

微型计算机丛书

〈 软件 〉



编 写 说 明

十年来,微型计算机发展迅速,已成了计算机发展的重要方向之一。微型机以其物美价廉、小巧玲珑而深深地渗入各个应用领域:无论工业、农业、商业、军事还是科学、文化、教育都有微型机的用武之地。

在我国,微型机的应用发展也很快。社会主义现代化事业要求更快地发展微型机,更广泛地应用微型机。为了适应这种形势的需要,国家科委计算机委员会办公室组织了王德霖、何文兴、陈俊强、张菊年、孙辑正、万家镭、孔庆时、朱巧生、阮祖望、杨宗霖、戴兆康、阴亚伟、陈平、潘 蓁、童志通、傅世海、储留大等同志编写了这部微型计算机丛书,以供国内培训干部或自学之用。

因编写时间仓促,水平有限,谬误之处请读者批评指正。

沈芾生、孟永炎、潘君卓等同志对本书作了审阅,在此谨致谢意。

1981年8月于北京

内 容 简 介

这本丛书取材于国内外有关资料，比较详细地介绍了微型计算机软、硬件系统。内容包括宏汇编、BASIC、微型机操作系统和微型计算机系统四大部分。宏汇编部分有宏汇编程序原理和程序编制法、编辑程序的输入和修改以及调试程序的原理和使用；BASIC部分介绍了扩展BASIC、编译BASIC和控制用BASIC；微型机操作系统部分主要叙述了微型机操作系统以及CDOS等操作系统实例；微型计算机系统部分包括典型的8位、16位微处理器、微型机结构系统、微型机的拓展与开发系统、微型机外部设备以及微型机在各领域中的应用。

本书用例丰富，实用与理论并重，讲述由浅入深，具体易懂，是目前国内微型机资料中较新、较全面、较系统的一种参考书。

本书适于作微型机培训教材或自学入门书。

目 录

宏 汇 编

第一章 概论	3
1.1 什么是汇编语言	3
1.2 汇编的几种方法	5
1.3 程序从编写到执行的过程	6
1.4 例	7
1.5 汇编语言程序的格式	10
第二章 Z80的指令系统	14
2.1 微型机系统的结构框图	14
2.2 Z80 CPU的结构	14
2.3 Z80指令代码及其长度	16
2.4 寻址方式	22
2.5 标志	28
2.6 八位传送类指令	32
2.7 十六位传送类指令	32
2.8 交换、数据块传送和查找类指令	35
2.9 八位算术运算和逻辑运算类指令	37
2.10 十六位算术运算类指令	38
2.11 通用算术和CPU控制类指令	39
2.12 循环和移位类指令	40
2.13 位置1, 置0 和测试类指令	43
2.14 转移, 转子和返回指令	44
2.15 输入、输出类指令	46
第三章 伪指令	48
3.1 什么是伪指令	48
3.2 名字说明	48
3.3 数据说明	49
3.4 外部标号说明(EXT)和入口标号说明(ENTRY)	50
3.5 汇编结束说明(END)	51
3.6 例NQUEEN	51
3.7 代码段说明(REL, ABS, DATA, COM)和汇编 起始地址说明(ORG)	53
3.8 程序名说明(NAME)	55
3.9 插入文件说明(*INCLUDE)	55
3.10 条件汇编(IF, ENDIF)	56
3.11 宏定义和宏调用(MACRO, MEND)	57

3.12 对打印文本的说明	57
3.13 小结	58
第四章 宏汇编	60
4.1 参数在宏中的应用	60
4.2 宏定义的嵌套和宏调用的嵌套	61
4.3 宏定义中的标号	63
4.4 宏指令对Z80指令的扩充	64
4.5 宏指令的特点和它与子程序的区别	64
第五章 程序设计方法	66
5.1 什么是程序	66
5.2 简单程序	66
5.3 框图法	66
5.4 分支程序	67
5.5 循环程序	68
5.6 堆栈和子程序	75
5.7 查表方法	80
5.8 链接方法的应用	84
5.9 输入和输出	87
5.10 中断处理	89
第六章 宏汇编程序	92
6.1 引言	92
6.2 汇编过程	92
*6.3 宏处理过程	96
*6.4 宏处理与汇编相结合	100
第七章 源程序的输入——编辑程序	102
7.1 基本概念	102
7.2 基本的编辑命令	104
7.3 控制字符	107
7.4 编辑命令概述	107
7.5 移动字符指针命令 $\pm nL$, $\pm nJ$, $\pm nC$, $\pm B$	108
7.6 移动字符指针并打印命令 $\pm n$, $\pm nP$, $\langle CR \rangle$	108
7.7 打印命令 $\pm nT$	109
7.8 删除命令 $\pm nK$, $\pm nX$, $\pm nD$	109
7.9 查找、替换字符串命令 $\pm nF$, $\pm nS$	110
7.10 暂存缓冲区命令 nY , nG	111
7.11 磁盘读写命令 R , W	112
7.12 重新启动编辑程序命令 H , O	113
7.13 输入文件命令 nA , nN	114
7.14 结束和取消编辑程序命令 E , Q	115
7.15 其他命令 $\pm U$, V , Z	115

