

《汇编语言》

图书基本信息

书名：《汇编语言》

13位ISBN编号：9787302345923

10位ISBN编号：7302345929

出版时间：2014-2

出版社：清华大学出版社

作者：Jeff Duntemann

页数：567

译者：梁晓晖

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

《汇编语言》

内容概要

《汇编语言：基于linux环境（第3版）》是风靡美国的经典汇编语言畅销书籍的最新版，美国计算机领域著名作者jeffduntemann的力作。作者以其渊博的专业知识，丰富的实战经验，结合生动详尽的实例，全面系统地介绍了linux环境下如何使用汇编语言进行程序设计以及与之有关的背景知识和相关工具的使用。本书写作风格独特，全书采用作者最有特色的对话式风格，结合大量源于生活的暗喻，将晦涩难懂的知识点条分缕析地呈现出来，以便读者能以轻松愉快的心情学习。

《汇编语言：基于linux环境（第3版）》适合刚涉足linux环境下汇编语言的读者，也可作为相关技术人员的参考书。

《汇编语言》

作者简介

jeff duntemann，从事计算机相关文章和书籍的写作三十余年，主题涉及编程、无线网络和系统管理等。他曾担任过知名it杂志dr.dobb's的专栏作家，历任很多知名电脑编程杂志的编辑，在工作之余，他爱好天文和无线电，也喜欢写博客和科幻小说。

书籍目录

目录

第1章 又一个令人愉快的星期六

1

1.1 一切尽在计划之中

1

1.1.1 步骤和测试

2

1.1.2 不止两种方式

3

1.1.3 计算机像我们一样思考

4

1.2 如果这是真实情况

4

1.3 此路不通，请绕行

5

1.3.1 Big Bux游戏

6

1.3.2 玩Big Bux游戏

8

1.4 像棋盘游戏一样的汇编语言编程

9

1.4.1 代码和数据

10

1.4.2 地址

11

1.4.3 隐喻，将军

12

第2章 外星基数

14

2.1 新数学怪物归来

14

2.2 在火星上计数

15

2.2.1 火星数字剖析

17

2.2.2 数字基数的本质

19

2.3 八进制：绿色精怪怎样偷走8和9的

19

2.4 十六进制：解决数字的短缺

23

2.5 从十六进制到十进制，从十进制到十六进制

27

2.5.1 从十六进制到十进制

27

2.5.2 从十进制到十六进制

28

2.5.3 练习！练习！再练习	
30	
2.6 十六进制运算	
31	
2.6.1 列和进位	
34	
2.6.2 减法和借位	
34	
2.6.3 跨多列借位	
36	
2.6.4 意义何在	
37	
2.7 二进制	
37	
2.7.1 二进制值	
39	
2.7.2 为什么使用二进制	
41	
2.8 二进制简写方式：十六进制	
42	
第3章 摘下面具	
44	
3.1 RAXie，我们怎么不认识你	
44	
3.2 开关、晶体管和存储器	
46	
3.2.1 如果走陆路，就是1	
47	
3.2.2 晶体管开关	
47	
3.2.3 难以置信的位缩水	
49	
3.2.4 随机访问	
51	
3.2.5 存储器访问时间	
52	
3.2.6 字节，字，双字，四字	
53	
3.2.7 精致的芯片排成一行	
54	
3.2.8 车间工长和流水线	
57	
3.2.9 对话内存	
57	
3.2.10 驾驭数据总线	
58	
3.2.11 车间工长的口袋	
59	
3.2.12 流水线	

60	
3.3 遵循计划行事的盒子	
60	
3.3.1 取指和执行	
61	
3.3.2 车间工长的内脏	
62	
3.3.3 改变航向	
64	
3.4 是什么vs.怎么做：体系结构和微体系结构	
65	
3.4.1 体系结构的演变	
66	
3.4.2 地下室里的秘密机制	
67	
3.5 工厂经理	
68	
3.5.1 操作系统：角落办公室	
69	
3.5.2 BIOS：是软件，但并不软	
69	
3.5.3 多任务魔术	
70	
3.5.4 内核提升	
71	
3.5.5 内核爆炸	
73	
3.5.6 计划	
73	
第4章 位置，位置，位置	
74	
4.1 内存模式的乐趣	
74	
4.1.1 16 位将带来64K存储空间	
75	
4.1.2 兆字节的本质	
79	
4.1.3 向后兼容和虚拟86模式	
80	
4.1.4 16位眼罩	
80	
4.2 段的本质	
81	
4.2.1 一个界限，而非一个位置	
84	
4.2.2 用两个16位寄存器构成20位地址	
84	
4.3 16位和32位寄存器	
87	

