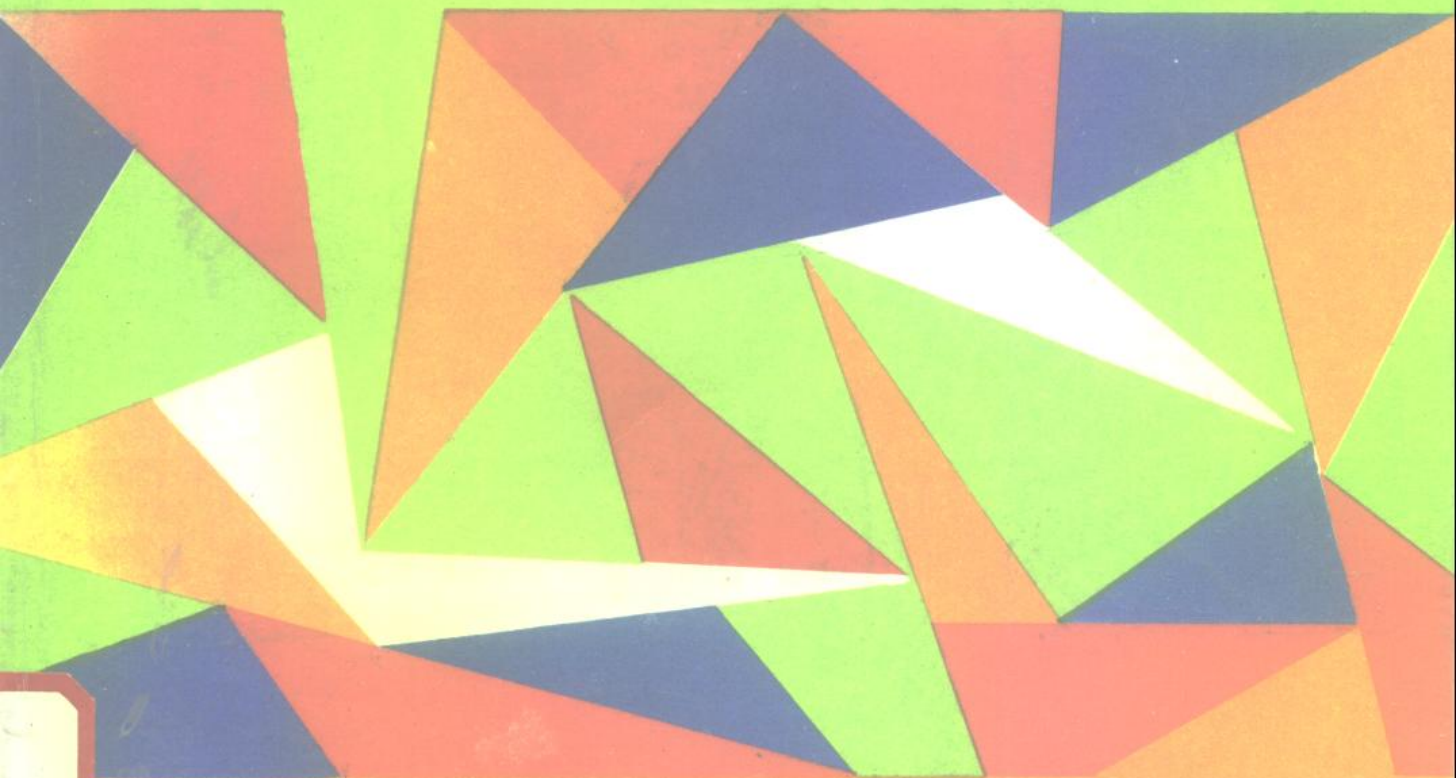
微机高级语言
与
汇编语言
接口技术和实例

天方图书创作室　李振格　编著



北京航空航天大学出版社

微机高级语言与汇编语言

接口技术和实例

天方图书创作室　　李振格　　编 著

北京航空航天大学出版社

(京)新登字166号

内 容 提 要

接口就是搭起一座资源共享的桥梁。Turbo C、Turbo Pascal、Turbo Prolog 等高级语言与汇编语言接口能直接使用 BIOS、DOS 的功能，直接对串行口、视频、游戏棒与鼠标等硬件进行存取，能进行内存驻留程序、设备驱动程序编程，提高了处理的速度(特别是图形处理速度)，扩展了控制。高级语言之间的接口能充分发挥各种语言的独特优势，使编程更具灵活性；高级语言与数据库管理语言的接口能减少繁琐的数据项管理(数据项插入、删除、检索、排序)；高级语言与 BIOS 和 DOS 的接口扩展了高级语言的低级功能，减少使用汇编带来的繁琐的负担。

