微型计算机及其应用丛书

微型计算机汇编语言的使用与分析

孔庆时　万加雷　编著

WEIXINGJISUANJIWEIXINGJISUANJI
WEIXINGJISUANJIWEIXINGJISUANJI

科学出版社

微型计算机及其应用丛书

微型计算机汇编语言的使用与分析

孔庆时　万加雷　编著

科学出版社

1988

内 容 简 介

本书引用最新技术资料，主要介绍 Z80，MC68000，8086（8088），Z8000，MC6800 等几种常用的 8 位和 16 位微处理器的指令系统，微型计算机汇编语言的结构、工作原理、汇编语言程序设计方法，编辑、汇编、连接、调试程序的应用．并结合作者近年来对微型计算机所作的开发工作，介绍了一些分析、开发系统软件的软件工具，以及一些系统软件的分析和应用．书中给出了大量的应用实例和完整的程序清单，可供编制汇编程序时直接引用或作参考．书后还附有各种微处理器的指令表，以方便读者检索和使用．本书可作为微型计算机培训班的教材或大专院校的教学参考书，也可作为工具书来使用．

微型计算机及其应用丛书

微型计算机汇编语言的使用与分析

孔庆时　万加雷 编著

责任编辑　李淑兰　孙月湘

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行　各地新华书店经售

*

1988 年 3 月第 一 版　开本：787×1092　1/16

1988 年 3 月第一次印刷　印张：20 3/4

印数：0001—9,500　字数：471,000

ISBN 7-03-000036-6/TP・4

定价：5.30 元

序　　言

一九七一年微处理器问世，短短十三年时间，微型计算机经历了难以预测的技术变革，获得了突飞猛进的发展．在技术性能方面，微型计算机比第一台电子管计算机提高了四至五个数量级；微型计算机的字长由4位、8位、16位发展到32位，覆盖了小型计算机和超级小型计算机的全部领域．多微型机分布结构的系统已经商品化．微型计算机网络技术已日趋成熟，亦有商品问世．多微型机的阵列机或多功能的微型计算机系统已向中、大型计算机系统发起了挑战．另一方面，价廉物美的个人微型计算机日新月异，如雨后春笋般蓬勃发展．总之，微型计算机冲击了计算机科学技术的全部领域，大大扩展了计算机的应用范围，为计算机渗入个人的工作、生活、家庭以及渗入社会的各个方面创造了必要的条件．这样就有力地促进了新的产业革命，加速了信息技术的发展，推动了信息社会向纵深发展．

一九七九年全国计算机委员会确定要大力发展微型计算机，推广微型计算机的应用并且在全国组织试点．四年来，微型计算机的应用扩展了几十倍、上百倍，应用领域遍及工业、农业、科学文化、军事国防等各个方面；应用计算机的人才成百倍地增长，取得了空前丰硕的成果．当前，迎接新的世界技术革命的浪潮澎湃向前，一个突出表现就是席卷全国的“微型计算机热”．毫无疑问，这个热潮必将推动四个现代化的事业向前迅进．

顺应形势，我们编写了这套“微型计算机及其应用丛书”．本丛书贯彻理论和实际相结合的原则，在介绍基本理论的同时引人许多实例．丛书的著者都是该领域的专门家，他们吸收消化了国外的经验，分析了国外的技术和系统，结合我国的情况进行开发、创造，取得了许多可喜的成绩．因此，他们写出的东西是学了即可用得上的．

本丛书并不打算解决所有层次的问题，只想在普及推广方面提供一套基本的系统的材料，使有志于微型计算机应用的人们能有一套参考书．这套丛书不仅包括基本的微型计算机硬件系统、软件系统，还包括了微型计算机的开发技术和实际应用等内容．这些内容中许多都是著者实际工作成果的总结，因此它有一个鲜明的特点：解决实际问题，与四化建设紧密相连．

目前国内外有许多关于微型计算机的著作和资料，不过象本丛书这样全面地讲述还很少见．我们希望这套丛书能为四化建设作出贡献．

丛书书目

微型计算机硬件系统(上、下)

微型计算机操作系统(上、下)

微型计算机汇编语言的使用与分析

微型计算机 BASIC 语言的应用与分析

微型计算机开发系统

微型计算机的系统设计及性能评价

微型计算机在企业管理中的应用
微型计算机在数控技术中的应用
微型计算机在图象信息处理中的应用

徐正春　张菊年

前　言

自六十年代后期微处理器问世以来已获得了迅速的发展，在短短的十几年里，由4位8位、16位一直发展到32位．由于以8位和16位微处理器为核心组成的微型计算机系统的性能越来越强，价格不断降低，在国外已普及到家庭之中，在我国也得到广泛的应用．它在信息革命中正起着越来越大的作用．

