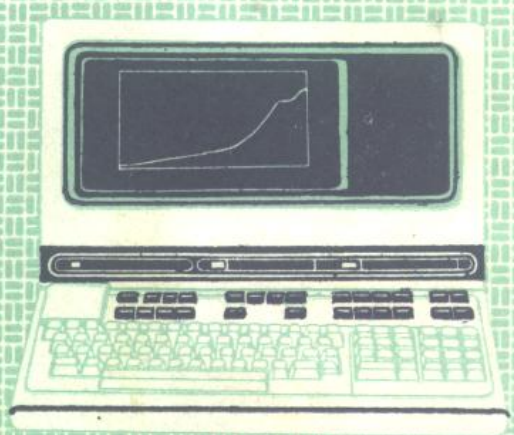
微型计算机
及其在测量中的应用

下　册

谈根林　李慧文　汪庆宝　李礼贤　编著



计　量　出　版　社

微型计算机及其在测量中的应用

（下　册）

谈根林　李慧文　汪庆宝　李礼贤　编著

计量出版社
1985·北京

内 容 提 要

本书以 Intel 8080 微处理机为典型机，按照硬件和软件并重、一般原理和实际应用并重的原则，详细介绍了微型计算机系统。全书共 9 章，分上下册。上册（1—4章）内容包括：微型计算机的基本知识；8080微处理机、指令系统及程序设计初步；微型计算机接口技术和系统连接；微型计算机中断系统和DMA 系统。下册（5—9章）内容包括：程序设计和软件；微处理机在电子测量和控制中的应用；标准总线；Z-80微型计算机和微型计算机系统等。

本书可作为高等院校计算机、自动控制、无线电技术、测量等专业的教学参考书和有关专业研究生的教材，也可供计量测试、自动控制和其他科技人员参考。

微型计算机
及其在测量中的应用
（下　册）

谈根林 李慧文 汪庆宝 李礼贤　编著

责任编辑　陈聪尔

*

计量出版社出版
（北京和平里11区7号）
三河县中赵甫印刷厂印刷
新华书店北京发行所发行　　各地新华书店经售

*

开本 787×1092 1/16　　印张 26 1/4
字数 635 千字　　印数 1—10000
1985年10月第一版　　1985年10月第一次印刷

统一书号 15210·413

定价 5.20元

前　言

从七十年代起，随着大规模集成电路的工艺和技术的发展，现在只用几块甚至一块LSI芯片就可以组成一台微型计算机（简称微型机）。由于微型机价格低廉、可靠性高、系统灵活，从而使计算机的应用更加广泛。原来用小型机的相当一部分场合，现在可用价格低廉得多的微型机来取代；而原来不能想象可以用计算机的许多场合，现在却成了微型机大显身手的地方。微型机的应用已深入到各个生产部门和各个科学技术领域，如：通讯、导航、仪器、仪表、过程控制、交通管理、汽车控制、人工智能、机械手、机器人以至家用电器、游戏用具、儿童玩具，……。如果说十八世纪蒸汽机作为动力机引起了第一次工业革命，那么如今微型机作为智能机将会带来新的工业革命。因此，大力发展和普及微型机的应用对于促进我国实现四个现代化有着重要的意义。

本书是为各行各业中准备使用微型机的技术人员编写的一本微型机入门书，由浅入深地、较全面地介绍了微型机的原理及其应用，原无计算机知识的读者也可阅读。

本书有下列几个特点：

1．以Intel 8080微型机为典型介绍微型机原理。这样做一方面在于8080微型机的原理与应用有普遍性，对于其它微型机如8085、Z-80乃至M6800等也都适用。我国050系列微型机也是以8080为参考的，因此，学习它有直接的实用价值。另一方面本书仅限于讨论一种微型机，可以节约篇幅，从而能对微型机的应用作较为深入的讨论。

2．硬件与软件紧密结合。本书从头至尾都是采用微型机硬件与软件紧密结合的方法来阐述的。当然，为了应用微型机，我们必须将硬件（即大规模集成（LSI）电路）组装成整机，也就是必须懂得硬件原理。但是，这些LSI芯片都是基于系统软件原理而设计的，不懂得软件就不可能懂得这些芯片的互连及其工作原理。只是正确地将硬件互连起来而没有软件配合电路仍不能正确工作。由于大规模集成使微型机硬件的工作量越来越少，相对而言，微型机应用的大部分工作量将是应用软件的设计。因此，现在的电子学专家与过去不同，其工作更多地转向软件；其创造性，常常体现在一个新的算法的提出，一个新的数据结构的设计。这就是为什么在本书中如此强调硬件和软件相结合的原因。

