

<<x86汇编语言>>

图书基本信息

书名：<<x86汇编语言>>

13位ISBN编号：9787121187995

10位ISBN编号：712118799X

出版时间：2013-1

出版时间：电子工业出版社

作者：李忠

页数：360

字数：620000

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：<http://www.tushu007.com>

## <<x86汇编语言>>

### 内容概要

每一种处理器都有它自己的机器指令集，而汇编语言的发明则是为了方便这些机器指令的记忆和书写。尽管汇编语言已经较少用于大型软件程序的开发，但从学习者的角度来看，要想真正理解计算机的工作原理，掌握它内部的运行机制，学习汇编语言是必不可少的。

本书采用开源的NASM汇编语言编译器和VirtualBox虚拟机软件，以个人计算机广泛采用的Intel处理器为基础，详细讲解了Intel处理器的指令系统和工作模式，以大量的代码演示了16 / 32 / 64位软件的开发方法，介绍了处理器的16位实模式和32位保护模式，以及基本的指令系统。

这是一本有趣的书，它没有把篇幅花在计算一些枯燥的数学题上。相反，它教你如何直接控制硬件，在不借助于BIOS、DOS、Windows、Linux或者任何其他软件支持的情况下显示字符、读取硬盘数据、控制其他硬件等。本书可作为大专院校相关专业学生和计算机编程爱好者的教程。

