

The Art of Assembly Language

汇编语言编程艺术

(美) Randall Hyde 著

陈曙晖 翻译
毛希平 审校



清华大学出版社

汇编语言编程艺术

(美) Randall Hyde 著

陈曙晖 翻译

毛希平 审校

清华大学出版社

北 京

Randall Hyde

The Art of Assembly Language

EISBN: 1-886411-97-2

Copyright© 2003 by No Starch Press.

Authorized translation from the English language edition published by No Starch Press.

All rights reserved. For sale in the People's Republic of China only.

Chinese simplified language edition published by Tsinghua University Press.

本书中文简体字翻译版由 No Starch 出版社授权清华大学出版社在中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾地区)独家出版发行。未经许可之出口视为违反著作权法,将受法律之制裁。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

版权所有,翻印必究。举报电话:010-62782989 13901104297 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用清华大学核研院专有核径迹膜防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

北京市版权局著作权合同登记号 图字:01-2003-8207

图书在版编目(CIP)数据

汇编语言编程艺术/(美)海德(Hyde, R.)著;陈曙晖 翻译.

—北京:清华大学出版社,2005.1

书名原文:The Art of Assembly Language

ISBN 7-302-09057-2

I. 汇… II. ①海…②陈… III. 汇编语言 IV. TP313

中国版本图书馆 CIP 数据核字(2004)第 070988 号

出 版 者:清华大学出版社 地 址:北京清华大学学研大厦

<http://www.tup.com.cn> 邮 编:100084

社 总 机:010-62770175 客 户 服 务:010-62776969

组稿编辑:曹 康

文稿编辑:崔 伟 李万红

封面设计:康 博

版式设计:康 博

印 刷 者:清华大学印刷厂

装 订 者:三河市金元装订厂

发 行 者:新华书店总店北京发行所

开 本:185×260 印 张:49.75 字 数:1273 千字

版 次:2005 年 1 月第 1 版 2005 年 1 月第 1 次印刷

书 号:ISBN 7-302-09057-2/TP·6399

印 数:1~4000

定 价:98.00 元

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770175-3103 或(010)62795704

序

汇编语言是计算机提供给用户的一种面向机器的编程语言，这种语言可以最大限度地利用计算机硬件特性并能通过汇编指令直接控制机器硬件，因此，利用汇编语言可以编写出在时间和空间上最具效率的程序。

要想很好地学习、掌握汇编语言，一本好书是必不可少的。《汇编语言编程艺术》就是近年来出现的一本好书。这本书以 80x86 系列机为背景，通过大量的程序代码详细地介绍了 80x86 汇编语言的基础知识，特别是汇编语言的编程方法和技巧。作者将多年来的开发和教学经验融合在大量的编程实例中，读者通过本书能快速学会汇编语言程序设计，掌握其中的编程技巧，并能从一开始就养成良好的编程风格，从而实现从初学者到高级编程人员的过渡。

本书的作者 Randall Hyde 在大学中教授汇编语言十多年，并且开发了好几个商用软件，具有丰富的汇编语言教学和开发经验。在本书中作者还介绍了一种高级汇编语言 HLA (High Level Assembly)，最开始 HLA 是加州大学的教授用来讲解汇编语言编程和机器组织的一个工具，经过几年的发展，如今已经成为了资深汇编语言程序员编写可读性强、功能强大的汇编程序的开发平台。它的变量声明、过程声明、过程调用等都使用与高级语言类似的语法，同时还可以使用函数库。如果读者熟悉像 C/C++、Pascal/Delphi、Java 或者 VB 这样的高级语言，就会发现使用 HLA 既像高级语言一样方便，又保持了汇编语言的高效率。

本书的英文书名是 “The Art of Assembly Language”。在以英语为母语的国家的国家中，人们习惯于把一切比较需要技术的、神奇的、难以用机械的方式进行重复的东西称之为 “Art”。我认为正是由于汇编语言所具有的丰富而又灵活的功能，才使程序员在程序设计中能充分发挥自己的编程技巧，就像艺术家一样，程序员也可创作出 “精美” 的程序来。