汇编语言(机器指令)作为软件和硬件之间的接口，是编制操作系统、高级语言及一切应用程序的基础．每一种微型计算机系统都配有一套相应的汇编软件，它是必备的基础软件之一．

本书引用最新技术资料，并结合作者近年来对微型计算机所作的开发工作而写成．它着重介绍国内流行的几种8位和16位微型计算机的各种指令系统，汇编程序的结构和工作原理．针对汇编程序比较难写的特点，还以较大的篇幅详细介绍了基于汇编语言的程序设计方法，并结合一个具体的机型介绍了与汇编有关的软件——编辑、汇编、连接，调试程序的用法，以及分析、开发汇编程序的软件工具．本书还给出大量的应用实例和从系统程序中整理出来的程序清单，目的在于更好地帮助读者掌握汇编语言的使用．

陈俊强同志仔细审阅了全部稿件，提出了许多宝贵意见，作者在此表示诚挚的谢意．

由于作者水平有限，不当之处请批评指正．

孔庆时　万加雷

1985.8

目　　录

第一章 概 论

1.1 什么是汇编语言

微处理器一般每次只能执行一项操作，如送数、加法、减法、移位、转移等，都比较简单．要实现较为复杂的功能，可以将许多操作组合在一起完成．指令是一个二进制位组合或二进制数，在取指周期时，它出现在微处理器的数据输入端，能使微处理器完成一个特定的操作．例如 00111101 这条指令能使 Z80 微处理器将其累加器 A 的内容减 1．这个二进制数称为指令代码．每条指令都有一个特定的代码与之对应，为了书写和阅读的方便，人们常将指令代码写成十六进制或八进制的形式．指令一般由操作码和操作数两部分组成．操作码指出进行何种操作．操作数指出哪个数参加操作，它可以是参加操作的数本身，也可以是存放该数的内存地址或寄存器名字．计算机的全部指令的集合构成了它的指令系统．第二章以 Z80 和 MC68000 作为 8 位和 16 位微处理器的代表，比较详细地介绍它们的指令系统，对其它微处理器的指令只作简单介绍．

实现特定功能的一个指令序列称为程序．由代码组成的程序称为机器语言程序．程序必须存放在内存（随机存贮器（RAM）或只读存贮器（ROM））中才能被执行．执行的顺序如下：中央处理器（CPU）按指令计数器（PC）所指示的内存地址取出指令，放在指令寄存器中，对此指令代码进行译码，控制计算机各部分执行该指令规定的操作，然后修改指令计数器，这条指令就执行完了．执行下一条指令时，CPU 按指令计数器所指示的新地址，从内存中取出新的指令继续执行．

下面是一段用 Z80 机器语言编写的延时程序，这段程序很短，只有三条指令：

地 址 （用十六进制表示）	指 令 （用二进制表示）	代 码 （用十六进制表示）
0 1 0 0	0 0 1 1 1 1 0 0	3E
0 1 0 1	0 1 1 1 1 1 1 1	7F
0 1 0 2	0 0 1 1 1 1 0 1	3D
0 1 0 3	1 1 0 0 0 0 1 0	C2
0 1 0 4	0 0 0 0 0 0 1 0	0 2
0 1 0 5	0 0 0 0 0 0 0 1	0 1

其功能如下：

第一条指令：把 127（十六进制为 7FH）送入 A 寄存器（简称 A）．它由两个字节组成，前一个字节是操作码，其意义是“送数至A”．后一个字节是操作数，即数值 127 本身，亦即 7FH．这两个字节放在内存单元 0100 和 0101 中．

第二条指令：A 的内容减 1．这条指令只占一个字节，放在内存单元 0102中．

第三条指令：检查A的内容，若不为零则跳转至 0102 单元，执行“A 减 1”的指令，

如此反复循环，达到延时的目的，直至A减为零，这段程序结束．这条指令由三个字节组成：第一个字节为操作码，其意义是“非零则跳转”．后两个字节为操作数，在这里的意义是跳转地址 0102．

这段程序很短，但要看懂也不太容易，因为它是用代码写成的，很难辨认和记忆，也很容易写错，写错了也不容易发现．这种用机器语言写成的程序，调试和修改都非常困难．为了解决这些问题，人们采用汇编语言代替机器语言．