本书针对以上问题，介绍了混合编程的基础、技巧、调试以及工程管理等，适合于编程人员使用。

书　　名：微机高级语言与汇编语言接口技术和实例
WEIJI GAOJI YUYAN YU HUIBIAN JISHU HE SHILI
作　　者：天方图书创作室　　李振格　编著
责任编辑：许传安
出　　版：北京航空航天大学出版社(北京市学院路37号，100083)
发　　行：新华书店总店科技发行所
销　　售：本社及各地新华书店
排　　版：天方科技公司
印　　制：北京朝阳区科普印制厂
开　　本：789×1092　1/16
印　　张：22
字　　数：563千字
印制日期：1994年5月第1次印刷
印　　数：8000册
书　　号：ISBN 7-81012-491-9/TP·120
定　　价：18.00元

前　言

Borland 公司推出众多风靡世界的 Turbo 系列语言 Turbo Pascal、Turbo C、Turbo Prolog、Turbo Basic、Turbo C++等。这些年轻的语言以其优良的性能和用户界面，很快获得世界各地用户的欢迎。Borland 率先使用的集成环境、联机帮助与热键驱动已成为当今软件用户界面的标准。独有的内部调试器使编程和调试效率倍增，开发周期骤缩。此外，Borland 辅助开发实用程序 Turbo Debugger、Turbo Profiler、MAKE、TLINK、TLIB、TCREF、GREP、TOUCH、OBJXREF 和 TC(P)HELP 等，更使 Turbo 系列语言大放光彩，形成了编程(集成环境)、调试(内部源级调试器和 Turbo Debugger)、运行(集成环境中模拟)、剖视(Turbo Profiler)和工程管理(MAKE、Project 性能)一体化的良好环境。

Turbo 系列语言都有其解决问题的方向，如果综合各自的优点，进行混合编程，那么编程水平就将更上一层楼。

现在世界软件市场异彩纷呈，Microsoft 公司开发的语言 Microsoft C、QuickC、Microsoft Pascal、Quick Pascal、Microsoft Fortran 和 Microsoft Basic、Quick Basic 也很受用户欢迎。如果能跨越 Borland 和 Microsoft 公司之间的障碍，把两者的优点结合在一起编程，那将无往而不胜。

Turbo C、Turbo Pascal、Turbo Prolog 等高级语言与汇编语言接口能直接使用 BIOS、DOS 的功能，直接对串行口、视频、游戏棒与鼠标等硬件进行存取，能进行内存驻留程序、设备驱动程序编程，提高了处理速度(特别是图形处理速度)，扩展了控制功能。

高级语言之间的接口能充分发挥各种语言的独特优势，使编程更具灵活性；高级语言与数据库管理语言的接口能减少繁琐的数据项管理(数据项插入、删除、检索、排序)；高级语言与 BIOS 和 DOS 的接口扩展了高级语言的低级功能，减少使用汇编带来的繁琐负担。

顾名思义，接口就是联络，搭起一座进行资源共享的桥梁。

接口的要素在于参数传递协议、函数返回协议、寄存器协议、数据的内部格式的差异、不同编译器生成的 OBJ 的兼容性(特别是相应的汇编语言的兼容性)、生成的 OBJ 文件所用的汇编级的选项的差别、启动代码的主次、重复的屏幕输入/输出和文件操作等相同功能的扬弃和内存管理的处理等。

本书针对以上难题进行探讨，介绍了混合编程的基础、混合编程的技巧、混合编程的调试、混合编程的工程管理等。虽然如此，但是接口技术属于高级编程的领域，加上此专题的资料不多，因此本书只是结合长期实践经验的一次写作尝试，如有不当，请读者不吝指教。

编　者

1994 年 2 月

目　录

第0章　概　述 …… 1

0.1　适合高级程序设计语言调用的汇编语言子程序的编写格式 …… 2
0.1.1　建立过程 …… 3
0.1.2　进入过程，建立一个参数表的固定基点 …… 3
0.1.3　分配局部数据空间 …… 3
0.1.4　保存调用代码的寄存器值 …… 4
0.1.5　存取参数，编写对参数的具体处理过程 …… 5
0.1.6　送返回值 …… 5
0.1.7　恢复寄存器，退出过程返回调用程序 …… 5
0.2　各高级程序设计语言调用汇编语言子程序的具体约定 …… 5
0.3　对于 Fortran 和 Pascal 的长返回值问题 …… 5