要说本书不足之处，我想可能就是它的 “大部头”，书中有的内容写的过细，会使读者觉得有点 “啰嗦”。另外，学习本书内容，需要配备 HLA 软件，但令人欣慰的是，HLA 软件可以免费下载。

很高兴看到，清华大学出版社及时引进该书的中文版和英文版，其中文版是由两位计算机博士翻译和审校的，他们较准确地把握和传递了全书的精髓。我相信，这本书一定会吸引广大专业工作者以及年青的程序设计爱好者。

清华大学教授 温冬婵

二零零四年十二月

目 录

第 1 章 进入汇编语言的世界	1
1.1 本章概述	1
1.2 HLA 程序的结构	1
1.3 运行第一个 HLA 程序	3
1.4 基本的 HLA 数据声明	4
1.5 布尔值	6
1.6 字符值	6
1.7 Intel 80x86 处理器简介	7
1.8 基本的机器指令	11
1.9 基本的 HLA 控制结构	15
1.9.1 HLA 语句中的布尔表达式	15
1.9.2 HLA 的 IF..THEN..ELSEIF..ELSE..ENDIF 语句	17
1.9.3 布尔表达式中的逻辑与、逻辑或以及逻辑非	19
1.9.4 WHILE..ENDWHILE 语句	21
1.9.5 FOR..ENDFOR 语句	21
1.9.6 REPEAT..UNTIL 语句	22
1.9.7 BREAK 和 BREAKIF 语句	23
1.9.8 FOREVER..ENDFOR 语句	23
1.9.9 TRY..EXCEPTION..ENDTRY 语句	24
1.10 HLA 标准库入门	27
1.10.1 STDIO 模块中的预定义常量	28
1.10.2 标准输入与标准输出	29
1.10.3 stdout.newln 例程	29
1.10.4 stdout.putiX 例程	29
1.10.5 stdout.putiXSize 例程	30
1.10.6 stdout.put 例程	31
1.10.7 stdin.getc 例程	32
1.10.8 stdin.getiX 例程	34
1.10.9 stdin.readLn 与 stdin.flushInput 例程	35
1.10.10 stdin.get 例程	36
1.11 关于 TRY..ENDTRY 的其他细节	36

1.11.1 TRY..ENDTRY 嵌套语句	37
1.11.2 TRY..ENDTRY 语句中不受保护的子句	39
1.11.3 TRY..ENDTRY 语句中的 ANYEXCEPTION 子句	42
1.11.4 寄存器与 TRY..ENDTRY 语句	42
1.12 高级汇编语言与低级汇编语言的比较	44
1.13 更多信息	45
第 2 章 数据表示	46
2.1 本章概述	46
2.2 数字系统	46
2.2.1 回顾十进制系统	46
2.2.2 二进制数字系统	47
2.2.3 二进制格式	48
2.3 十六进制数字系统	49
2.4 数据结构	51
2.4.1 位	51
2.4.2 半字节	51
2.4.3 字节	52
2.4.4 字	53
2.4.5 双字	54
2.4.6 四字与长字	55
2.5 二进制数与十六进制数的算术运算	56
2.6 关于数字及其表示法	56
2.7 位逻辑运算	59
2.8 二进制数和位串的逻辑运算	61
2.9 有符号数和无符号数	63
2.10 符号扩展、零扩展、压缩和饱和	67
2.11 移位和循环移位	71
2.12 位域和压缩数据	75
2.13 浮点运算简介	79
2.13.1 IEEE 浮点格式	82
2.13.2 HLA 为浮点数值提供的支持	85
2.14 BCD 数据表示	88
2.15 字符	89
2.15.1 ASCII 字符译码	89
2.15.2 HLA 对 ASCII 字符提供的支持	92