4.3.1 通用寄存器	88
4.3.2 半寄存器	90
4.3.3 指令指针寄存器	91
4.3.4 标志寄存器	92
4.4 三种主要的汇编编程模型	93
4.4.1 实模式平面模型	93
4.4.2 实模式段模型	95
4.4.3 保护模式平面模型	97
4.5 保护模式下不再允许我们做的事情	100
4.5.1 内存映射视频系统	100
4.5.2 直接访问端口硬件	101
4.5.3 直接调用BIOS	102
4.6 展望未来：64位“长模式”	102
第5章 汇编的权利	105
5.1 文件及其包含的内容	106
5.1.1 二进制文件 vs. 文本文件	106
5.1.2 用 Bless 编辑器查看文件内容	108
5.1.3 解释原始数据	112
5.1.4 “字节序”	113
5.2 文本进去, 代码出来	116
5.2.1 汇编语言	117
5.2.2 注释	119
5.2.3 当心“只写”源代码	120
5.2.4 目标代码和连接器	120
5.2.5 重定位能力	

123	
5.3 汇编语言开发过程	124
5.3.1 工作目录规范	125
5.3.2 编辑源代码文件	126
5.3.3 编译源代码文件	126
5.3.4 汇编错误	127
5.3.5 回到编辑器	129
5.3.6 汇编警告	129
5.3.7 连接目标代码文件	130
5.3.8 连接错误	131
5.3.9 测试.EXE文件	131
5.3.10 错误vs.漏洞	132
5.3.11 我们还在那里吗	133
5.3.12 调试器和调试	133
5.4 沿着汇编小路旅行	134
5.4.1 安装软件	135
5.4.2 第1步：在编辑器中编辑程序	137
5.4.3 第2步：使用NASM编译程序	138
5.4.4 第3步：使用LD连接器	140
5.4.5 第4步：测试可执行文件	141
5.4.6 第5步：在调试器中观察程序运行	142
5.4.7 准备好要来真格的了吗	148
第6章 有地儿，有工具	149
6.1 Kate 编辑器	151
6.1.1 安装Kate编辑器	151

6.1.2 启动Kate	152
6.1.3 配置	154
6.1.4 Kate 会话	156
6.1.5 Kate编辑器的文件管理	158
6.1.6 向工具栏添加菜单项	161
6.1.7 Kate编辑器的编辑控制	162
6.1.8 编程期间使用Kate编辑器	166
6.2 Linux 和终端	169
6.2.1 Linux 控制台	169
6.2.2 Konsole中的字符编码	170
6.2.3 三个标准Unix文件	172
6.2.4 I/O 重定向	173
6.2.5 简易文本过滤器	175
6.2.6 使用转义序列进行终端控制	177
6.2.7 为什么不用汇编语言编写GUI应用程序呢	178
6.3 使用Linux Make	179
6.3.1 依赖条件	180
6.3.2 文件何时最新	182
6.3.3 依赖链	182
6.3.4 从Kate编辑器内部调用Make实用工具	184
6.3.5 使用touch命令强制执行生成操作	187
6.4 Insight 调试器	187
6.4.1 运行Insight	188
6.4.2 Insight的众多窗口	189
6.4.3 快速体验Insight	

190	
6.4.4 拿起你的工具	
193	
第7章 跟踪指令	
194	
7.1 为自己建立一个沙盒	
194	
7.1.1 一个最小的NASM程序	
195	
7.1.2 指令及其操作数	
197	
7.1.3 源操作数和目标操作数	
197	
7.1.4 立即数	
198	
7.1.5 寄存器数据	
200	
7.1.6 内存数据	
202	
7.1.7 混淆数据和它的地址	
203	
7.1.8 内存数据的尺寸	
204	
7.1.9 糟糕的过去	
204	
7.2 CPU的标志位	
205	
7.2.1 标志规范	
208	
7.2.2 使用INC指令和DEC指令加1和减1	
208	
7.2.3 从Insight中观察标志	
209	
7.2.4 标志如何改变程序的执行	
211	
7.3 有符号值和无符号值	
214	
7.3.1 补码和NEG	
214	
7.3.2 符号扩展和MOVSX	
217	
7.4 隐式操作数和Mul	
219	
7.4.1 MUL 和进位标志	
221	
7.4.2 使用DIV实现无符号除法	
221	
7.4.3 x86中的“慢动作”指令	
223	