3．微型机化（或微处理器化）设计方法。鉴于微型机的成本低，目前一个新的趋势是在测量和控制设备中尽可能用软件来取代硬件。这样使设备成本降低，便于修改与维护，具有通用性，并提高可靠性。这样就更迫使我们面向软件。微处理器芯片的价格是低廉的，在仪器和控制装置中增加微处理器芯片就使这些设备具有“智能”，使这些设备的功能更强。但实际用户看不到微型机，它隐藏在仪器设备内部，这就是微处理器化。鉴于这一方向的重要性，本书始终贯彻这种微型机化设计思想。因此，本书决不是向读者介绍一台现成的微型机，而是重点介绍组成微型机的建筑模块（LSI芯片），及其建筑技术（系统互连方法）。有了这个基础就可以在应用中随意剪裁，创造出各式各样的经济、灵活、可靠的系统。

4．低级语言和高级语言相结合。在开始应用微型机时，由于缺少软件开发工具，同时程序也不大，多用汇编语言写程序。本书用相当大篇幅介绍汇编语言程序设计，目前这仍是必

要的。但随着开发工具的发展，今后微型机软件应尽可能用高级语言来写，以提高应用软件研究的速度。本书对常用微型机语言BASIC、FORTRON、PL/M、PASCAL等作了介绍。我们特向读者推荐PASCAL语言，由于它逻辑结构严密，且具有丰富的数据结构，便于书写软件。可以预言，它将成为微型机今后的主要语言，应该加以推广。

现在我们简要地介绍一下本书各章的内容：

第一章是为各行各业准备使用微型机的非电子专业的技术人员介绍一些计算机的基本知识。有了这些知识，基本上就能够阅读后面的内容了。

第二章介绍8080微处理器芯片。

第三章介绍8080微处理器指令系统和程序举例。这一章的程序举例一方面使读者加深对8080的指令系统的理解，另一方面获得程序设计的初步概念。

第四章微型计算机的接口技术。这一章说明如何用微处理器（CPU），ROM，RAM及I/O接口芯片来组成微型机系统。对I/O接口芯片及接口技术将进行详细讨论。重点是8080的中断系统和DMA系统。此章虽然没有介绍任何一台现成的微型机，但它建立了组成任何微型机系统的基础。

第五章程序设计和软件。介绍了微型机的基本系统软件（汇编程序和监控程序）的设计原理和四种高级语言以及利用微型机开发系统开发软件的方法。

第六章是微处理机在电子测量和控制中的应用。这一章以三个简单的微处理机化系统为例介绍微处理机化设计方法，讨论几个微处理机化电子测量仪器例子以及几种常用算法。最后对微型机开发工具作一概括介绍。

第七章标准总线。将对微总线、S-100总线、多总线、IEEE-488标准接口总线以及EIA-RS-232C串行总线标准作较详细的介绍。

第八章Z-80微型计算机。将详细介绍Z-80CPU的指令系统、CPU硬件结构、四种接口芯片（PIO、SIO、CTC、DMA），并对Z-80微型机系统进行较详细的介绍。

第九章微型机系统。这一章以三个例子来说明微型机系统的设计方法，其目的并不在于介绍这些具体系统，而在于说明微型机系统设计所要考虑的一些问题、硬件和软件设计的相互关系以及微型机化设计的一些特点和优点。

在附录3中给出了8085微型机与8080微型机的比较。因为8085是对8080向上兼容的，在对8080系统有了一定了解之后，可以通过阅读这个附录迅速熟悉它们。

此外，4-5-2节、4-6-4节、附录2是几个专用接口芯片。不使用这些芯片的读者可以不看这部分内容。

由于微型计算机这门新兴学科正在飞速发展，新东西层出不穷，如：最近产生的新16位微型机在原理上和系统结构上均有重大变化，但考虑到本书的基础性质，这些内容只好割爱，感兴趣的读者请参看有关文献。

参加本书编写工作的有四位同志。李慧文编写了第二章、第三章、第八章（其中8-4节为王开西同志编写）及附录1，3；谈根林编写了第四章、第九章及附录2；汪庆宝编写了第五章；李礼贤编写了第一章、第六章、第七章及4-9节；最后由谈根林统稿；并聘请天津大学计算机系刘家松同志审阅。

本书是我们在1978年所编“INTEL 8080微型计算机及其在电子测量仪器中的应用”讲义的基础上编写的。该讲义是为1978年11月第四机械工业部在北京举办的微型计算机学习班而

编写的，后来曾在多个单位（如北方交通大学电信系研究生班、中国计量科学研究院、北京市纺织研究所等等）所举办的微型计算机学习班上用作教材。在教学过程中，同志们对讲义内容提出了很多宝贵意见，对于本书的编写有很大帮助，在此对这些同志表示衷心感谢。

在本书的编写过程中得到北京工业大学无线电电子学系系领导的大力支持，特别是系主任张德有同志对本书的编写工作给予了很多的鼓励和帮助；此外，本单位很多同事对本书初稿提出了宝贵意见，在此我们一并表示感谢。