7.16 宏命令或命令迭代	116
7.17 条件编辑命令	116
7.18 暂存和执行命令串命令±M	117
7.19 编辑程序信息汇总表	118
第八章 汇编程序的调用	119
8.1 调用格式	119
8.2 汇编时的选择项	119
8.3 打印文本的格式	121
8.4 错误信息	126
第九章 目标程序的连接和装入	129
9.1 命令格式	129
9.2 连接开关	129
9.3 例	131
9.4 连接和装入过程概述	132
*9.5 REL 文件的格式	133
9.6 连接程序时的错误信息	136
第十章 目标程序的调试	137
10.1 引言	137
10.2 汇编和反汇编命令	138
10.3 断点、启动和跟踪命令	140
10.4 显示和修改内存与寄存器内容的命令	142
10.5 移动和检查内存命令	144
10.6 输入输出命令	146
10.7 与磁盘有关的命令	146
10.8 十六进制运算和十进制转换命令H	147
10.9 写入PROM命令P	147
10.10 用@命令加速调试工作	147
10.11 例	148
第十一章 汇编程序的子程序库	153
11.1 引言	153
11.2 系统调用子程序	153
11.3 ASMLIB	154
11.4 DEMOLIB	157
11.5 例	158
第十二章 附录	160
12.1 为PROM编制程序的步骤	160
12.2 8080到Z80的翻译程序TRANSLAT	160
12.3 建立FORTRAN新逻辑部件编号表(LUN)的方法	161
12.4 ASMLIB 打印文本	162
12.5 DEMOLIB 打印文本	174

BASIC

第一章BASIC 的立即执行方式	185
1.1 预备知识	185
1.2 算术运算符及常数类型	185
1.3 变量及算术表达式	186
1.4 内部函数	187
第二章basicBAS IC	189
2.1 预备知识	189
2.2 BAS I C 的程序执行方式	190
2.3 赋值语句	191
2.4 逻辑运算及关系表达式	191
2.5 控制转移语句	193
2.6 启动执行及停止语句	197
第三章 输入、输出语句	200
3.1 输入、输出语句	200
3.2 输出格式编辑语句	203
第四章 BASIC 程序的编写和调试	208
4.1 初始化语句	208
4.2 BAS I C程序编写举例	210
4.3 BAS I C程序的调试及有关指令	216
4.4 条件转移指令(续)	221
4.5 其它一些与调式有关的指令	223
第五章 字符串处理	224
5.1 字符串及字符串变量	224
5.2 子字符串说明格式	226
5.3 字符串的运算和比较	230
5.4 用于处理字符串的几个函数	232
第六章 数据文件输入输出	237
6.1 预备知识	237
6.2 文件的建立和打开	238
6.3 数据文件的读写	240
6.4 数据文件的关闭	249
6.5 一个简单的例	250
第七章 与机器直接有关的指令	257
7.1 与机器直接有关的指令	257
7.2 与输入输出及内存有关指令	262
7.3 BAS I C解释程序的一些有关信息	263
第八章 BASIC 软件设计举例	270
8.1 问题总述	270