第一部分　Turbo C 与其它语言的接口

第一章　Turbo C 与汇编语言的接口 …… 8

1.1　在 Turbo C 中使用嵌入式汇编 …… 8
1.1.1　嵌入式汇编如何工作 …… 10
1.1.1.1　Turbo C 如何知道使用嵌入式汇编模式 …… 13
1.1.1.2　激活 Turbo Assembler 处理嵌入式汇编 …… 14
1.1.1.3　Turbo C 在何处汇编嵌入式汇编码 …… 14
1.1.1.4　将-1 开关用于 80186/80286 指令 …… 15
1.1.2　嵌入式汇编语句的格式 …… 16
1.1.2.1　嵌入式汇编中的分号 …… 16
1.1.2.2　嵌入式汇编中的注解 …… 16
1.1.2.3　访问结构/联合的元素 …… 17
1.1.3　嵌入式汇编示例 …… 18
1.1.4　嵌入式汇编的限制 …… 22
1.1.4.1　内存和地址操作数限制 …… 22
1.1.4.2　嵌入式汇编中缺少隐含的自动变量大小 …… 23
1.1.4.3　必须保存寄存器 …… 24
1.1.5　嵌入式汇编码相对于纯 C 代码的缺点 …… 25
1.1.5.1　降低了可移植性和可维护性 …… 25
1.1.5.2　降低了编译速度 …… 25
1.1.5.3　仅可由 TCC 使用 …… 25
1.1.5.4　损失了优化能力 …… 25
1.1.5.5　限制了对错误的反跟踪 …… 26

1.1.5.6 调试限制 …… 26
1.1.5.7 用C开发而用嵌入式汇编编译最终代码 …… 26
1.2 从Turbo C中调用Turbo Assembler函数 …… 27
1.2.1 Turbo C与Turbo Assembler的接口机制 …… 27
1.2.1.1 内存模式和段 …… 28
1.2.1.2 公共量和外部量 …… 34
1.2.1.3 链接器命令行 …… 38
1.2.2 Turbo Assembler与Turbo C的交互性 …… 38
1.2.2.1 参数传递 …… 39
1.2.2.2 保存寄存器 …… 45
1.2.2.3 返回值 …… 45
1.2.3 从Turbo C中调用Turbo Assembler函数 …… 46
1.2.4 Pascal调用约定 …… 49
1.3 在Turbo Assembler中调用Turbo C …… 50
1.3.1 链入C的启动码 …… 50
1.3.2 确保已正确设置了段 …… 51
1.3.3 执行调用 …… 51
1.3.4 在Turbo Assembler调用Turbo C函数 …… 52

第二章 Turbo C与DOS、BIOS的接口 …… 54

2.1 寄存器 …… 54
2.2 中断 …… 55
2.2.1 使用DOS中断的注意事项 …… 55
2.3 利用功能调度器实现中断 …… 56
2.4 使用BIOS中断 …… 93
2.5 小结 …… 100

第二部分 Turbo Pascal与其它语言的接口

第三章 Turbo Pascal与汇编语言的接口 …… 102

3.1 扩展Turbo Pascal …… 102
3.2 嵌入指令 …… 104
3.3 外部过程 …… 105
3.3.1 外部函数 …… 105
3.3.2 使用全程数据和过程 …… 107
3.3.3 使用Turbo Assembler …… 109
3.4 嵌入代码与外部过程的比较 …… 112
3.5 使用Turbo Debugger …… 112