由于我们在微型计算机这门新兴学科方面也是初学，在理论上和实践经验上都很欠缺，因此，书中免不了出现一些缺点和错误，恳切地希望专家和读者提出批评和意见，以便再版时进行修改。

编　者

1981.4.20 于北京

目　　录

第五章　程序设计和软件

5—1　概　　述

程序设计是包括微型计算机在内的所有数字计算机应用中最重要的环节之一。因为不论是微型计算机还是大、中、小型数字计算机，都是在事先编制好并装入计算机的存贮器内的计算机程序指挥之下运行的。一台一点程序软件都不带的计算机（又称为“裸机”），是什么工作也不会做的。通常，几台同一型号的计算机，在外部硬件环境完全一致的条件下，只由于配备的软件不同，其功能就大不相同。对于用微型计算机控制的系统或微处理器化的电子仪器，虽然所用的处理器芯片都一样（例如都采用 Intel　8080A），但由于所配置的程序软件不一样，它们就能完成不同的功能。由此可见，对从事微型计算机应用的人员来说，有关程序设计和软件问题的学习和研究，就显得格外重要。

在前面几章，特别是第三和第四章，已经结合指令系统和输入/输出接口讨论了一些程序设计问题。本章将系统地阐述程序设计的一些基本概念和方法，以及前面还没有涉及的一些具体问题。除了以汇编语言程序设计为主以外，还要讨论几种常用高级语言（如BASIC、FORTRAN、PL/M、PASCAL等）的程序设计问题。

这一章的另一部分内容是讨论有关微型计算机软件设计的一些问题。目前，各种微型计算机的有关系统，软件非常丰富。限于篇幅，不可能讨论所有各种软件的设计问题。本章将重点讨论用于8080系统的一种监控程序（MONITOR）软件和汇编程序（ASSEMBLER）软件，一方面说明它们的功能和结构，另一方面通过分析某些功能的实现方法，使读者对软件设计的一般方法有所了解。这些方法不仅适用于系统软件，对于应用软件也是很有用的。此外，本章的最后一节还要叙述微型计算机软件的开发过程，并简单介绍一些与微型计算机开发系统有关的问题。下面，首先简单讨论有关程序设计和软件的一些基本概念。

5—1—1　计算机程序和计算机语言

程序本来是人们把一些要做的事或工作，根据它们互相之间的关系，以及自然或人们的需要，按一定先后次序排列起来的计划。然后，工作就按照这个计划，有步骤有次序地进行。计算机程序本来的意义也就是这样。一台数字计算机进行工作的“能力”表现为它的各种指令的功能。我们把具有不同功能的各种计算机指令按一定算法所要求的次序排列起来，并给予指令执行中所需要的各种数据，这就形成了计算机程序，通常简称为程序。

各种计算机程序尽管最后都表现为一系列计算机机器指令码和数据的有序排列，但在计算机的应用和日常的程序软件工作中，计算机程序却往往可以采用多种不同的表现形式。通常，完成同一工作任务的计算机程序，由于其采用的计算机语言不同而表现形式也相差甚远。按照计算机程序的表现形式，可以把它们分为两大类：一类是面向计算机的机器语言程序；另一类是面向使用者的类似于英语的语言程序。后者还可分为低级语言程序和高级语言程

序。我们这里所说的语言是指计算机语言，或者更确切地说是计算机程序设计语言。

语言是人们交换信息的媒介。人们编制计算机程序指挥计算机如何运行也是一种信息的传递。因此，我们把它通过的媒介称为语言是理所当然的。

数字计算机应用的早期，程序设计采用的都是机器指令码，即机器语言。众所周知，机器指令码都是一串一串的“0”和“1”的二进制数，尽管它很方便地为计算机所接受，或被计算机“理解”，但人们辨认和记忆起来却十分不便。或者说，它的“可读性”很低。虽然有时这种二进制的“0”、“1”码也可以转换成其它八进制、十进制、十六进制的数码，但本质上还是一样，特别是对于记忆，十分困难。这样，计算机程序设计就成了一门相当难学的专门学问，从而给计算机应用的推广带来极大的困难。

最先出现的面向使用者的计算机语言是汇编语言，即用一些英语单词的缩写形式作为计算机指令的助记符号，如用ADD代表“加”，用SUB代表“减”等。这种用助记符号编写的计算机程序称为符号汇编，或称汇编语言程序。但这种程序计算机是不“懂”的，因而它无法接受并遵照执行，只有把它变成用机器言表示的目标程序（或目的程序），才能由计算机执行完成。和目标程序相对应的汇编语言程序称为源程序，源程序所用的语言称为源语言。把汇编语言程序转换为机器语言的目标程序这一过程称为汇编。汇编过程虽然也可以用人工进行，但现在几乎都毫无例外地由计算机程序完成。实现汇编过程的计算机程序称为汇编程序或汇编软件。