8.2 命令处理	272
8.3 X(Y)轴处理及两点连线处理	274
8.4 四种图形的处理	275
8.5 实用程序	276
第九章 数据文件管理系统 (ZDFMS)	281
9.1 系统引导程序(ZDFMS. ZHU)	281
9.2 准备程序(PREPARE. ZHU)	282
9.3 主控程序(MAIN. ZHU)	283
9.4 建立主文件(及其分类索引文件)功能模块(功能1, ZDFMS1. ZHU)	284
9.5 单记录查阅、删除、修改功能模块(功能2, ZDFMS2. ZHU)	285
9.6 增加(少量)新记录功能模块(功能3, ZDFM3. ZHU)	285
9.7 统计功能模块(功能4, ZDFMS4. ZHU)	286
9.8 分类功能模块(功能5, ZDFMS5. ZHU)	287
9.9 求某项平均值模块(功能6, ZDFMS6. ZHU)	287
9.10 阅读功能模块(功能7, ZDFMS7. ZHU)	288
9.11 保存和复制功能模块(功能8, ZDFMS. ZHU)	288
9.12 刷新功能模块(功能9, ZDFMS9. ZHU)	289
9.13 选择修改功能模块—兼管理用流水帐(功能10, ZDFMS10. ZHU)	289
9.14 计算功能模块(功能11, ZDFMS11. ZHU)	290
9.15 更改记录构成(增加项目)功能模块(功能12, ZDFMS12. ZHU)	291
9.16 查看系统情况功能模块(功能13, ZDFMS13. ZHU)	291
9.17 固定修改功能模块(功能14, ZDFMS13. ZHU)	292
9.18 修改全部记录功能模块(功能15, ZDFMS15. ZHU)	292
9.19 连接功能模块(功能16, ZDFMS16, ZDFMS16. ZHU)	293
附录: 变量在内存里的存储形式	294

操 作 系 统

第一篇 微型计算机操作系统原理

第一章 绪论	301
1.1 操作系统的任务	301
1.2 微型机操作系统的发展	302
第二章 存储管理	305
2.1 单一连续分配	305
2.2 分区分配	305
2.3 选体分配	306
2.4 可重定位分区分配	306
2.5 分页分配	307
2.6 请求分页分配	308

2.7 分段分配	309
2.8 段页分配	309
第三章 CPU 管理	310
3.1 进程	310
3.2 作业调度	311
3.3 进程调度	312
3.4 交通控制	313
第四章 设备管理	314
4.1 外部设备分类	314
4.2 设备管理技术	314
4.3 逻辑设备	315
第五章 信息管理	317
5.1 地址换算 (结构映象)	317
5.2 文件结构	320
5.3 文件目录	321
5.4 盘区管理	322
5.5 文件的保护、保密与共享	323
5.6 文件操作	324
5.7 文件系统结构	324

第二篇 CDOS 的使用

第六章 CDOS 的命令	326
6.1 命令、文件标记的格式与分类	326
6.2 目录命令	331
6.3 打印与保存命令	334
6.4 磁盘初始化及系统状态命令	336
6.5 复制命令	339
6.6 批命令	342
6.7 RAM 测试命令	345
6.8 系统启动	345
第七章 系统调用	347
7.1 概述	347
7.2 有关输入输出设备的系统调用	349
7.3 有关磁盘的系统调用	352
7.4 有关文件管理的系统调用	354
7.5 其他系统调用	361
7.6 文件FCB1, FCB2 和命令行缓存区	364
7.7 系统调用小结	367
第八章 系统生成	369
8.1 CDOS 的生成	369

8.2 打印CDOS	371
------------------	-----

第三篇 CDOS 分析

第九章 CDOS 中的数据结构	377
9.1 概述	377
9.2 指令	377
9.3 字符串	378
9.4 堆栈	378
9.5 表	380
9.6 表树及查表方法	381
9.7 缓存区	382
第九章 附录	382
第十章 设备驱动	386
10.1 概述	386
10.2 输入输出设备驱动程序	387
10.3 软磁盘驱动程序	392
10.4 硬磁盘驱动程序	398
第十章 附录	403
第十一章 文件管理	406
11.1 概述	406
11.2 文件传送	406
11.3 文件控制块F C B	409
11.4 目录管理	412
11.5 盘束管理	413
11.6 文件操作	415
第十二章 命令处理、CDOS 引导与初始化	419
12.1 命令处理	419
12.2 内部命令	420
12.3 外部命令	424
12.4 CDOS 引导	425
12.5 CDOS 的初始化	426
第十三章 CDOS 结构与总框图	428
13.1 CDOS 结构	428
13.2 CDOS 总框图	429

第四篇 其他微型机操作系统

第十四章 RDOS	431
14.1 概述	431
14.2 引导程序	432
14.3 监控程序	433

14.4 监控命令	434
14.5 RDOS 3 与 RDOS 1	439
第十五章 CP/M 与 MP/M	441
15.1 CP/M	441
15.2 MP/M	441
15.3 在 CP/M, CDOS, MP/M 三个操作系统下软件的兼容性	444
第十六章 几种微型机的操作系统	446
16.1 TRSDOS	446
16.2 Z80 RIO 与 ZMOS	448
16.3 RT-11 与 TSX	451