第四章　再论与 Turbo Pascal 和汇编语言的接口 …… 117

4.1　Turbo Pascal 内存映象 …… 117
4.1.1　程序段前缀 …… 117
4.1.2　代码段 …… 118
4.1.3　全局数据段 …… 118
4.1.4　堆栈 …… 119
4.1.5　堆 …… 119
4.2　Turbo Pascal 中寄存器的用法 …… 119
4.3　近调用还是远调用？ …… 119
4.4　与 Turbo Pascal 共享信息 …… 120
4.4.1　$L 编译伪指令和外部子程序 …… 120
4.4.2　PUBLIC 伪指令：使 Turbo Pascal 可利用 Turbo Assembler 的信息 … 121
4.4.3　EXTRN 伪指令：使 Turbo Assembler 可利用 Turbo Pascal 的信息 … 121
4.4.4　使用段定位 …… 124
4.4.5　无效代码的消除 …… 124
4.5　Turbo Pascal 参数传递约定 …… 124
4.5.1　值参 …… 125
4.5.1.1　标量类型 …… 125
4.5.1.2　实型 …… 125
4.5.1.3　单精度、双精度、扩展的和复合型：8087 类型 …… 125
4.5.1.4　指针 …… 125
4.5.1.5　串 …… 125
4.5.1.6　记录和数组 …… 125
4.5.1.7　集合 …… 126
4.5.2　变量参数 …… 126
4.5.3　栈的维护 …… 126
4.5.4　存取参数 …… 126
4.6　Turbo Pascal 中的函数结果 …… 129
4.6.1　标量函数结果 …… 129
4.6.2　实型函数结果 …… 129
4.6.3　8087 函数结果 …… 129
4.6.4　串函数结果 …… 129
4.6.5　指针函数结果 …… 129
4.7　为局部数据分配空间 …… 129
4.7.1　分配私有静态存贮区 …… 130
4.7.2　分配动态存贮区 …… 130
4.8　由 Turbo Pascal 调用汇编语言子程序的例子 …… 131
4.8.1　通用 16 进制转换子程序 …… 131

4.8.2 交换两个变量 …… 134
4.8.3 扫描 DOS 环境 …… 137

第五章 Turbo Pascal 与 DOS 和 BIOS 的接口 …… 142

5.1 8088 寄存器 …… 142
5.2 DOS 单元 …… 143
5.3 寄存器集 …… 144
5.4 磁盘驱动功能调用 …… 146
5.4.1 报告磁盘空闲空间 …… 146
5.4.2 读取和设置文件属性 …… 147
5.4.3 目录列表 …… 151
5.5 视频功能调用 …… 155
5.5.1 报告当前视频模式 …… 155
5.5.2 设置光标大小 …… 156
5.5.3 从屏幕读字符 …… 157
5.6 时间和日期功能 …… 159
5.6.1 获取系统日期 …… 159
5.6.2 设置系统日期 …… 160
5.6.3 获取和设置系统时间 …… 161
5.6.4 获取和设置文件的时间和日期 …… 164
5.6.5 报告换档键状态 …… 169
5.7 Turbo Pascal DOS 单元 …… 171
5.7.1 DOS 单元常量 …… 171
5.7.2 DOS 单元数据类型 …… 172
5.7.2.1 DateTime 类型 …… 173
5.7.2.2 SearchRec 类型 …… 173
5.7.3 DosError 变量 …… 173
5.7.4 DOS 单元过程与函数 …… 174
5.7.4.1 中断支持子程序 …… 174
5.7.4.2 日期和时间例程 …… 174
5.7.4.3 磁盘和文件例程 …… 174
5.7.5 进程例程 …… 175

第三部分 Turbo Basic 与其它语言的接口

第六章 Turbo Basic 与 Turbo Assemlber 的接口 …… 188

6.1 传递参数 …… 188
6.1.1 不在当前数据段的变量 …… 190
6.1.2 什么类型的调用? …… 190

6.2 弹出堆栈 …… 191
6.3 为 Turbo Basic 创建一个汇编程序 …… 191
6.4 调用一个在线汇编子程序 …… 191
6.5 在内存中安装一个 Turbo Basic 子程序 …… 193
6.5.1 隐藏串 …… 194
6.5.2 绝对调用(CALL ABSOLUTE) …… 195
6.5.2.1 到一固定内存位置作 CALL ABSOLUTE …… 196
6.5.2.2 到内存不定位置作 CALL ABSOLUTE …… 196
6.5.2.3 CALL ABSOLUTE 的其他问题 …… 197
6.6 调用中断 …… 197
6.7 样本程序 …… 198

第四部分 Turbo Prolog 与其它语言的接口

第七章 Turbo Prolog 与 Turbo C 的接口 …… 202

7.1 声明外部谓词 …… 202
7.2 调用约定和参数压栈顺序 …… 202
7.3 命名约定 …… 203
7.4 Turbo Prolog 调用 Turbo C 过程 …… 204
7.4.1 说明外部谓词 …… 204
7.4.2 建立 C 函数源程序 …… 204
7.4.3 Turbo C 编译选项和连接 …… 204
7.4.4 动态存贮分配 …… 205
7.4.5 传递复合对象到其它语言的程序 …… 206
7.4.6 例子 …… 207
7.5 Turbo C 调用 Turbo Prolog …… 210