## &lt;&lt;x86汇编语言&gt;&gt;

## 书籍目录

## 目 录

## 第1部分 预备知识

## 第1章 十六进制计数法 3

## 1.1 二进制计数法回顾 3

## 1.1.1 关于二进制计数法 3

## 1.1.2 二进制到十进制的转换 3

## 1.1.3 十进制到二进制的转换 4

## 1.2 十六进制计数法 4

## 1.2.1 十六进制计数法的原理 4

## 1.2.2 十六进制到十进制的转换 5

## 1.2.3 十进制到十六进制的转换 6

## 1.3 为什么需要十六进制 6

## 本章习题 7

## 第2章 处理器、内存和指令 8

## 2.1 最早的处理器的 8

## 2.2 寄存器和算术逻辑部件 8

## 2.3 内存储器 10

## 2.4 指令和指令集 11

## 2.5 古老的Intel 8086处理器 13

## 2.5.1 8086的通用寄存器 13

## 2.5.2 程序的重定位难题 14

## 2.5.3 内存分段机制 17

## 2.5.4 8086的内存分段机制 18

## 本章习题 21

## 第3章 汇编语言和汇编软件 22

## 3.1 汇编语言简介 22

## 3.2 NASM编译器 24

## 3.2.1 从网上下载NASM安装程序 24

## 3.2.2 安装NASM编译器 25

## 3.2.3 下载配书源码和工具 26

## 3.2.4 用Nasmide体验代码的书写和编译过程 28

## 3.2.5 用HexView观察编译后的机器代码 29

## 本章习题 30

## 第4章 虚拟机的安装和使用 31

## 4.1 计算机的启动过程 31

## 4.1.1 如何将编译好的程序提交给处理器 31

## 4.1.2 计算机的加电和复位 31

## 4.1.3 基本输入输出系统 32

## 4.1.4 硬盘及其工作原理 33

## 4.1.5 一切从主引导扇区开始 35

## 4.2 创建和使用虚拟机 35

## 4.2.1 别害怕, 虚拟机是软件 35

## 4.2.2 下载Oracle VM VirtualBox 36

## 4.2.3 安装Oracle VM VirtualBox 36

## &lt;&lt;x86汇编语言&gt;&gt;

- 4.2.4 创建一台虚拟PC 37
- 4.2.5 虚拟硬盘简介 42
- 4.2.6 练习使用FixVhdWr工具向虚拟硬盘写数据 43
- 第2部分 16位处理器下的实模式
- 第5章 编写主引导扇区代码 49
- 5.1 欢迎来到主引导扇区 49
- 5.2 注释 49
- 5.3 在屏幕上显示文字 50
- 5.3.1 显卡和显存 50
- 5.3.2 初始化段寄存器 52
- 5.3.3 显存的访问和ASCII代码 53
- 5.3.4 显示字符 55
- 5.4 显示标号的汇编地址 56
- 5.4.1 标号 56
- 5.4.2 如何显示十进制数字 60
- 5.4.3 在程序中声明并初始化数据 61
- 5.4.4 分解数的各个数位 61
- 5.4.5 显示分解出来的各个数位 65
- 5.5 使程序进入无限循环状态 66
- 5.6 完成并编译主引导扇区代码 67
- 5.6.1 主引导扇区有效标志 67
- 5.6.2 代码的保存和编译 68
- 5.7 加载和运行主引导扇区代码 68
- 5.7.1 把编译后的指令写入主引导扇区 68
- 5.7.2 启动虚拟机观察运行结果 70
- 5.7.3 程序的调试 70
- 本章习题 71
- 第6章 相同的功能, 不同的代码 72
- 6.1 代码清单6-1 72
- 6.2 跳过非指令的数据区 72
- 6.3 在数据声明中使用字面值 72
- 6.4 段地址的初始化 73
- 6.5 段之间的批量数据传送 74
- 6.6 使用循环分解数位 75
- 6.7 计算机中的负数 77
- 6.7.1 无符号数和有符号数 77
- 6.7.2 处理器视角中的数据类型 80
- 6.8 数位的显示 82
- 6.9 其他标志位和条件转移指令 83
- 6.9.1 奇偶标志位PF 83
- 6.9.2 进位标志CF 83
- 6.9.3 溢出标志OF 84
- 6.9.4 现有指令对标志位的影响 84
- 6.9.5 条件转移指令 85
- 6.10 NASM编译器的\$和\$\$标记 87
- 6.11 观察运行结果 87
- 本章习题 88

## &lt;&lt;x86汇编语言&gt;&gt;

- 第7章 比高斯更快的计算 89
  - 7.1 从1加到100的故事 89
  - 7.2 代码清单7-1 89
  - 7.3 显示字符串 89
  - 7.4 计算1到100的累加和 90
  - 7.5 累加和各个数位的分解与显示 90
    - 7.5.1 堆栈和堆栈段的初始化 90
    - 7.5.2 分解各个数位并压栈 92
    - 7.5.3 出栈并显示各个数位 94
    - 7.5.4 进一步认识堆栈 95
  - 7.6 程序的编译和运行 96
  - 7.7 8086处理器的寻址方式 96
    - 7.7.1 寄存器寻址 96
    - 7.7.2 立即寻址 97
    - 7.7.3 内存寻址 97
  - 本章习题 101
- 第8章 硬盘和显卡的访问与控制 102
  - 8.1 本章代码清单 102
    - 8.1.1 本章意图 102
    - 8.1.2 代码清单8-1 103
  - 8.2 用户程序的结构 103
    - 8.2.1 分段、段的汇编地址和段内汇编地址 103
    - 8.2.2 用户程序头部 106
  - 8.3 加载程序(器)的工作流程 109
    - 8.3.1 初始化和决定加载位置 109
    - 8.3.2 准备加载用户程序 110
    - 8.3.3 外围设备及其接口 111
    - 8.3.4 I/O端口和端口访问 112
    - 8.3.5 通过硬盘控制器端口读扇区数据 114
    - 8.3.6 过程调用 116
    - 8.3.7 加载用户程序 121
    - 8.3.8 用户程序重定位 122
    - 8.3.9 将控制权交给用户程序 126
    - 8.3.10 8086处理器的无条件转移指令 126
  - 8.4 用户程序的工作流程 128
    - 8.4.1 初始化段寄存器和堆栈切换 128
    - 8.4.2 调用字符串显示例程 129
    - 8.4.3 过程的嵌套 130
    - 8.4.4 屏幕光标控制 131
    - 8.4.5 取当前光标位置 131
    - 8.4.6 处理回车和换行字符 132
    - 8.4.7 显示可打印字符 133
    - 8.4.8 滚动屏幕内容 134
    - 8.4.9 重置光标 134
    - 8.4.10 切换到另一个代码段中执行 135
    - 8.4.11 访问另一个数据段 135
  - 8.5 编译和运行程序并观察结果 135