从第三章的讨论已知，尽管现代的汇编语言中增加了一些伪指令和宏指令，增强了汇编语言的功能，但汇编语言程序和目标程序基本上还是一一对应的。就这一特点来说，可以从两个不同的侧面加以考虑。一方面由于和目标程序的固定关系而使程序设计工作十分繁琐，但另一方面却可以使程序工作者非常细致灵活地设计程序，从而使程序结构、执行时间等都达到最经济合理，并且最充分地发挥计算机的能力。这一重大优点是汇编语言程序设计至今仍为人们十分重视的主要原因，尤其是对于微型计算机。

汇编语言是一种最低级的计算机程序设计语言，也是包括微型计算机在内的各种计算机必备的基本程序设计语言。但由于汇编语言距离人们习惯上常用的思维形式较远，而且设计一个较好的程序常常需要很多经验和技巧，工作量大、周期长。对于某些应用，例如大量的科学计算和日常的应用，非常不便，对于不同机种，互相间很难共享彼此的程序资源等。从五十年代起便开始了对高级计算机语言的研制，至今已获得非常巨大的成果，成为计算机科学中的一个重要领域。

高级语言的特点是其表现形式比较接近人们在科技工作中常用的一些语言形式。由于计算机存储容量和处理能力的限制，各种高级语言都有其各自主要适用的范围●，其语言的形式和语法结构都有严格的定义，不允许有任何多义性的情况发生。多年以来，在科技界流行最广的是FORTRAN、ALGOL、BASIC等；而在商业、管理等数据处理上用得最广的是COBOL，最近，又出现综合了多种语言优点的PASCAL等。这些高级计算机语言的一个重大特点是学习掌握非常方便，初步学会一种语言往往只要两周至一个月左右的时间。在掌握了一种语言以后，学习其它语言则更快。当然，要熟练地运用一种语言编制程序，特别是编制出高质量的计算机程序，还是要作比较大的努力的。

高级语言的计算机程序，当然不能直接由计算机接受，必须经过加工处理，“翻译”成

● 因为各种高级语言都有其适用范围，故又称为面向过程的计算机语言。

用机器语言表示的目标程序，才能为计算机接受并执行。这种将高级源语言表示的计算机源程序“翻译”为用机器语言表示的目标程序的过程，通常称为编译。它由一个称为编译程序（Compiler）的软件来完成。很明显，在某一种计算机上是否能用某种语言设计程序，就取决于该计算机是否配备了相应的编译软件。目前的微型计算机也大都配有一些不同的编译软件，以便于机器的推广应用。通常，完整的 Intel 8080 系统除了配备有能处理伪指令和宏指令的宏汇编软件（macro assembler）以外，还可配备BASIC、FORTRAN IV、PL/M、COBOL以及PASCAL等语言的编译软件。这给微型计算机应用的工作带来极大的方便。

5-1-2 计算机程序和计算机软件

为了用计算机解决各种各样的问题，人们设计编制了数量无法统计的各种计算机程序。有一类程序，是人们为了解决某些临时性的特殊问题而设计编制的。这类程序在使用一次或若干次之后，便完成了它的历史使命，而被废弃。还有一类计算机程序，由于各种原因而使它具有长期或经常使用的价值，以致成为计算机工作必不可少的部分，就如同计算机的机器部件一样。这就是计算机软件，或称软设备。“设备”一词是指它具有长期的使用价值，“软”设备是相对于由电子线路和机械结构组成的“硬”设备而言的。称为计算机软件的程序，有的是为了进一步改进发挥计算机的效能，帮助人们合理而快速地管理计算机的运行，有的是为了方便程序的编制，有的是为了帮助人们检查计算机的故障，也有的是为了计算机长期、经常执行某些工作等。当然，计算机软件和一般的计算机程序之间并没有严格的界限，也有人把计算机程序都叫软件。

计算机软件通常可分为两大类。一类是通用性较强，或者对于某类机型基本上是常备的软件，它们通常由计算机制造厂家提供。这类软件称为系统软件，如前面提到的处理计算机程序设计语言的汇编软件、编译软件等，还有管理计算机运行的操作系统、检查测试计算机的故障诊断程序等都是。另一类软件是为特殊的专门用途而研制设计的，它的专用性很强。这一类软件一般都由应用单位根据其具体情况要求而研制设计，故称专用软件或应用软件。由于计算机应用范围的广度和深度都在日益发展，特别是在微型计算机大量问世以后，更加速了这个过程。因此，对计算机应用软件研制的需要量也与日俱增。应用软件的研制是软件工作的一大组成部分，也是计算机应用特别是微型计算机应用工作中最重要、工作量最大的一部分。