第八章 Turbo Prolog 与 Turbo Assembler 的接口 …… 213

8.1 声明外部谓词 …… 213
8.2 调用约定和参数压栈 …… 213
8.3 命名约定 …… 214
8.4 编写汇编语言谓词 …… 214
8.5 用多重流模式实现谓词 …… 219
8.6 从汇编函数调用 Turbo Prolog 谓词 …… 220
8.7 表和函子 …… 222

第九章 Turbo Prolog 与 MS-Fortran 4.0 的接口 …… 226

9.1 系统设置 …… 226
9.2 Turbo Prolog 调用 MS-Fortran 过程 …… 226

9.2.1 在 Prolog 中说明外部谓词 …… 226
9.2.2 定义 Fortran 子程序并建立源程序 …… 227
9.2.2.1 命名约定 …… 227
9.2.2.2 参数约定 …… 227
9.2.2.3 屏幕输出 …… 227
9.2.3 连接步骤 …… 228
9.2.4 例子 …… 228
9.3 Fortran 调用 Turbo Prolog …… 230
9.4 常用接口例程库、预处理程序的建立以及 Fortran 库的改造 …… 231
9.4.1 常用接口例程库的建立 …… 231
9.4.2 预处理程序 …… 232
9.4.3 Fortran 库的改造 …… 234

第十章 Turbo Prolog 访问 dBASE Ⅲ 数据文件 …… 240

10.1 Prolog 事实与 dBASE Ⅲ 记录 …… 240
10.2 dBASE Ⅲ 中 DBF 的存贮结构 …… 240
10.3 把 DBF 记录转换成 Turbo Prolog 事实 …… 241
10.4 利用 Turbo Prolog 工具库访问 dBASE Ⅲ 数据文件 …… 242
10.4.1 一次读出 dBASE Ⅲ 文件的所有记录 …… 242
10.4.2 一次读出一个 dBASE Ⅲ 记录 …… 243

第十一章 Turbo Prolog 与 DOS 系统级的接口 …… 250

11.1 访问 DOS …… 250
11.1.1 system/1 …… 250
11.1.2 system/3 …… 250
11.1.3 envsymbol/2 …… 251
11.1.4 date/3 和 time/4 …… 252
11.1.5 comline/1 …… 252
11.2 访问硬件：低级支撑 …… 253
11.2.1 bios/3 和 bios/4 …… 253
11.2.2 ptr-dword/3 …… 254
11.2.3 membyte/3 和 memword/3 …… 254
11.2.4 port-byte/2 …… 255
11.3 例子： …… 255

第五部分 混合编程程序的调试

第十二章 Turbo Debugger 调试的一个快速示例 …… 258

12.1 演示程序 …… 258

12.2 使用 Turbo Debugger …… 259
12.2.1 菜单(The menus) …… 259
12.2.2 状态行(The status line) …… 260
12.2.3 窗口(The windows) …… 260
12.3 使用 C 演示程序 …… 261
12.3.1 设置断点(Setting breakpoints) …… 262
12.3.2 利用监视(Using watches) …… 262
12.3.3 考察简单的 C 数据对象 …… 263
12.3.4 考察复杂的 C 数据的对象 …… 264
12.3.5 改变 C 数据值 …… 265
12.4 使用 Pascal 示例程序 …… 266
12.4.1 设置断点(Setting breakpoints) …… 267
12.4.2 使用监视(Using watches) …… 268
12.4.3 考察简单的 Pascal 数据对象 …… 268
12.4.4 考察复杂的 Pascal 数据对象 …… 269
12.4.5 改变 Pascal 数据值 …… 269