2.16	Unicode 字符集	96
2.17	更多信息	96
第 3 章	存储器的访问与结构	97
3.1	本章概述	97
3.2	80x86 的寻址方式	97
3.2.1	80x86 寄存器寻址方式	97
3.2.2	80x86 的 32 位存储器寻址方式	98
3.3	运行时存储器的结构	104
3.3.1	代码段	105
3.3.2	静态段	106
3.3.3	只读数据段	107
3.3.4	存储段	108
3.3.5	@NOSTORAGE 属性	108
3.3.6	Var 段	109
3.3.7	程序中声明段的结构	110
3.4	HLA 如何为变量分配内存	111
3.5	HLA 对数据对齐的支持	112
3.6	地址表达式	115
3.7	类型强制转换	117
3.8	寄存器类型强制转换	119
3.9	栈段与 PUSH 及 POP 指令	120
3.9.1	基本的 PUSH 指令	120
3.9.2	基本的 POP 指令	121
3.9.3	用 PUSH 和 POP 指令来保护寄存器	123
3.9.4	栈的 LIFO 数据结构	123
3.9.5	其他的 PUSH 和 POP 指令	125
3.9.6	不使用出栈而从栈内移除数据	127
3.9.7	访问已入栈而未出栈的数据	129
3.10	动态内存分配和堆段	130
3.11	INC 和 DEC 指令	134
3.12	获取存储器对象的地址	134
3.13	更多信息	135
第 4 章	常量、变量与数据类型	136
4.1	本章概述	136

4.2	一些额外的指令: INTMUL、BOUND、INTO	136
4.3	TBYTE 数据类型	141
4.4	HLA 常量和数值声明	141
4.4.1	常量类型	145
4.4.2	字符串和字符字面常量	145
4.4.3	CONST 段中的字符串常量与文本常量	147
4.4.4	常量表达式	149
4.4.5	HLA 程序中的多个 CONST 段以及它们的顺序	151
4.4.6	HLA 的 VAL 段	151
4.4.7	在程序中的任意位置修改 VAL 对象	152
4.5	HLA 的 TYPE 段	153
4.6	ENUM 和 HLA 枚举数据类型	154
4.7	指针数据类型	155
4.7.1	在汇编语言中使用指针	156
4.7.2	在 HLA 中声明指针	157
4.7.3	指针常量和指针常量表达式	158
4.7.4	指针变量和动态内存分配	159
4.7.5	指针的常见问题	160
4.8	HLA 标准库 CHARS.HHF 模型	164
4.9	复合数据类型	166
4.10	字符串	167
4.11	HLA 字符串	169
4.12	访问字符串中的某个字符	175
4.13	HLA 字符串模块和其他与字符串相关的例程	177
4.14	存储器内转换	188
4.15	字符集	190
4.16	在 HLA 中实现字符集	190
4.17	HLA 字符集常量和字符集表达式	192
4.18	HLA HLL 布尔表达式中的 IN 操作符	193
4.19	HLA 标准库对字符集的支持	194
4.20	在 HLA 程序中使用字符集	197
4.21	数组	198
4.22	在 HLA 程序中声明数组	199
4.23	HLA 数组常量	200
4.24	访问一维数组的元素	201

4.25	多维数组	204
4.25.1	以行为主排列	205
4.25.2	以列为主排列	208
4.26	多维数组的存储空间分配	209
4.27	汇编语言中多维数组元素的访问	211
4.28	大数组和 MASM(只适用于 Windows 程序员)	212
4.29	记录	212
4.30	记录常量	215
4.31	记录数组	216
4.32	数组/记录作为记录字段	217
4.33	控制记录中的字段偏移量	220
4.34	对齐记录中的字段	221
4.35	记录指针	223
4.36	联合	224
4.37	匿名联合	226
4.38	变量类型	227
4.39	联合常量	228
4.40	命名空间	229
4.41	汇编语言中的动态数组	232
4.42	HLA 标准库数组支持	234
4.43	更多信息	237
第 5 章	过程与单元	238
5.1	本章概述	238
5.2	过程	238
5.3	机器状态的保存	240
5.4	过程的提前返回	244
5.5	局部变量	245
5.6	其他局部和全局符号类型	250
5.7	参数	250
5.7.1	值传递	251
5.7.2	引用传递	254
5.8	函数和函数的结果	257
5.8.1	返回函数结果	257
5.8.2	HLA 的指令合成	258
5.8.3	HLA 过程的@RETURNS 选项	260