首先，用某些符号，例如用表示操作的英文缩写代替指令代码，如用 LD A(LoaD A，向A送数的缩写)代替 00111110，用 DEC A(DEC rement A，A 的内容减 1) 代替 00111101，用 JP NZ (JumP if Not Zero，非零跳转)代替 11000010，指令的意思就容易记住．这些符号称为指令符号或助记符．

指令符号的采用给程序设计带来了很大的方便，但用绝对地址来编写程序仍感非常不便．如上例中，若要在第一条指令后再加一条指令，将A寄存器的内容暂存到B寄存器中(LDB，A)，则下一条指令 DEC A 必须后推至 0103 单元．于是为了使 JP NZ 指令仍然跳转执行 DEC A，跳转地址就必须从 0102 改为 0103．这样，在一个庞大的程序中每插入或删去一条指令，都必须修改一连串的地址，使程序员难以应付．如果用一组字符(称为标号)代替指令的地址，例如用 LOOP 代表指令 DEC A 的地址，跳转地址也相应地用 LOOP 表示（跳转指令变为 JP NZ,LOOP)，那么无论 DEC A 的地址(即 LOOP 的值) 如何变化，跳转地址都会自动地与它保持一致．程序员不必再为上述问题操心了．由此可以看出采用标号的必要性．

上面用机器语言编写的延时程序，可用指令符号和标号改写成汇编语言程序：

```
START: LD A, 7FH
LOOP:  DEC A
       JP NZ, LOOP
       END START
```

这样，各条指令的意思就一目了然，整个程序也容易看懂了．

但这样写成的程序，计算机是无法直接执行的，必须先翻译成机器语言．这个翻译过程称为“汇编”．用汇编语言写成的程序称为“源程序”．汇编后产生的机器代码程序称为“目标程序”．汇编过程可以由人(程序员)手工完成，但是做起来既繁琐单调，又容易出错；而计算机完全能胜任此工作，所以现在汇编工作绝大部分都由计算机的汇编程序来完成．

程序一般由两部分组成，一个是指令区，用来安放指令；另一个是数据区，用来安放所需的数据．为了告诉汇编程序在内存什么位置安放什么数据，留多少内存单元作临时存贮区，程序从内存什么位置开始安放，汇编到哪里结束等等，程序中需要有另一套无需 CPU 执行的指令，这种指令称为伪指令．

指令符号、标号、伪指令及它们的使用规则(语法)构成了汇编语言．汇编语言的特点是，它的指令与机器指令一一对应．因此，即使完成一个简单的工作，也得需要大量的指令，故给使用带来不便．为了节省源程序编写和输入的工作量，人们采用一组字符(称为宏指令)来代替反复出现的一组指令，让计算机在汇编时再把这些宏指令置换成它所代表的那一种指令，而且可以引进参数．具有这种宏功能的汇编语言称为宏汇编语言．

1.2 汇编语言与高级语言

汇编语言(以及相应的汇编程序)的出现,使程序设计工作前进了一大步,与机器语言相比,它的特点是明显的. 现在每种计算机都有它自己的汇编语言.

但是,用汇编语言不便于描述我们要求计算机完成的任务. 例如,让计算机打印一行字符,就要使用许多条指令. 使用汇编语言编程序还要求对计算机的结构, 例如寄存器、内存等有具体的了解,而且编出来的程序在其它型号的计算机上不能运行,因为不同型号的计算机,其汇编语言是不同的, 于是出现了 FORTRAN, COBOL 等高级语言. 这些高级语言是面向用户的,更接近于人们使用的自然语言和数学语言,因此易学易用. 这些高级语言的一个语句相当于一大批机器语言指令, 因此编写程序的速度比用汇编语言快许多倍. 编出的程序简短明瞭, 便于调试和修改. 高级语言的出现使程序设计又前进了一大步.

可是高级语言一般都有各自的特长和适用范围. 例如, 用 FORTRAN 语言处理数学计算,用 COBOL 语言进行事务处理是非常方便的, 但用它们编制过程、仪表、通信等控制程序时,程序冗长,执行时速度往往达不到要求,不如用汇编语言方便自然,可以达到最高执行速度. 对微型计算机的应用来说,往往希望尽量缩小内存,以降低成本. 而高级语言的编译程序(把高级语言翻译成机器语言或汇编语言的程序)没有相当大的内存就根本无法运行. 总之,与高级语言相比,汇编语言具有下列优点:

(1) 节省内存和 CPU 资源;

(2) 执行速度快;

(3) 能直接调动计算机的全部资源,能准确地掌握执行时间, 故适用于实时控制.