微 型 计 算 机 丛 书

宏 汇 编

万加镭 孔庆时 编著

第一章 概论

1.1 什么是汇编语言

微型计算机一般每次只执行一项操作，如送数、加法、减法、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来完成。指令规定了计算机能完成的操作。它分成两个部分：操作码和操作数。操作码指出进行何种操作。操作数指出哪个数参与操作，它可以是参加操作的数本身或该数存放的内存地址及寄存器名字。每一指令都有一个特定的代码与之对应，称之为指令代码，它一般用二进制数来表示，为阅读和书写方便常写成十六进制或八进制的形式。某一计算机所有指令的集合构成该机的“指令系统”。Z80微处理器(CPU)的指令系统在第二章详细介绍。实现特定功能的一组指令，称之为机器语言编号的“程序”。第五章将要详细介绍如何利用Z80的指令系统来编制程序。

程序通常必须先放入内存(RAM或ROM)中，才能被执行。执行的顺序如下：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，控制计算机的各个部分执行该指令规定的操作，然后修改指令计数器，这条指令就算执行完了。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：

1. 把127(十六进制为7F)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，也即7F。这条两字

地 址	指令代码	指令代码
(用十六进制表示)	(用二进制表示)	(用十六进制表示)
01 00	00 11 11 10	3 E
01 01	01 11 11 11	7 F
01 02	00 11 11 01	3 D
01 03	11 00 00 10	C 2
01 04	00 00 00 10	02
01 05	00 00 00 01	01

节长的指令放在内存单元0100和0101中。

2. A的内容减1。这条指令只占一个字节，放在内存单元0102中。

3. 检查A的内容，若不为0则跳转至0102单元，执行“A减1”的指令，如此反复循环，达到拖延时间的目的，直到A减为0，程序执行结束。这条指令由3个字节组成。第一个字节为操作码，其意义是“非0则跳移”，后两个字节为操作数，在这里的意义是跳转地址为0102。注意，地址的数值在指令代码中和在内存存放时，是将低8位放在前，高8位放在后。在这条指令中地址0102的低位02放在0104单元内，而高位01放在0105单元内，与一般书写习惯相反。

这段程序很短，但要看懂也不大容易，这是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。这种用机器语言来写的程序，调试和修改都非常困难。为了帮助记忆和使用机器语言指令，采用了汇编语言来代替机器语言。

首先用某些符号，例如操作码的英文缩写来代表指令的意义，会给使用带来很大方便。如用LD A(LoaD A，向A送数的缩写)代替00 11 11 10，用DEC A(DECrement A，A的内容减1)代替00 11 11 01，用JP NZ(JumP if Not Zero，非0跳转)代替11 00 00 10就容易记住它们的意思。

使用绝对地址来编写程序仍使程序设计非

常困难。如上例中在第一条指令后再想加一条指令,将A寄存器的内容暂存到B寄存器中去(LD B, A), 下一条指令DEC A则后推到0103单元中,而JP NZ的指令后推到0104, 0105和0106单元中。于是为了使跳转指令仍跳转到执行DEC A指令,跳转地址就必须由0102改为0103。因此在一个大的程序中,每插入或删除一条指令,则一连串的地址都要修改。如果用一组字符(称为标号)代替某条指令的地址,例如用LOOP代表指令DEC A的地址,跳转地址也相应用LOOP来表示(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,这样程序员不必再为上述问题担心,可见用标号代替绝对地址很有好处。标号不仅可以代表地址,还可以用来代表其它量(如常数等)。

上面那段用机器语言写的延时程序可用指令符号和标号改写成汇编语言程序如下:

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

这样,各条指令的意义就一目了然,整个程序也容易看懂。

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”。汇编过程可由人(程序员)手工来完成,但目前绝大部分已由计算机系统配置的软件(汇编程序)来完成。

一个程序一般由两个部分组成,一个是指令区用来安放指令串,另一个是数据区用来安放所需的数据。对数据区的说明要用到机器指令之外,由汇编语言提供的新指令,这种指令称为伪指令,另外要说明该段程序从那儿开始执行,汇编到那儿结束,也要用伪指令加以说明。上例中END一行表示汇编到此结束,其后的标号START表示将来程序从标号START