第十三章 启动 Turbo Debugger …… 272

13.1 准备待调试的程序 …… 272
13.1.1 准备 Turbo C 程序 …… 272
13.1.2 准备 Turbo Pascal 程序 …… 272
13.1.3 准备 Turbo 汇编程序 …… 273
13.1.4 准备 Microsoft 程序 …… 273
13.2 运行 Turbo Debugger …… 273
13.3 命令行选择项 …… 274
13.3.1 装载配置文件(-c) …… 274
13.3.2 显示更新方式(-d) …… 274
13.3.3 获取帮助(-h 与-?) …… 274
13.3.4 进程 ID 转换(-i) …… 274
13.3.5 击键记录(-k) …… 274
13.3.6 汇编模式启动(-l) …… 275
13.3.7 设置堆大小(-m) …… 275
13.3.8 鼠标器支持(-p) …… 275
13.3.9 远程调试(-r) …… 275
13.3.10 源代码处理(-s) …… 275
13.3.11 视频硬件(-v) …… 276
13.3.1.2 覆盖池大小(-y) …… 276
13.4 配置文件 …… 276
13.5 选项菜单 …… 277

13.5.1 语言命令 …… 277
13.5.2 宏菜单 …… 277
13.5.2.1 创建(Create) …… 277
13.5.2.2 停止记录(Stop Recording) …… 277
13.5.2.3 删除(Remove) …… 278
13.5.2.4 全清(Delete All) …… 278
13.5.3 显示选择命令 …… 278
13.5.3.1 显示切换 …… 278
13.5.3.2 整数格式 …… 278
13.5.3.3 屏幕行数 …… 278
13.5.3.4 制表键大小 …… 279
13.5.4 源命令路径 …… 279
13.5.5 保存选择项命令 …… 279
13.5.6 恢复选择项命令 …… 280
13.6 在 Turbo Debugger 中运行 DOS …… 280
13.7 返回 DOS …… 280

第六部分 混合编程的参考资料

附录 A TASM 命令行参考 …… 282
A.1 在 DOS 中启动 Turbo Assembler …… 282
A.2 命令行选择项 …… 284
附录 B 混合编程实用程序 …… 295
B.1 独立的 MAKE 实用程序 …… 295
B.1.1 一个快速示例 …… 295
B.1.1.1 创建一个 make 文件 …… 296
B.1.1.2 使用一个 make 文件 …… 297
B.1.1.3 步进 …… 298
B.1.2 创建 make 文件 …… 298
B.1.2.1 Make 文件的组成 …… 298
B.1.3 使用 MAKE …… 309
B.1.3.1 命令行语法 …… 309
B.1.3.2 中止 MAKE 的说明 …… 310
B.1.3.3 BUILTINS.MAK 文件 …… 310
B.1.3.4 MAKE 是如何查找 make 文件的 …… 310
B.1.3.5 TOUCH 实用程序 …… 311
B.1.3.6 MAKE 命令行选择项 …… 311
B.1.4 MAKE 出错信息 …… 311
B.1.4.1 致命错 …… 311
B.1.4.2 一般错 …… 312

B.2 Turbo Link …… 313
B.2.1 调用 TLINK …… 313
B.2.2 使用应答文件 …… 314
B.2.3 TLINK 选择项 …… 315
B.2.3.1 /x,/m,/s 选择项 …… 315
B.2.3.2 /l 选择项 …… 316
B.2.3.3 /i 选择项 …… 316
B.2.3.4 /n 选择项 …… 316
B.2.3.5 /c 选择项 …… 316
B.2.3.6 /d 选择项 …… 317
B.2.3.7 /e 选择项 …… 317
B.2.3.8 /t 选择项 …… 317
B.2.3.9 /v 选择项 …… 317
B.2.3.10 /s 选择项 …… 317
B.2.4 一些限制 …… 317
B.2.5 出错消息 …… 318
B.2.5.1 致命错 …… 318
D.2.5.2 非致命错 …… 319
B.2.5.3 警告 …… 319
B.3 TLIB:Turbo 库管理员 …… 320
B.3.1 使用目标模块库的优点 …… 320
B.3.2 TLIB 命令行的组成 …… 320
B.3.3 操作表(Operations) …… 321
B.3.4 使用应答文件 …… 322
B.3.5 改进的操作:/c 选择项 …… 322
B.3.6 例子 …… 323
B.3.7 创建一扩展词典: /E 选择项 …… 323
B.4 GREP:一种文件查找实用程序 …… 324
B.4.1 GREP 选择项 …… 324
B.4.1.1 优先级次序 …… 325
B.4.2 查找串 …… 325
B.4.2.1 正则表达式中的操作符 …… 325
B.4.3 文件说明 …… 326
B.4.4 带说明的例子 …… 326
B.5 OBJXREF:目标模块交叉引用实用程序 …… 328
B.5.1 OBJXREF 命令行 …… 328
B.5.1.1 命令行选择项 …… 329
B.5.2 应答文件 …… 329
B.5.2.1 自由形式的应答文件 …… 330