5.9	递归	262
5.10	过程的向前引用	266
5.11	过程的底层实现与 CALL 指令	267
5.12	过程与堆栈	269
5.13	活动记录	272
5.14	标准入口序列	275
5.15	标准出口序列	276
5.16	自动(局部)变量的底层实现	277
5.17	参数的底层实现	279
5.17.1	在寄存器中传递参数	279
5.17.2	在代码流中传递参数	281
5.17.3	在堆栈中传递参数	284
5.18	过程指针	304
5.19	过程参数	307
5.20	无类型的引用参数	308
5.21	管理大型程序	309
5.22	#INCLUDE 伪指令	310
5.23	忽略重复的 #INCLUDE 操作	311
5.24	单元与 EXTERNAL 伪指令	312
5.24.1	伪指令 EXTERNAL 的行为	316
5.24.2	HLA 中的头文件	317
5.25	命名空间的污染	319
5.26	更多信息	321
第 6 章	算术运算	322
6.1	本章概述	322
6.2	80x86 的整数运算指令	322
6.2.1	MUL 和 IMUL 指令	322
6.2.2	DIV 和 IDIV 指令	325
6.2.3	CMP 指令	327
6.2.4	SETcc 指令	331
6.2.5	TEST 指令	333
6.3	算术表达式	334
6.3.1	简单赋值语句	335
6.3.2	简单表达式	336
6.3.3	复杂表达式	338

6.3.4 可交换运算符	342
6.4 逻辑(布尔)表达式	343
6.5 机器特征与运算技巧	345
6.5.1 不使用 MUL、IMUL 或 INTMUL 的乘法	346
6.5.2 不使用 DIV 或 IDIV 的除法	347
6.5.3 使用 AND 实现模 N 计数器	347
6.5.4 疏忽使用机器特性	348
6.6 浮点运算	348
6.6.1 FPU 寄存器	348
6.6.2 FPU 的数据类型	355
6.6.3 FPU 的指令集	356
6.6.4 FPU 的数据转移指令	356
6.6.5 换算指令	358
6.6.6 算术运算指令	360
6.6.7 比较指令	365
6.6.8 常量指令	367
6.6.9 超越指令	367
6.6.10 其他指令	369
6.6.11 整数操作	370
6.7 浮点表达式到汇编语言的转换	370
6.7.1 算术表达式到后缀表示法的转换	372
6.7.2 把后缀表达式转换成汇编语言	373
6.8 HLA 标准库对浮点算术运算的支持	375
6.8.1 函数 stdin.getf 和 fileio.getf	375
6.8.2 HLA 数学库中的三角函数	375
6.8.3 HLA 数学库中的指数函数和对数函数	376
6.9 算术运算小结	377
第 7 章 低级控制结构	378
7.1 本章概述	378
7.2 低级控制结构	378
7.3 语句标号	378
7.4 无条件控制转移(JMP)	380
7.5 条件跳转指令	383
7.6 “中级”控制结构: JT 和 JF	386
7.7 使用汇编语言实现通用控制结构	386