那一行开始执行。伪指令在第三章介绍。

汇编语言和机器语言的指令基本上是一一对应的。因此,即使完成一件不大的工作,也需要大量的指令,使用仍不便,不易掌握。为了节省源程序编写和输入的工作量,采用一组字符(宏名字)来代替多次重复出现的一组指令,让计算机在汇编时再把这些宏名字重新置换成它所代表的那组指令,而且可以引进参数,这样“宏”就类似于高级语言中的自定义函数或过程。宏在第四章详细介绍。

指令符号、标号、伪指令和宏及其它们的使用规则(语法)便构成了汇编语言。

汇编语言(以及相应的汇编程序)的出现,使程序设计工作大大前进了一步,与机器语言相比,它的优点是非常明显的。现在几乎每一种计算机都有自己的汇编语言。汇编语言是目前程序设计,特别是微型机程序设计使用的主要语言之一。许多系统软件都是用它写成的。

汇编语言还是低级语言中的一种,虽然宏汇编语言增加了不少功能,但它仍然是面向机器为主的语言。为了方便用户使用微型机,最近在许多微型机上都纷纷配上高级语言,像BASIC, FORTRAN, COBOL, PASCAL等。这些高级语言是面向用户的,更接近于大多数人习惯使用的语言或数学语言,因此容易学习和运用。它们的一个语句就相当于一大批机器语言或汇编语言的指令,编出来的程序更加简单明了,便于调试、修改。高级语言的出现使程序设计又前进了一大步。

虽然出现了高级语言,但汇编语言仍有它的优点,故仍不失为主要程序设计手段之一。

与高级语言相比,汇编语言的优点是:

1. 节省内存和CPU资源;
2. 执行速度快;
3. 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。

从表1-1可以看出汇编语言在计算机使用的各种语言中的位置。

本编着重介绍CROMEMCO Z80宏汇编语言及相应的宏汇编程序、连接装入程序调试

表 1-1 汇编语言、机器语言和高级语言的关系

语 言	例	子
高 级 语 言	PL/I	DO I = 1 TO 20 WHILE XSMALL; J(I)=INDEX(A, B); XSMALL=INDEX(A, C); END;
	COBOL	MULTIPLY RATE BY HOVRS GIVING TOTAL
	FORTRAN	WRITE (3,11) 11 FORMAT (IX, 'K=')
	BASLC	10 A=127 15 A=A-1 20 IF A<>0 THEN GOTO 15 25 STOP
低 级 语 言	汇编语言	START: LD A, 7FH LOOP: DEC A JD NZ, LOOP END START
	记忆码机器语言	0103 LD A, 7F 0105 DEC A 0106 JP NZ, 0105 0109 JP 0000
	用十六进制表示的二进制机器语言	0103 3E7F 0105 3D 0106 C2 0501 0109 C3 0000

程序、子程序库以及输入源程序用的编辑程序。为了帮助读者更好地了解汇编过程，我们加上一些简单的原理说明。带*的章节是为愿意更多了解汇编原程的读者而写的，对初学者可先略去不看。

1.2 汇编的几种方法

汇编好的目标程序有的可以直接运行，有的不能直接运行，还需经过连接和装入之后才能直接运行，这主要决定于汇编程序的种类。主要有以下几种类型：

1. 汇编成绝对地址并能立即执行。这种汇编程序将源程序按绝对地址格式产生目标程序，并将它放在内存中，在汇编完成后，开始运行该目标程序。

其优点是：实现起来简单，也容易理解。缺点是：a. 缩小了用户区，因为在运行时汇编程序也留在内存中；b. 一次只能汇编一个模

块，而不能汇编几个模块；c. 每次执行程序之前都要汇编一次，浪费时间。

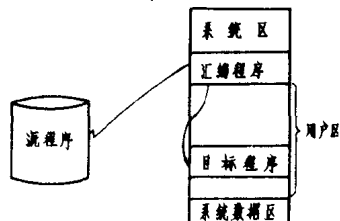


图 1-1 第一种汇编方式框图

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，然后用装入程序将它们装入内存相应地址，并执行该目标程序。其优点：a. 装入程序很简单，占的地方也小；b. 可连接几个不同的模块，包括用不同语言写成的模块；

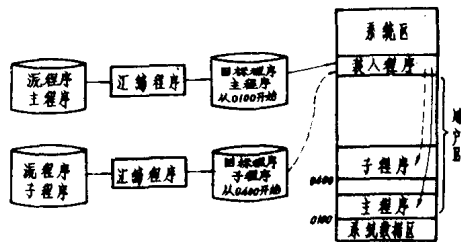


图 1-2 第二种汇编方式框图

缺点：a. 要由用户来管理内存，一不小心可能发生冲突，或者每个模块之间空隙太大，浪费内存；

b. 每次执行时都要规定好连接和装入那几个模块，操作费事。

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，每一模块的地址均从 0 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目标程序，这个目标程序才是可以运行的。