B.5.2.2 连接器应答文件 …… 330
B.5.2.3 /D命令 …… 330
B.5.2.4 /O命令 …… 330
B.5.2.5 /N命令 …… 330
B.5.3 OBJXREF报告样本 …… 331
B.5.3.1 按公用名报告(/RP) …… 332
B.5.3.2 按模块报告(/RM) …… 332
B.5.3.3 按引用报告(/RR)(缺省方式) …… 332
B.5.3.4 按外部引用报告(/RX) …… 333
B.5.3.5 按模块长度报告(/RS) …… 333
B.5.3.6 按类报告(/RC) …… 333
B.5.3.7 按未引用符号名报告(/RV) …… 334
B.5.3.8 冗长报告(/RV) …… 334
B.5.4 使用OBJXREF的例子 …… 334
B.5.5 OBJXREF出错信息和警告 …… 335
B.5.5.1 出错信息 …… 335
B.5.5.2 警告 …… 335
B.6 TCREF：源模块交叉引用实用程序 …… 335
B.6.1 应答文件 …… 336
B.6.2 与TLINK的兼容 …… 336
B.6.2.1 开关 …… 336
B.6.2.2 全局(或连接器级)报告 …… 336
B.6.2.3 局部(或模块级)报告 …… 337

参考文献 …… 338

第0章 概 述

随着微型机应用的不断普及，在各类应用软件开发过程中，为了利用各种高级程序设计语言之间的混合编程技术，人们一直关注着高级程序设计语言与汇编程序设计语言之间的接口技术问题。

众所周知，高级语言具有如下的特点：

- 数据结构丰富，具有现代化语言的各种数据结构。高级语言一般都包括整型、实现、字符型、数组类型、指针类型、结构类型、联合类型等。
- 具有结构化的控制语句如 if...else 语句、while 语句、case 语句、for 语句、repeat...until 语句、do...while 语句等等。用函数作为程序作为程序模块以实现的模块化，结构化的语言，符合现化编程风格要求。
- 丰富的函数，如字符类型判定函数、目录控制函数、图形函数、输入/输出函数、接口函数、数学函数、进行程控制函数、文本窗口显示函数和时间与日期函数等等，一般的语言都具有上百个函数，多的可达几百个函数。标准的函数可以充分加快程序的开发。
- 运算符丰富。高级语言的运算符包罗万象，如加减乘除、括号、赋值和强制类型转换等等。
- 用语言使用表意的关键字编写的程序可读性强。
- 用高级语言编写的程序可移植性好。

而汇编程序设计语言是一种除机器外最靠近机器的面向机器的语言，除伪指令与宏指令外，它与机器指令是一一对应的。程序设计者可以充分利用机器指令的各种各种特有功能，发挥编程技巧，编写出占空间小、运行效率高的高质量程序模块来。

采用汇编语言编写程序存在下列问题：

- 降低了可移植性和可维护性
- 降低了编译速度
- 损失了优化能力
- 限制了对错误的反跟踪
- 调试限制

总之，汇编语言存在着不易掌握，编程困难且费时，程序可读性差，且其程序质量直接到编程人员的技术水平的影响的缺点。与汇编语言相比，面向问题的高级程序设计语言，如 Fortran，Pascal 和 C 等具有易掌握、编写程序既容易又省时、程序可读性好、容易移植等优点。但是，一般来说，各高级程序设计语言很难充分利用硬件所具有的全部功能。

基于上述原因和各类程序设计语言的特点，权衡利弊，在开发应用程序时，人们自然而

然地将注意力转向高级程序设计语言与汇编程序设计语言的混合编程技术上。设想用汇编程序设计语言来编写应用软件中少量的、最低层的、调用频率最高的且直接影响系统效率的或用高级语言难以实现的程序模块，而用高级程序设计语言来编写其它的大量的上层模块。这样，两者综合使用，充分利用两类程序设计语言的长处，既提高了软件的质量，又不影响软件的开发周期。

当然，要实现这一全理的设想，其关键在于必须搞清高级程序设计语言与汇编程序设计语言之间的接口技术和约定。

另外，各种高级语言各有专攻，优劣之处各不相同。一个大系统软件可能存在着使用不同语言的众多程序员；另外，一个程序员可能改变了使用的语言，比如从 Pascal 转到 C 或从 C 转到 Pascal，而想重用已前的编写的函数。因而也有必要在高级语言之间能互通有无，互相调用函数和过程。