7.8 选择	386
7.8.1 IF..THEN..ELSE 序列	388
7.8.2 将 HLA 的 IF 语句翻译成纯汇编语言语句	391
7.8.3 使用完全布尔求值实现复杂的 IF 语句	396
7.8.4 “短路”布尔求值	397
7.8.5 “短路”布尔求值与完全布尔求值	399
7.8.6 汇编语言中 IF 语句的高效实现	401
7.8.7 SWITCH/CASE 语句	405
7.9 状态机和间接跳转	415
7.10 “面条式”代码	418
7.11 循环	418
7.11.1 WHILE 循环	419
7.11.2 REPEAT..UNTIL 循环	420
7.11.3 FOREVER..ENDFOR 循环	421
7.11.4 FOR 循环	422
7.11.5 BREAK 和 CONTINUE 语句	423
7.11.6 寄存器的使用与循环	427
7.12 性能提高	428
7.12.1 将结束条件判断放在循环结尾	429
7.12.2 反向执行循环	431
7.12.3 循环不变计算	432
7.12.4 循环展开	433
7.12.5 归纳变量	434
7.13 HLA 中的混合控制结构	435
7.14 更多信息	437
第 8 章 文件	438
8.1 本章概述	438
8.2 文件组织	438
8.2.1 作为记录列表的文件	438
8.2.2 二进制文件与文本文件的比较	440
8.3 顺序文件	442
8.4 随机访问文件	449
8.5 ISAM 文件	453
8.6 截断文件	456
8.7 更多信息	458

第 9 章 高级算术运算	459
9.1 本章概述	459
9.2 多精度操作	459
9.2.1 扩充精度操作的 HLA 标准库支持	459
9.2.2 多精度加法操作	462
9.2.3 多精度减法操作	464
9.2.4 扩充精度比较操作	465
9.2.5 扩充精度乘法操作	470
9.2.6 扩充精度除法操作	473
9.2.7 扩充精度 NEG 操作	482
9.2.8 扩充精度 AND 操作	483
9.2.9 扩充精度 OR 操作	484
9.2.10 扩充精度 XOR 操作	484
9.2.11 扩充精度 NOT 操作	485
9.2.12 扩充精度移位操作	485
9.2.13 扩充精度循环操作	488
9.2.14 扩充精度 I/O	489
9.3 对不同长度的操作数进行操作	509
9.4 十进制算术运算	510
9.4.1 文字 BCD 常量	512
9.4.2 80x86 的 DAA 指令和 DAS 指令	512
9.4.3 80x86 AAA、AAS、AAM 和 AAD 指令	514
9.4.4 使用 FPU 的压缩十进制算术操作	515
9.5 表	517
9.5.1 通过表查找进行函数计算	517
9.5.2 域调节	521
9.5.3 产生表	522
9.5.4 表查找的性能	526
9.6 更多信息	526
第 10 章 宏与 HLA 编译时语言	527
10.1 本章概述	527
10.2 编译时语言	527
10.3 #PRINT 和 #ERROR 语句	528
10.4 编译时常量和变量	530
10.5 编译时表达式和操作符	530

10.6	编译时函数	533
10.6.1	类型转换编译时函数	533
10.6.2	数字编译时函数	535
10.6.3	字符分类编译时函数	535
10.6.4	编译时字符串函数	535
10.6.5	编译时模式匹配函数	536
10.6.6	编译时符号信息	537
10.6.7	其他编译时函数	538
10.6.8	编译时 TEXT 对象的类型转换	539
10.7	条件编译(编译时决定)	540
10.8	重复编译(编译时循环)	544
10.9	宏(编译时过程)	547
10.9.1	标准宏	548
10.9.2	宏的参数	550
10.9.3	宏中的局部符号	556
10.9.4	作为编译时过程的宏	559
10.9.5	使用宏模拟函数重载	559
10.10	编写编译时“程序”	565
10.10.1	在编译时构造数据表	565
10.10.2	循环展开	570
10.11	在不同的源文件中使用宏	571
10.12	更多信息	571
第 11 章	位操作	573
11.1	本章概述	573
11.2	位数据	573
11.3	位操作指令	574
11.4	作为位累加器的进位标志位	581
11.5	位串的压缩与解压缩	581
11.6	接合位组与分布位串	584
11.7	压缩的位串数组	586
11.8	搜索位	588
11.9	位的计数	590
11.10	倒置位串	593
11.11	合并位串	595
11.12	提取位串	596

11.13	搜索位模式	598
11.14	HLA 标准库的位模块	599
11.15	更多信息	600
第 12 章	字符串指令	602
12.1	本章概述	602
12.2	80x86 字符串指令	602
12.2.1	字符串指令的操作过程	603
12.2.2	REP/REPE/REPZ 和 REPNZ/REPNE 前缀	603
12.2.3	方向标志位	604
12.2.4	MOVS 指令	606
12.2.5	CMPS 指令	611
12.2.6	SCAS 指令	614
12.2.7	STOS 指令	615
12.2.8	LODS 指令	615
12.2.9	从 LODS 和 STOS 构建复杂的字符串函数	616
12.3	80x86 字符串指令的性能	617
12.4	更多信息	617
第 13 章	MMX 指令集	618
13.1	本章概述	618
13.2	判断 CPU 是否支持 MMX 指令集	618
13.3	MMX 编程环境	619
13.3.1	MMX 寄存器	619
13.3.2	MMX 数据类型	621
13.4	设计 MMX 指令集的目的	622
13.5	饱和算法和回转模式	622
13.6	MMX 指令操作数	623
13.7	MMX 技术指令	625
13.7.1	MMX 数据传递指令	625
13.7.2	MMX 转换指令	625
13.7.3	MMX 压缩算术指令	630
13.7.4	MMX 逻辑指令	632
13.7.5	MMX 比较指令	634
13.7.6	MMX 移位指令	637
13.7.7	EMMS 指令	639

13.8	MMX 编程方案	640
13.9	更多信息	650
第 14 章	类与对象	651
14.1	本章概述	651
14.2	通用原则	651
14.3	HLA 中的类	653
14.4	对象	656
14.5	继承	657
14.6	重载	658
14.7	虚拟方法与静态过程	659
14.8	编写类方法和过程	661
14.9	对象实现	665
14.9.1	虚拟方法表	668
14.9.2	带继承的对象表达式	669
14.10	构造函数和对象初始化	673
14.10.1	构造函数中的动态对象分配	674
14.10.2	构造函数和继承	676
14.10.3	构造函数的参数和过程重载	680
14.11	析构函数	680
14.12	HLA 的 “_initialize_” 和 “_finalize_” 字符串	681
14.13	抽象方法	687
14.14	运行时类型信息(RTTI)	690
14.15	调用基类的方法	691
14.16	更多信息	692
第 15 章	混合语言编程	693
15.1	本章概述	693
15.2	在同一程序中混合使用 HLA 和 MASM/Gas 代码	693
15.2.1	在 HLA 程序中内嵌(MASM/Gas)汇编代码	693
15.2.2	链接 MASM/Gas 汇编模块和 HLA 模块	696
15.3	使用 Delphi/Kylix 和 HLA 编程	700
15.3.1	链接 HLA 模块与 Delphi/Kylix 程序	701
15.3.2	寄存器保存	704
15.3.3	函数的结果	705
15.3.4	调用惯例	711

15.3.5 Kylix 中的值传递、引用传递、CONST 参数和 OUT 参数	717
15.3.6 Delphi/Kylix 和 HLA 之间对应的标量数据类型	718
15.3.7 在 Delphi/Kylix 和 HLA 代码之间传递字符串数据	720
15.3.8 在 HLA 和 Kylix 之间传递记录数据	722
15.3.9 在 Delphi/Kylix 和 HLA 之间传递集合数据	726
15.3.10 在 HLA 和 Delphi/Kylix 之间传递数组数据	727
15.3.11 从 HLA 代码中引用 Delphi/Kylix 对象	727
15.4 使用 C/C++和 HLA 编程	730
15.4.1 链接 HLA 模块和 C/C++程序	731
15.4.2 寄存器保存	734
15.4.3 函数结果	734
15.4.4 调用惯例	734
15.4.5 C/C++中的值传递和引用传递	738
15.4.6 C/C++和 HLA 之间的标量数据类型一致性	738
15.4.7 在 C/C++和 HLA 代码之间传递字符串数据	740
15.4.8 在 HLA 和 C/C++之间传递记录/结构数据	740
15.4.9 在 HLA 和 C/C++之间传递数组数据	742
15.5 更多信息	742
附录 A ASCII 字符集	743
附录 B 80x86 指令集	747