所以, 尽管出现了高级语言, 汇编语言仍不失为微型计算机的一种主要程序设计语言, 目前微型计算机的大部分系统程序都是用汇编语言编制的. 汇编语言主要适用于编制:

(1) 短的和中长程序;

(2) 要求占用内存有限的程序;

(3) 实时控制程序;

(4) 频繁使用的程序;

(5) 主要功能不是计算或数据处理,而是输入/输出的程序.

1.3 几种汇编方法

汇编产生的目标程序, 有的可以直接运行, 有的还需经过连接和装入后才能直接运行. 这决定于汇编程序的类型. 主要有下列几种情况:

1. 汇编成绝对地址并立即装入内存执行

这种汇编程序将源程序按绝对地址格式产生目标程序,并将其装入内存,汇编完成后开始执行该目标程序,如图 1-1 所示. 其优点是: 实现起来简单,也容易理解. 缺点是:

(1) 缩小了用户区,因为运行时汇编程序亦留在内存中; (2) 一次只能汇编一个模块,一个程序不能分为几个模块,分别由几个人同时编制和调试; (3) 程序每执行一次,就要汇编一次,浪费时间.

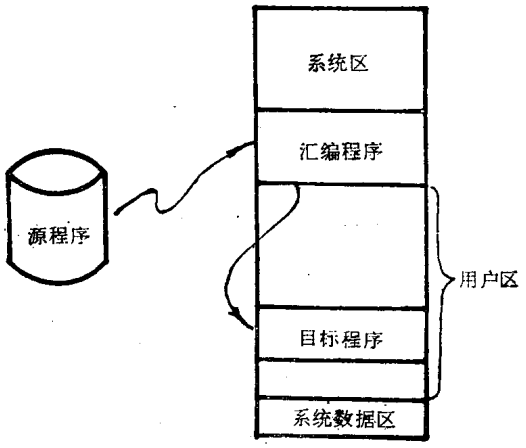

图 1-1 第一种汇编方式框图

2. 汇编成绝对地址格式, 然后由装入程序装入并运行

这种汇编程序先将源程序的不同模块分别汇编成各自的绝对地址目标程序,然后用装入程序装入内存相应地址,并执行,如图 1-2 所示. 其优点是: (1) 装入程序简单,占用内存少,这是因为汇编程序已不在内存,故用户区较大; (2) 可连接不同的几个模块, 包括用不同语言写成的模块. 缺点是: 要由用户管理内存,稍不注意,不是使两模块发生冲突,就是在它们之间留下太大的间隙,浪费内存.

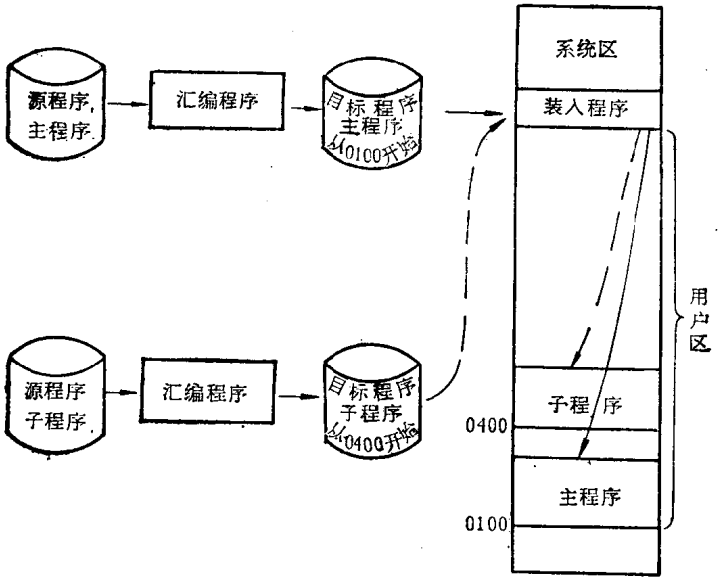

图 1-2 第二种汇编方式框图

3. 直接连接和装入

汇编程序将各个模块汇编成相对地址的浮动目标程序. 各模块的地址均从零开始,然后由连接和装入程序将这些模块连接成一个完整的绝对地址的可运行目标程序, 需要时调入内存运行,如图 1-3 所示. 其优点是: (1) 可以连接几个模块,甚至是用不同语言写成的模块; (2) 连接后的目标程序很紧凑,没有间隙,节省内存; (3)对于程序员来说,主程序和子程序的连接简单, 主程序用伪指令 EXT 说明有哪些是外部子程序之后, 即可调用它们,其绝对地址由连接装入程序解决. 缺点是: (1) 连接程序较复杂; (2) 子程序连接模块全部连接上去,不管主程序是否用到它们,因而整个程序占用内存偏大.