近年来，作者在教学与科研中经常遇到这个问题。在项目开发中集中对 Borland 公司高级程序设计语言，如 C、Pascala、C、Basic 和其宏汇编程序设计语言(Turbo Assmbler、TASM)之间的接口技术与约定作了深入的探讨与实践，进而又对 Microsoft 系统的高级程序设计语言和汇编语言之间的接口技术进行对比与分析，摸索出一套较完整的接口技术与规则，有效地解决了上述问题。现将它综述于下，希望它有助于促进软件开发。

0.1 适合高级程序设计语言调用的汇编语言子程序的编写格式

无论是 Microsoft 公司的，还是 Borland 公司的系统语言，调用程序与被调用程序之间都是通过栈来传递参数的，这是这两个公司的产品的共同点，所以，一般要编写一个能被高级程序设计调用的汇编语言子程序都应按下现规则与格进行编写。就具体的程序设计语言而言，只要在使用这些规则与格式时注意到具体语言的某些特殊约定就可以了。

这种汇编子程序的格式与编写步骤如下：

- 建立过程；
- 进入过程；
- 分配局部数据存储区；
- 保留寄存器的值；
- 按各高级程序语言的参数传递规则存、取各参数值，编写处理这些数据的子程序体；
- 传送返回值；
- 恢复寄存器值并退出过程。返回时，根据不同的高级程序设计语言的约定，删除(或不删除)栈中的参数，恢复 sp 到调用它之前的值。

其具体框架如下：

```
        .MODEL[small |large]            ;建立过程
        CODE
                PUBLIC procname
procname    PROC [near|far]
        PUSH    BP                      ;进入过程
```

```
                MOV     BP,SP
                SUB     SP,Napace       ;分配局部数据空间
                PUSH    SI              ;保存调用者的寄存值
                PUSH    DI
                .   .   .               ;根据约定取参数值、过程体
                POP     DI              ;恢复保留的各寄存器之值
                POP     SI
                MOV     SP,BP           ;释放局部数据空间
                POP     BP              ;恢复 BP
                RET     [size]          ;恢复(或不恢复)SP,并返回
procname        ENDP
                END
```

下面逐一解释框架中各步骤的作用:

0.1.1 建立过程

以 MASM(或 TASM)5.0为例,可用简化段指示符来说明. MODEL 定义存储模式,它可有 SMALL(小模式)、MEDIUM(中模式)、LARGE(大模式)和 HUGE(巨模式)等几种选择,要注意的是它必须与高级语言所采用的存储模式相一致。通常,FORTRAN 常用 LARGE 模式,而 C 常用 SMALL 模式. CODE 定义代码段,用 PUBLIC 指示符说明过程名 procname 为公用的。过程名的命名规则应与相应的高级语言相一致,具体见表一。

0.1.2 进入过程,建立一个参数表的固定基点

指令

```
PUSH    BP
MOV     BP,SP
```

就是为此目的而设立的。首先保留调用代码的 BP 值,然后将栈指针传送给 BP。这样,汇编子程序中可以以 BP 为基准的来存取位于栈中的各参数的值(或地址)了。这时,各参数都可以编址 BP 为基准的一个固定的偏移位置上,便于后面代码的引用。以 FORTRAN 程序中的调用语句为例,CALL SUBI(A,B,C)执行后,它将实在参数的远地址(段址:偏移)顺序压入栈(STACK)之中,再压入一个返回地址(段址:偏移),接着将控制转到了程序 SUBI 的入口处。执行 PUSH BP 和 MOV BP、SP 之后,其栈的状况如图0.1所示。

从图中看到,实参 A、B、C 在栈中的地址均可用 BP 加一常数偏移量,分别用 BP+14、BP+10、BP+6来表示。

0.1.3 分配局部数据空间

这一步可以根据子程序中的需要而定,不是必须的。通常在栈中保留一段空间,用 SUB SP、Napace 指令来实现。建议 Napace 最好为偶数,这是因为栈的 PUSH 和 POP 操作指令都是以2字节为单位的。假设 Nspace=4、那么指令:SUBSP,4执行后,变将 SP 就改变了,并保留了4个字节的空间,这个局部空间的位置可用相对于 BP 的负位移值来确定,分别可用 BP-2和 BP-4来表示。如有必要初始化,则可用指令序列: