程序流程图

<程序5.2.2>

```
DATA    SEGMENT
ADRA    DB    0FFH          ;  A初值为真
ADRB    DB    00H           ;  B初值为假
ADRC    DB    00H           ;  C初值为假
ADRD    DB    0FFH          ;  D初值为真
ADRE    DB    ?             ;  存结果
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS, CODE, DS, DATA
START:  MOV   AX, DATA
        MOV   DS, AX
        MOV   AL, ADRA      ;  AL=A
        NOT   AL            ;  AL=Ā
        OR    AL, ADRB      ;  AL=Ā∨B
        MOV   AH, ADRD      ;  AH=D
        NOT   AH            ;  AH=D̄
        OR    AH, ADRC      ;  AH=C∨D̄
        AND   AL, AH        ;  AL=(Ā∨B)∨(C∨D̄)
        MOV   ADRE, AL      ;  存AL
              HLT
CODE    ENDS
        END   START
```

从此例可以看到，顺序结构的程序算法步骤之间的逻辑关系简单，程序中不必使用控制转移的指令语句，所以在编制程序时，可以把精力集中在对数据处理指令的理解和应用上。下面通过几个实例进一步了解顺序设计在处理算术及逻辑表达式中的使用。

三、顺序结构程序举例

例5.2.3　设多项式为

$$f(x) = 5x^3 + 4x^2 - 3x + 21$$

编制程序，计算 f（6）的值。

1、分析该表达式，如果采用代入法，直接计算多项式的值，则至少要6次乘法和3次加减法，并且要使用较多的寄存器。如果采用秦九韶算法，把 f（x）变形为：

$$f(x) = ((5x + 4)x - 3)x - 21$$

则只要做3次乘法和3次加减法，并且减少了使用的寄存器。

2、因为算法比较简单，我们可以直接写出程序流程图5.2.3。图中的乘法操作用乘法指令实现。累加器AX和存储器操作数x执行无符号数相乘，结果在DX和AX中返回一个双字长的数。由于该表达式不会有超过16位二进制数表示范围的结果，所以DX中结果为0，程序中只要处理AX中的内容。

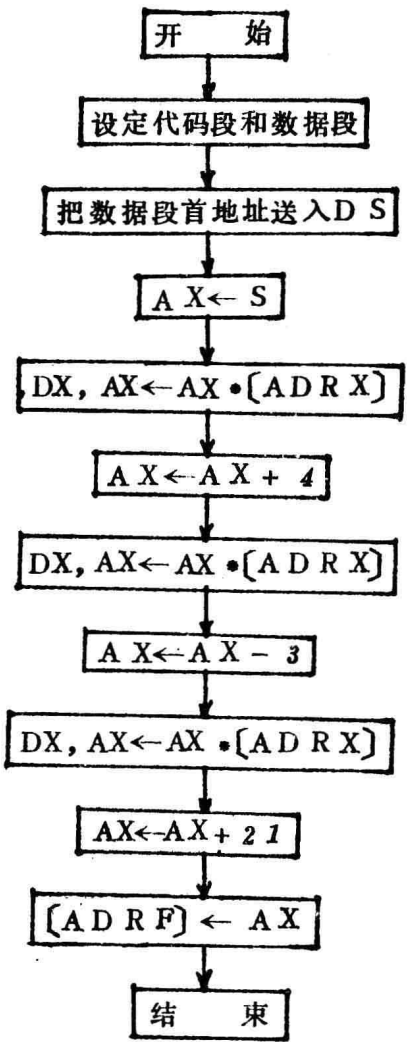


图 5.2.3　程序流程图

程序5.2.3

```
DATA    SEGMENT
ADRX    DW      6           ;   x初值为6
ADRF    DW      ?           ;   存F(6)
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS, CODE, DS, DATA
START:  MOV  AX, DATA
        MOV  DS, AX
        MOV  AX, 5
        MUL  ADRX
        ADD  AX, 4
        MUL  ADRX
        SUB  AX, 3
        MUL  ADRX
        ADD  AX, 21
        MOV  ADRF, AX
        HLT
CODE    ENDS
        END  START
```

从此例可以看到，表达式中的一些常数可以在指令语句中通过立即数的形式得到，不必存放在存贮单元中，从而使程序更简洁。另一方面，立即数的使用虽然能节省一些存贮空间，但指令的执行周期却相应的增加了，在程序编制时我们没有考虑。

例5.2.4　设在内存单元M_1存放着二位以BCD码表示的十进制数，每一位BCD码是4位二进制数。现要把此二位十进制数分别存放在MIG（高位）和MIL（低位）单元中。

1、分离二位十进制数的程序设计，其要求可以用下述示例说明。

设　〔M1〕= 10010110B

分离后〔MIG〕= 00001001B

〔MIL〕= 00000110B

2、根据示例，可以确定分离算法如下：取出该二位十进制数送寄存器后，利用逻辑与指令屏蔽其高四位，得到的低四位存入MIL单元。再次取出该二位十进制数，屏蔽其低四位，得到的高四位逻辑右移四位后存入MIG单元。