7.5 阅读和使用汇编语言参考资料

223

7.5.1 对于复杂记忆的唤醒文件

224

7.5.2 初学者汇编语言参考指南

224

7.5.3 标志

225

7.6 NEG : 求补(求补码; 即, 与-1相乘)

225

7.6.1 合法形式

227

7.6.2 操作数符号

227

7.6.3 示例

228

7.6.4 注解

228

7.6.5 这里没有包含的内容

229

第8章 我们的崇高目标

230

8.1 汇编语言程序的基本框架

230

8.1.1 最开始处的注释块

232

8.1.2 .data段

233

8.1.3 .bss?段

233

8.1.4 .text段

234

8.1.5 标号

234

8.1.6 已初始化变量

235

8.1.7 字符串变量

235

8.1.8 通过EQU和\$推导字符串的长度

237

8.2 通过堆栈实现后进先出

239

8.2.1 每小时500个盘子

239

8.2.2 堆栈的内容上下颠倒

241

8.2.3 Push-y指令

242

8.2.4 POP 指令

244	
8.2.5 临时存储	
246	
8.3 通过INT80使用Linux内核服务	
247	
8.3.1 不中断任何事情的中断	
247	
8.3.2 再次返回	
252	
8.3.3 通过使用 INT 80h退出一个程序	
253	
8.3.4 软件中断VS硬件中断	
254	
8.3.5 INT 80h和可移植性盲目崇拜	
255	
8.4 设计一个有价值的程序	
256	
8.4.1 问题定义	
257	
8.4.2 从伪代码开始	
258	
8.4.3 连续改进	
259	
8.4.4 不可避免的“哎呀！”时刻	
263	
8.4.5 扫描缓冲区	
264	
8.4.6 缓冲溢出(“Off By One”)错误	
266	
8.4.7 进一步学习	
271	
第9章 位、标志、分支和表	
272	
9.1 位就是二进制位(字节也是二进制位)	
272	
9.1.1 位编号	
273	
9.1.2 逻辑操作	
273	
9.1.3 与指令	
274	
9.1.4 位屏蔽	
275	
9.1.5 或指令	
276	
9.1.6 异或指令	
276	
9.1.7 非指令	
278	

9.1.8 段寄存器对逻辑操作没有反应	278
9.2 移位操作	279
9.2.1 根据什么进行移位操作	279
9.2.2 移位指令的工作原理	279
9.2.3 将位移入进位标志	280
9.2.4 循环移位指令	280
9.2.5 将已知值存入进位标志CF	282
9.3 位操作	282
9.3.1 将一个字节分解为两个“半字节”	285
9.3.2 将高半字节移入低半字节	286
9.3.3 使用查找表	286
9.3.4 通过移位和相加来实现相乘	288
9.4 标志、测试和分支	291
9.4.1 无条件转移	292
9.4.2 条件转移指令	292
9.4.3 条件“缺席”时进行跳转	293
9.4.4 标志	294
9.4.5 通过CMP进行比较操作	295
9.4.6 转移指令的错综复杂之处	296
9.4.7 “大于”与“以上”	296
9.4.8 使用TEST指令查找位1	298
9.4.9 使用BT指令查找位0	300
9.5 保护模式下内存寻址详解	301
9.5.1 有效地址计算	303
9.5.2 位移量	

303	
9.5.3 基址+位移量寻址方式	
304	
9.5.4 基址+索引寻址方式	
304	
9.5.5 索引 × 缩放比例+位移量寻址方式	
305	
9.5.6 其他寻址方式	
307	
9.5.7 LEA：最机密的数学机器	
309	
9.5.8 16位寄存器的负担	
311	
9.6 字符表转换	
312	
9.6.1 转换表	
312	
9.6.2 用MOV或者XLAT进行转换	
315	
9.7 用表来代替计算	
319	
第10章 分治	
321	
10.1 盒子里面的盒子	
322	
10.2 调用和返回	
331	
10.2.1 调用中的调用	
334	
10.2.2 意外递归的危险	
335	
10.2.3 一个需要提防的标志规范Bug	
336	
10.2.4 过程及其所需的数据	
337	
10.2.5 保存主调程序的寄存器	
338	
10.2.6 局部数据	
341	
10.2.7 更多的表格技巧	
342	
10.2.8 在过程定义中加入常量数据	
344	
10.3 局部标号和跳转的长度	
345	
10.3.1 “强行”访问局部标号	
348	
10.3.2 短转移、近转移和远转移	
349	