现代计算机科学技术的发展表明，计算机已日益形成为一种由软件和硬件有机地、不可分割地组成的系统。如何合理地分配、协调系统的软件与硬件两部分的分工和连接，是计算机系统和计算机应用系统设计的主要关键问题。

5-1-3 程序设计过程和流程图

流程图或流图，也称为框图，是一种表示事件进程的数学图形方法，也是程序设计中用以表示程序进程的有力工具。按照图形理论来说，流程图是一种有方向的图，它用带有箭头的线段表示程序进程的方向，用一些约定的符号表示程序的开始、结束和各种处理过程。此外，它还用一定的符号表示过程中的条件逻辑判断所引起的程序进程的分支情况。这样，就使程序的进展过程在图上一目了然、清楚地呈现出来了。因此，流程图已成为程序设计尤其是复杂程序设计中不可缺少的工具，也是程序软件文字资料中不可缺少的重要内容。

图5-1-1可概括地说明流程图的性质和作用。这个流程图说明程序要完成的工作可以分为几个步骤，而且根据情况的不同，进程也会有差别。从图5-1-1可见，程序中可能进行五种计算处理，但是就每次具体的执行过程来说，却并不是每种计算处理都要执行。根据分析分支条件的结果可以发现，实际上存在四种可能的进程顺序，这就是：a.计算处理1、2、4；b.计算处理1、2、5；c.计算处理1、3、4；d.计算处理1、3、5。进程的不同在于判断条件1和条件2是否满足而产生。由条件1的判断决定第二步是执行处理2还是执行处理3，而由条件2的判断决定第三步是执行处理4还是执行处理5。通常，条件的产生有两种来源：一种直接来自前面的计算处理结果，例如计算结果是否为零，或是否为负等；另一种是检测输入端口的信息（包括中断请求），根据输入端口信息的性质特征决定程序的进程是做哪一种操作。后面例8判读键盘输入字符以决定程序分支就是一个典型的例子。值得注意的是，在图5-1-1这个例子中，每种进程都执行三种处理，实际上则多半是不定的，满足某一种条件时步骤可能少一些，而满足另一种条件时可能步骤要多一些。这必须看具体要求而定。

目前，流程图中所采用的符号花样很多，为了简单起见，本书不拟采用过于繁杂的形式，而采用图5-1-2所示的几种最常用的符号。根据这些符号在流程图中的作用，可以注意到它们在流程图中的特征。开始符是程序的入口，它只有引出线而没有引入线，而且引出线只有一条；结束符是程序的出口，显然，它只有引入线而没有引出线，而且引入线可以不止一条。接续符和这两种类似，它是为了简化图形用的。在其它几种符号中，除了判断分支符号应该有两条引出线外，其余都只能有一条引出线，但它们的引入线则都是不限制的。注意以上这几个特征可以帮助我们正确画出流程图或检查一个流程图是否合理。

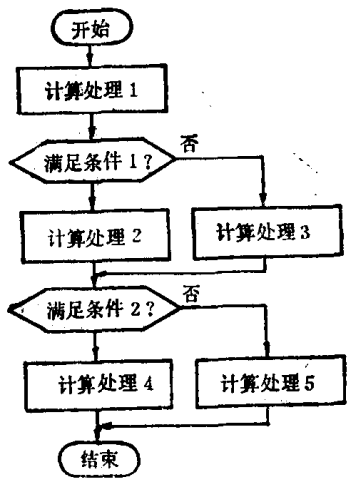


图 5-1-1　程序流程图

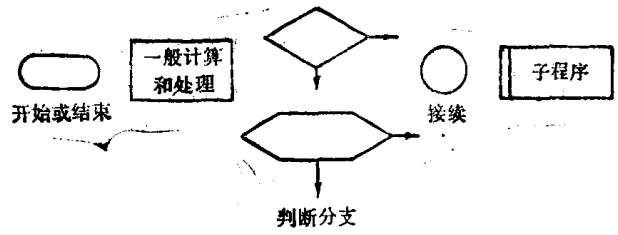


图 5-1-2　流程图符号

众所周知，任何一个计算机程序总是为一定的目标服务的，例如完成一个数学计算，或一个数据处理，或控制一个过程等。这就是说，程序设计的目的是要使计算机按规定的方法和步骤，对代表不同信息的数据进行运算处理，然后输出结果，或根据这些结果控制某种过程。归纳一般的程序设计过程，大致可用图5-1-3所示的流程图表示。以下将图5-1-3中所示的各步骤分别加以说明。

定义程序需要解决的问题是程序设计开始必须完成的任务。通常定义一个程序所要解决的问题应该包括：给程序的输入、处理运算的详细内容、程序的输出。

数学模型是程序设计的重要环节，特别是对于大型程序，数学模型的好坏对于程序设计的质量具有决定性的作用。所谓数学模型，就是把人们要求计算机完成的工作表述为一个数

学过程。例如要用计算机程序模拟某种物理运动过程，我们就应该取得描述这种过程的数学方程式，然后通过解这个数学方程式完成这一模拟过程。当前，对各种数学模型的研究，已形成各门学科中的重要分支，有的甚至成为独立的学科体系。