其优点：a. 可以连接几个模块，甚至是用不同语言写成的模块；b. 连接后的目标程序很紧凑，节省了内存；c. 主程序和子程序的连接简单，主程序用 EXT 说明有那些是外部子程序之后即可调用它，其绝对地址由连接

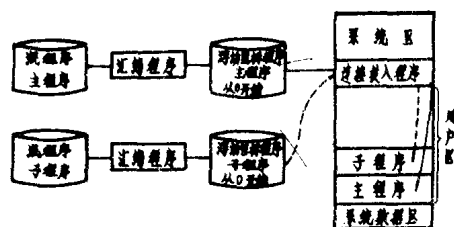


图 1-3 第三种汇编方式框图

程序解决。

缺点：a. 连接程序较复杂；b. 子程序模块全部连接上去，不管主程序是否用到某个子程序，因而整个程序模块偏大。

4. 动态连接和装入。这是最好的一种方法。汇编程序仍产生浮动目标程序，而且只有当一个模块在被调用时才连接和装入该模块。

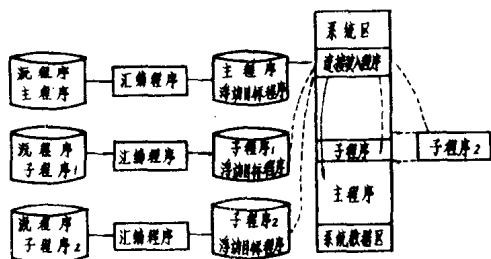


图 1-4 第四种汇编方式框图

其优点：a. 非常节省内存，子程序模块可重迭装入内存；b. 同样可连接几个用不同语言写成的模块；c. 对多用户系统一个模块可为几个用户共享。

缺点：a. 非常复杂的连接和装入程序；b. 当一个模块几次被调用时，需要调用连接和装入程序几次，效率降低，用特别的硬件支持可改进处理速度。

本机是采用第三种方式进行汇编和连接装入的。先通过二次扫描的汇编程序将主程序和子程序的源程序模块分别汇编成浮动目标程序模块放在磁盘中，然后按主程序、子程序的顺序连接起来并装入内存，形成完整的绝对地址目标程序，才能运行该程序，或将该目标程序存入磁盘，在需要时再调入内存运行。若主程序调用了系统提供的或用户早先编好的子程序

库中的子程序，则必须把这个子程序库模块连接到主程序上去。

1.3 程序从编写到执行的过程

为了产生能够由机器执行的绝对地址目标程序，必须经过一系列的过程，见图1-5。

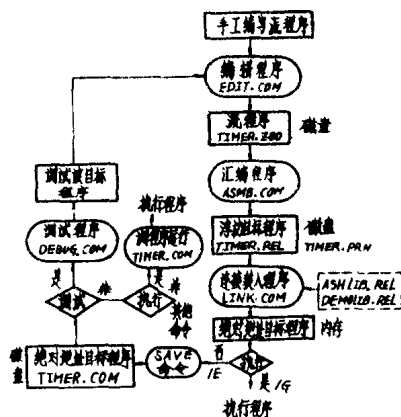


图 1-5 程序设计过程

首先，必须通过控制台的键盘把用汇编语言编好的程序输入计算机，存到磁盘上，成为一个源程序文件。其文件类型应为 Z 80，供后来的汇编程序加工用。这个工作由系统提供的软件 EDIT. COM来完成。源程序的修改，像删去、增加或调换部分源程序的内容，也由编辑程序 EDIT. COM来完成。编辑程序的详细说明见第七章。

当用编辑程序把源程序输入计算机，并修改好之后，就可以用汇编程序 ASMB. COM把源程序翻译成浮动目标程序，其文件类型为 REL。为了便于查出程序中的错误和进行调试及修改工作，汇编过程中还可以根据命令产生一个打印文本，其文件类型为 PRN。它给出汇编后的浮动目标程序代码信息和与之对应的源程序，出错信息及一些附表来反映标号的位置以及用到那些操作码等(详见第八章)。为了方便，一般较大的程序可写成几个模块，每一个模块单独进行编写、汇编和调试，最后用连接和装入程序 LINK. COM将这些浮动目标