10.4 生成外部过程库	350
10.4.1 全局声明和外部声明	351
10.4.2 全局过程和外部过程的机制	353
10.4.3 连接库文件到程序中	361
10.4.4 太多过程和太多库的危险	362
10.5 自定义过程的艺术	362
10.5.1 可维护性和可重用性	363
10.5.2 确定哪些代码应该成为一个过程	364
10.5.3 使用注释标头	365
10.6 Linux控制台下的简单光标控制	366
10.7 创建和使用宏	374
10.7.1 宏定义机制	375
10.7.2 定义带参数的宏	380
10.7.3 宏调用机制	382
10.7.4 宏内部的局部标号	383
10.7.5 像包含文件一样的宏库文件	384
10.7.6 宏?vs.?过程：优点和缺点	385
第11章 字符串奏鸣曲	387
11.1 汇编语言字符串的概念	387
11.1.1 彻底颠覆你的“字符串感觉”	388
11.1.2 源字符串和目标字符串	388
11.1.3 虚拟文本显示屏	389
11.2 REP STOSB，软件机枪	397
11.2.1 机枪扫射虚拟显示器	397
11.2.2 执行STOSB 指令	

398	
11.2.3 STOSB 和方向标志(DF)	
399	
11.2.4 在显示缓冲区中定义行	
400	
11.2.5 将缓冲区发送到Linux控制台	
401	
11.3 半自动武器：不带REP的STOSB	
401	
11.3.1 是谁递减了ECX	
402	
11.3.2 LOOP指令	
402	
11.3.3 在屏幕上显示一个标尺	
403	
11.3.4 MUL并非 IMUL	
404	
11.3.5 添加ASCII数字	
406	
11.3.6 调整AAA	
408	
11.3.7 Ruler过程的教训	
409	
11.3.8 STOS指令的16位版本和32位版本	
409	
11.4 MOVSB：快速块拷贝	
409	
11.4.1 DF和重叠块移动	
411	
11.4.2 使用Insight单步调试REP字符串指令	
413	
11.5 将数据存储到不连续的字符串中	
414	
11.5.1 显示一个ASCII表	
414	
11.5.2 嵌套指令循环	
416	
11.5.3 当ECX变为0时进行跳转	
416	
11.5.4 关闭内层循环	
417	
11.5.5 关闭外层循环	
418	
11.5.6 Showchar小结	
419	
11.6 命令行参数和堆栈检查	
419	
11.6.1 两块虚拟内存	
420	

11.6.2 Linux堆栈剖析	422
11.6.3 为什么堆栈的地址是不可预测的	424
11.6.4 使用Insight设置命令行参数	424
11.6.5 通过Insight的内存视图查看堆栈	425
11.7 使用SCASB进行字符串搜索	427
11.7.1 REPNE v.s. REPE	431
11.7.2 从堆栈中弹出，还是对堆栈寻址	432
11.7.3 额外的学分	434
第12章 C语言	435
12.1 什么是GNU	436
12.1.1 “瑞士军刀”编译器	437
12.1.2 以GNU的方式生成代码	437
12.1.3 如何在汇编工作中使用gcc	439
12.1.4 为什么不用gas	440
12.2 连接到标准的C函数库	441
12.2.1 C调用公约	442
12.2.2 建立一个框架	443
12.2.3 保存和恢复寄存器	443
12.2.4 建立堆栈帧	444
12.2.5 销毁堆栈帧	446
12.2.6 通过puts()输出字符	447
12.3 使用printf()格式化文本输出	448
12.4 使用fgets()和scanf()进行数据输入	452
12.5 驾驭时间	459
12.5.1 C库的时间机制	