在数学模型建立以后，往往还要研究确定解决这些数学问题所用的计算方法。因为同样的数学问题常常可以用不同的方法求解。不仅复杂的科学计算中的数学问题可以采用不同的计算方法解决，对于一些简单的问题，例如比较两个数据是否相等，也可以采用减、补码加的算术运算或“异或”逻辑运算等不同的方法。计算方法的好坏常常影响所得结果的精确性、程序的质量、计算机利用的效率等等。有时，还可能影响结果的正确性。因为大多数计算方法都有其适用条件，如果使用条件不适当，完全可能产生错误的结果，至少也会影响结果的质量和计算机的工作效率。因此，必须针对所需解决的问题，研制或选定所用的计算方法。

前已说过，流程图是程序设计中的有力工具。根据确定的数学模型和计算方法，便可画出程序的流程图。程序设计中的流程图，繁简程度不一，有的很简单，只是大致说明程序工作过程；有的很详细，几乎和具体的程序差不多。我们也可以把上述过程结合流程图的设计加以说明。这对于某些数值计算较少的小型程序设计可能更为适用。

最简单的一种流程图称为概念级流程图。它是结合问题的定义，把程序所要完成的主要工作内容，有次序地表述出来。图5-1-4就是一个微型计算机监控程序的简单概念级流程图。这个图概略地说明了该监控程序执行操作的过程。它说明，在程序启动完成了系统各种条件的准备工作（总名为“初始化”）以后，首先在控制台键盘CRT终端显示器或电传打字机上，显示出一个约定的，表示程序已准备好等待键盘输入命令的符号，这个符号称为提示符。各种系统的提示符可由系统设计者设定，通常用的有星号（*）、尖角号（>）、箭头（—>）、横线（—）等。然后扫描键盘，等待操作人员从键盘打入的命令。在接到命令后，先判断命令是否有错误，如果有错误就指示出错，重新等待输入；如果没有错误，就分析这个命令，根据分析结果辨认命令的要求，再去执行要求的操作；最后，操作完成后再显示一

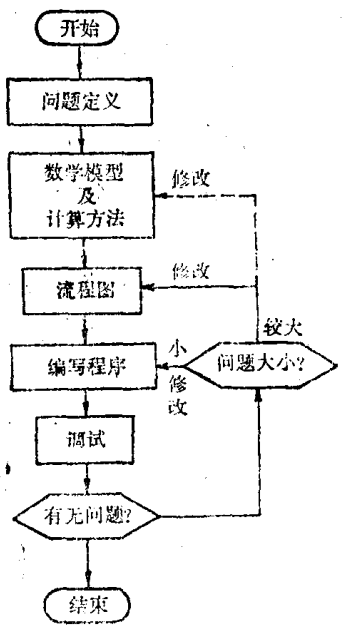


图 5-1-3 程序设计过程

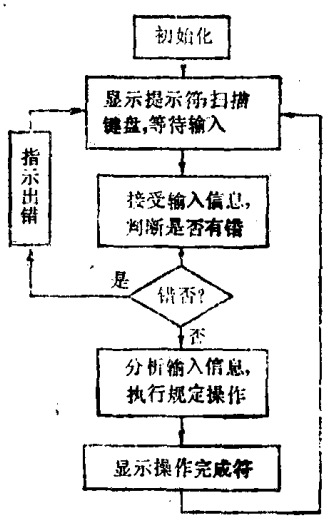


图 5-1-4 概念级流程图

定的符号，转回至程序开始，继续进行类似的操作。关于这个程序的主要细节，我们还要在5-4节中讨论。当然，有的概念级流程图也可以比这个图详细一些。

概念级流程图的进一步是算法级流程图，它大体上和图5-1-3中的数学模型及计算方法阶段相对应。也可以说，算法级流程图是概念级流程图的展开，它表示达到所需要的结果的详细步骤。所谓算法，不一定是指我们通常所说的数值计算方法。粗略地说，它泛指完成一个给定任务的计算机操作过程❶。这种过程可以包括计算方法技巧，可以包括对一些信息的处理变换，也可以包括一些为某些特殊目的所要求的操作。完成同一个任务可以采用不同的算法。这里，我们将图5-1-4所示概念级流程图的中间部分，展开成算法级流程图如图5-1-5所示。