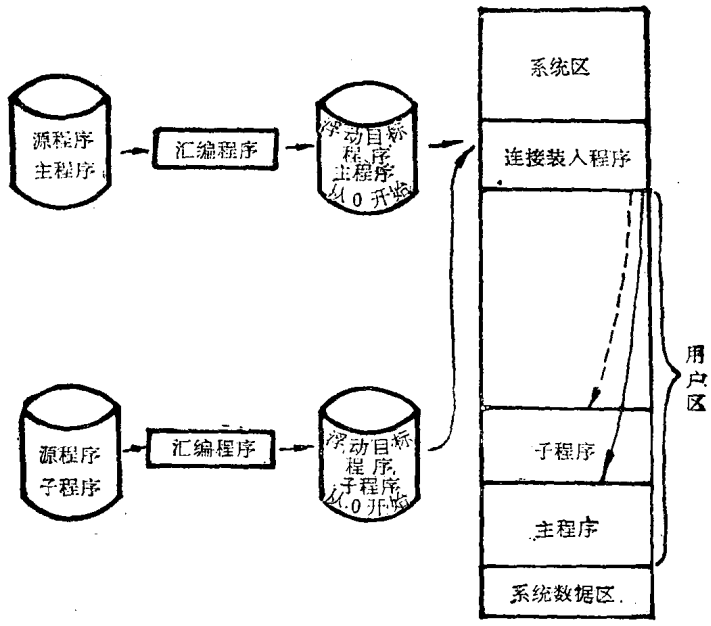

图 1-3　第三种汇编方式框图

4. 动态连接和装入

这是一种最好的方法. 汇编程序仍产生浮动目标程序，但一个模块只在被调用时才装入内存并进行连接，如图 1-4 所示. 其优点是：(1) 节省内存，同一内存区域在不同时刻可装入不同的模块，因此可以运行超过实有内存的大程序(分块)，或在有限内存空间内

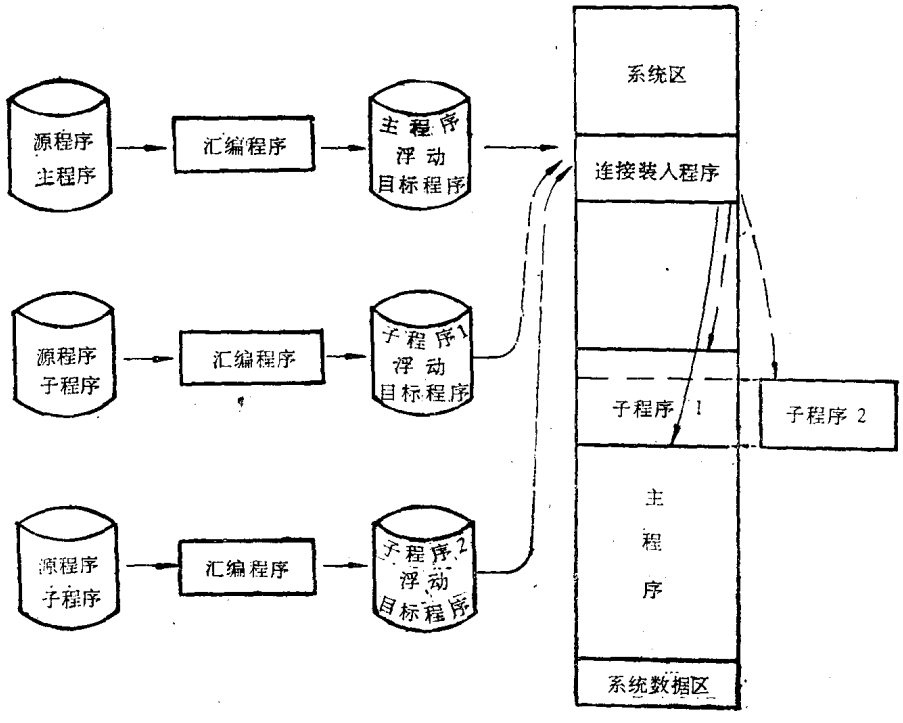

图 1-4　第四种汇编方式框图