459
12.5.2 从系统时钟中取出time_t值
461
12.5.3 将time_t 值转换为一个格式化字符串
461
12.5.4 生成单独的本地时间值
462
12.5.5 通过使用MOVSD复制glibc的tm结构
463
12.6 理解 AT&T 指令助记符
467
12.6.1 AT&T助记符公约
467
12.6.2 查看gcc创建的gas源文件
468
12.6.3 AT&T 内存引用语法
471
12.7 产生随机数
472
12.7.1 利用srand()为随机生成器“播种”
473
12.7.2 产生伪随机数
474
12.7.3 有些位比其他位更具随机性
479
12.7.4 调用寄存器中的地址
481
12.8 C如何看待命令行参数
482
12.9 简单文件输入/输出
484
12.9.1 通过sscanf()将字符串转换为数字
485
12.9.2 创建和打开文件
486
12.9.3 使用fgets()从文件中读取文本
488
12.9.4 使用fprintf()写文本到文件中
490
12.9.5 关于收集过程到库中的注解
492
结论：不是结束，而是刚刚开始
501
附录A 部分x86指令集
505
附录B 字符集图
565

《汇编语言》

精彩短评

- 1、每次拿起这本书，看着看着就生气，这书的节奏我是真的跟不上，呵呵。。。
- 2、2星给翻译的，原版节奏拖拉，读起来还行，翻译不知道是否体现了机器翻译的水平，无法阅读。感受一下“不是你不一定受它们愚弄，而是你已经不能受它们愚弄”
- 3、没有通用寄存器的详细介绍，没有堆栈帧的详细介绍，写的过于宽泛和简单。
- 4、强烈表达对翻译的不满！翻译梁晓晖！翻译给一星，原著不错！
- 5、从体系的高度讲解汇编，很好的一本书
- 6、文科生写给文科生看的汇编语言（非贬义）
- 7、内容结构不错，有几个词翻译的不是很好

1、用了四天来回上下班坐地铁的时间把本书看完了，因为已经工作了几年，汇编也有用到，所以看起来稍微会快一点。说说大体的感受吧。先说翻译，国内对外文技术图书的翻译可以说一直都不怎么样，国内技术类图书的翻译人员无外乎两类，一种是专门的翻译人员，精通外语，但是技术方面薄弱，因此翻译出来的东西技术人员是完全看不懂的，专业用词全给翻译成普通用法。第二类就是专业的技术人员，对技术很了解，知道书里说的是什么，但是对翻译却不是很擅长，结果出来的成果就是技术人员都能看，但是总觉得说话很别扭。本书大体上有点基础的人还是能看懂的，能知道作者说的是什么，但是中文读起来确实不流畅，所以这里给的经验就是，可以读读中译本，如果实在读不懂，还有比较重要的地方，去看英文原版吧，电驴上有。再说书本身。目前我见过写的比较好的汇编书籍一共有三本，王爽的《汇编语言》，罗云彬《win32汇编编程》和这本了。王爽的汇编语言讲的是8086体系，专注于语言本身，讲解思路很清晰。但是8086毕竟是存在于教科书上的东西，真正使用意义已经不大。《win32》那本书专注于系统编程，对汇编要求高一些，如果没有很强的汇编基础，建议这本书还是放一放。本书是讲解x86体系结构和汇编语言很好的入门书籍，同时又是基于linux操作系统的，补充了市面上缺少GNU方式汇编的空缺，而且使用了大量的图片来讲解概念，同时又将很多东西与实际生活联系起来，让读者很容易能理解设计的理念。如果你是个有基础的人，可能会觉得这样的写法很罗嗦，但是对于初学者，还是很有帮助的。本书的前半部分偏重于x86体系结构和linux环境的，而个人觉得本书的后半部分，也就是8至12章可以说是本书的精华，通过一个个有意义的代码程序来解释各种汇编语言的原理，而非跟很多编程书中只是写一下简单的毫无实际使用意义的用例。最后说说自己的想法。目前汇编语言已经是一个不太出现在媒体中的编程语言了，在这个被各种高级语言所包围的计算机世界中，汇编语言很低调的进行着自己的工作，默默的看着各种语言的发展。汇编语言目前所应用的领域都是比较专的，例如计算机安全，还有就是嵌入式中，嵌入式中各种芯片的启动都是由汇编完成的。这也就是说，目前对汇编的学习，我觉得只要能读懂，能过通过它能了解程序或硬件背后的机制和原理，就很不错了。除非你的领域非得用其不可，再用汇编去写程序，有点得不偿失了。即使现在用汇编比较多的嵌入式行业，汇编也只是芯片厂商会使用。应用生产的厂商拿到的一般是芯片厂家已经封装好的API，使用汇编对其更改的几率也不大。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu000.com