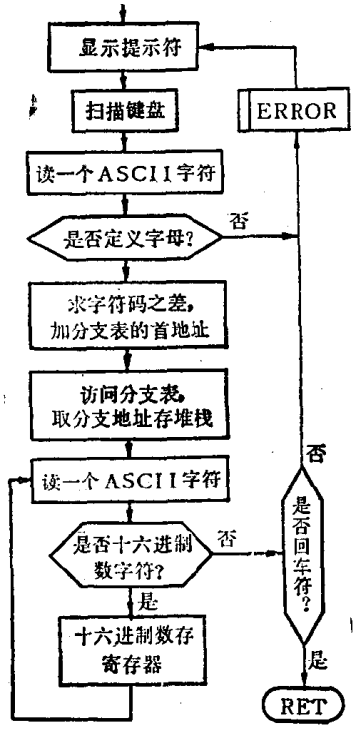


图 5-1-5 算法级流程图举例

从图5-1-5可见，在显示提示等待符号后，程序即扫描键盘，准备接受输入。对于接受到的单字母字符，可根据定义比较判断，是否为合格的字母符号。由于合格的字母符号在ASCII字符的编码中是连续的，故可求合格的字符码之差而得出不同字符之间的数码关系。这个关系使我们可以利用相应的分支地址表指示要求分支的相应地址❷。因此，按这个关系访问分支地址表，将分支地址寄存在堆栈顶上，以备用RET指令分支转移。然后，再次接受键盘输入。由于按该监控程序的定义，在规定的字母命令后面是几位十六进制的操作数。因此，需要判断输入的数字字符，并用移位的方式形成所要求的十六进制数据，存在工作寄存器中。回车符是该程序规定的命令输入结束符，因此，在接收到回车符以后，程序就用RET指令分支转移到堆栈顶内容所指示的分支地址去，执行需要的操作。

在这个图中，几处不合格的判断都转向子程序ERROR，然后回到等待状态。ERROR是显示出错信息的子程序。

从这个流程图我们可以比较详细地了解该程序采用什么样的操作步骤完成给定的任务。如用字符码之差访问分支地址表，把分支地址放在堆栈中用RET指令分支转移，以及其它一些操作处理步骤。

算法级流程图再进一步展开就形成语句级或指令级流程图。这种流程图可以说是最详细的程序流程图。通常，这种流程图的一个方框只相应于一、二个程序语句或二、三条指令，有时甚至一个方框相应于一条指令。图5-1-3的程序设计过程中的流程图阶段，通常相应于算法级和语句级之间的流程图。

综上所述并结合图5-1-3明显可见，在程序流程图完成以后，尤其是如果作出了语句级

---

❶ 算法（algorithm）和计算方法（method of computation）的意义是不一样的。关于算法的严格定义，请参见D.E.Knuth，《The Art of Computer Programming》vol. 1，1973。中译本《计算机程序设计技巧》第一卷，管纪文、苏世霖译，国防工业出版社1980年版。

❷ 关于分支地址表法，请参见例5-2-8。

或指令级的流程图，将它转换成计算机程序就是非常容易的事。因此，如果图 5-1-3 中所示的从开始到流程图阶段之间的各步骤都完成得很好，那么编写程序就成为非常简单的工作了。

至此，虽然计算机程序都已编写出来，但整个程序设计工作还未完成。实践证明，不论经验多么丰富的程序工作者，编写程序时总会不可避免地发生这样或那样的错误。因此，任何程序在编写出来以后都要经过查错调试，证明没有错误以后，才能使用。在调试过程中可以查出各种不同的问题或错误，经过适当的修改，纠正其错误或改善其性能，一直达到正常运行，整个程序设计才算完成。对于较大型的程序，有的问题可能在调试时都未能发现，这就必须在使用中进一步解决或改进。所以，有时为了慎重起见，调整好的程序还要试运行一段时间，再交付使用。这样，程序的可靠性当然就更高些。

关于微型计算机程序的调试，特别是许多应用软件程序的调试，还有一些具体问题，将在5-6节中讨论，这里就不再详述了。

最后需要指出，上述程序设计步骤只是一般的大致过程。每个程序工作者在自己的工作中还可以结合实际的具体情况进行自己的程序设计工作。关于大型程序软件的设计开发工作，经过多年的发展，目前已形成一些较为系统的方法。这需要在经过一些具体问题的讨论之后，才便于说明。这些内容我们都在5-6节中讨论。

5-2 汇编语言程序设计

汇编语言的程序设计，在目前的微型计算机的程序软件工作中应用最广。上一节中已经讨论了程序设计的一般问题。在这一节中，我们将通过一些常用的具体例子，说明 Intel 8080 微型计算机的汇编语言程序设计。这些例子有的是数据处理中常用的，有一些是在各种系统软件或应用软件中常用的。虽然它们都是一些单个的小程序，但却是某些大型程序软件中的基本部分。因此，熟悉它们是有实用价值的。

5-2-1 简单数据处理程序

数据处理是计算机应用中经常碰到的问题。除了加、减、乘、除等算术运算外，常用的处理还有分选排队、求最大值或最小值、成组传送、查表、求累加和、求平均值等等。前面第三章中已经讨论过的简单算术运算，这里不再重复，以下主要介绍几种常用的处理方法和程序。

1. 求最大值、最小值

在一大批数据中寻找其中的最大值或最小值，或者最大值和最小值同时都要找到，这是统计数据处理中经常碰到的问题。我们来看如何用计算机完成这一工作。首先研究如何求最大值（求最小值可以采用类似的方法）。假定这一批数据一共有N个，连续存放在存储器中首地址为DATA的数据区内。为了要找到这些数据中的最大值，我们采用对数据逐个进行比较的方法。按照这种方法，首先把第一个数取到累加器，假定它是第一次找到的最大值。接着，把累加器中这个假设的临时最大值依次和后面的数进行比较，只要发现后面的数比累加器中的这个数大，就把较大的数放进累加器中，代替原来的最大值，而作为新发现的最大值，然后继续和后面的数进行比较。象这样一个个地比较下去，显然，最后累加器必然就是这N个数据中的最大值。而对N个数据来说，这一比较检查过程一共要重复

有时，除了要求找出最大值或最小值外，还需要知道这个数在数据区中的相对位置。这也可以由计算机程序很容易地完成。由上述讨论可知，为了完成N-1次比较操作，可以采用循环式程序，并用循环计数器控制循环的结束。很容易看出，在计算机执行比较操作时，循环的次数和被比较的数据在数据区中相对位置的关系非常简单，即第 i 次循环是把累加器中的数和第 i + 1 个数相比。在比较中，如果发现这第 i + 1 个数比累加器中的数大，就把这个数作为新发现的最大值，把它记在累加器中代替原来的最大值。我们假定最后找到的最大值就是这第 i + 1 个数据，那么，只要在把这个数存入累加器的同时，记下循环次数 i，经过简单的换算，就可以得到这个数据在数据区中的相对位置。

在具体程序中，因为循环计数器通常都是设其初值为需要循环的次数，然后在循环中每次减 1，最后由判断计数器之值是否为零来控制循环结束。因此，上述关系并不是 i 和 i + 1 的形式。具体来说，这里循环计数器的初值应该是 N－1，即第一次循环中第一个数和第二个数相比较时，计数器之值为 N－1；而第 i 次循环中和第 i+1 个数比较时，计数器之值为 N－i。按照上述假定，如果这第 i+1 个数为最大值，就应该记下计数器的 N－i 这个值，然后经过计算转换，得到 i+1 这个结果。这个计算可用以下公式表示：

$$N+1-(N-i)=i+1$$

它在其它类似情况下也能应用。这就是说，用 N+1 减去记下的计数器之值 N－i，就可以得到这个结果。

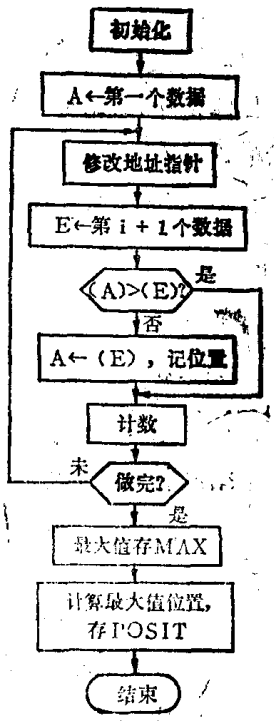


图 5-2-1 求最大值流程图

把上述方法的过程画成流程图如图 5-2-1 所示。以下例题就是实现这一过程的一个具体程序。

例 1 设有10个数据（正数）存在首地址为DATA的数据区内。要求设计一个程序，从这10个数据中找出其中的最大值，并指出该最大值在数据区中的相对位置。

解 按照上述原理和流程图，为了要对数据区内的数据进行依次的比较，首先，我们设HL寄存器对为数据区地址指针，以便于访问数据区。其次，设B寄存器为循环计数器，并置其初值为 N－1＝9。为了在比较过程中记下存入累加器的较大值的循环次数，用C寄存器保存这种情况下的循环计数器之值。值得注意的是，如果第一个数是最大值，则上述操作将始终不会发生。因此，必须给C寄存器赋一个初值，使得在这种情况下能得到 i+1＝1 的结果。很明显，由前述公式，应给C寄存器赋以N（这里是10）的初值。对于其后的具体操作来说，以上这些内容就是程序的初始化部分。

为了便于比较起见，我们除了用累加器存放已取得的当前最大值外，还用E寄存器存放被比较的第 i+1 个数，以使比较过程明确清楚。此外，最后找到的最大值将存放在工作单元MAX中，最大值在数据区中的相对位置则存放在POSIT单元中。

这一程序如下：

```
LXI   H, DATA  ;设数据区地址指针的初值
```