## &lt;&lt;x86汇编语言&gt;&gt;

- 本章习题 136
- 第9章 中断和动态时钟显示 137
  - 9.1 外部硬件中断 137
    - 9.1.1 非屏蔽中断 138
    - 9.1.2 可屏蔽中断 138
    - 9.1.3 实模式下的中断向量表 140
    - 9.1.4 实时时钟、CMOS RAM和BCD编码 141
    - 9.1.5 代码清单9-1 145
    - 9.1.6 初始化8259、RTC和中断向量表 145
    - 9.1.7 使处理器进入低功耗状态 147
    - 9.1.8 实时时钟中断的处理过程 148
    - 9.1.9 代码清单9-1的编译和运行 150
  - 9.2 内部中断 150
  - 9.3 软中断 151
    - 9.3.1 常用的BIOS中断 151
    - 9.3.2 代码清单9-2 155
    - 9.3.3 从键盘读字符并显示 155
    - 9.3.4 代码清单9-2的编译和运行 155
- 本章习题 156
- 第3部分 32位保护模式
- 第10章 32位Intel微处理器编程架构 159
  - 10.1 IA-32架构的基本执行环境 164
    - 10.1.1 寄存器的扩展 162
    - 10.1.2 基本的工作模式 162
    - 10.1.3 线性地址 163
  - 10.2 现代处理器的结构和特点 164
    - 10.2.1 流水线 164
    - 10.2.2 高速缓存 165
    - 10.2.3 乱序执行 165
    - 10.2.4 寄存器重命名 166
    - 10.2.5 分支目标预测 167
  - 10.3 32位模式的指令系统 168
    - 10.3.1 32位处理器的寻址方式 168
    - 10.3.2 操作数大小的指令前缀 169
    - 10.3.3 一般指令的扩展 171
- 本章习题 174
- 第11章 进入保护模式 175
  - 11.1 代码清单11-1 175
  - 11.2 全局描述符表 175
  - 11.3 存储器的段描述符 177
  - 11.4 安装存储器的段描述符并加载GDTR 180
  - 11.5 关于第21条地址线A20的问题 182
  - 11.6 保护模式下的内存访问 184
  - 11.7 清空流水线并串行化处理器 188
  - 11.8 保护模式下的堆栈 189
    - 11.8.1 关于堆栈段描述符中的界限值 189
    - 11.8.2 检验32位下的堆栈操作 190

## &lt;&lt;x86汇编语言&gt;&gt;

11.9 程序的编译和运行	191
本章习题	191
第12章 存储器的保护	192
12.1 代码清单12-1	192
12.2 进入32位保护模式	192
12.2.1 话说mov ds,ax和mov ds,eax	192
12.2.2 创建GDT并安装段描述符	193
12.3 修改段寄存器时的保护	195
12.4 地址变换时的保护	197
12.4.1 代码段执行时的保护	197
12.4.2 堆栈操作时的保护	198
12.4.3 数据访问时的保护	200
12.5 使用别名访问代码段对字符排序	201
12.6 程序的编译和运行	203
本章习题	203
第13章 程序的动态加载和执行	204
13.1 本章代码清单	204
13.2 内核的结构、功能和加载	205
13.2.1 内核的结构	205
13.2.2 内核的加载	206
13.2.3 安装内核的段描述符	208
13.3 在内核中执行	211
13.4 用户程序的加载和重定位	213
13.4.1 用户程序的结构	213
13.4.2 计算用户程序占用的扇区数	215
13.4.3 简单的动态内存分配	216
13.4.4 段的重定位和描述符的创建	217
1	

## 章节摘录

版权页：插图：第13章 程序的动态加载和执行 像我一样，很多人在了解了保护模式的基本工作原理之后，会产生一个疑问。

那就是，所有的段在使用之前，都必须以描述符的形式在描述符表中进行定义，那么，像操作系统这样的软件，又怎么能够加载和执行其他各种用户程序呢？

毕竟，你并不知道这些程序都定义了哪些段，每个段是什么类型，有多长。

未必所有人都会产生这样的疑惑，但我确实算一个，可能我还不够聪明。

事实上，这仅仅是一层窗户纸，一旦捅破了，才发现原来竟是那么简单。

从某种意义上来说，保护模式的工作机制对用户程序的加载和执行非但没有增加困难，反而带来了很大的便利。

一套能够充分说明问题的例子需要很大的代码量，也许把本书的汉字都去掉，全部换成代码也不够。

不过，只要能说明问题，也不一定非得完善周全、面面俱到。

因此，本章中用于加载和处理用户程序的做法，不一定，甚至根本就不是操作系统采用的方法。

这一点，务必明了。

计算机硬件之上是软件。

软件分两个层次，一是操作系统，二是应用（用户）程序。

通常，用户程序只关心问题的解，就是采用各种算法来解决实际问题。

至于软件是怎么加载到内存的，怎么定位的，不是它所操心的事。

但是，它有义务提供一些必要的信息，来帮助操作系统将自己加载到内存中。

相反，操作系统则必须考虑采用什么方法来加载用户程序，并在适当的时候将处理器的执行流转移到用户代码中去。

同时，为了减轻用户程序的工作量，操作系统还应当管理硬件，并提供大量的例程供用户程序使用。

比如，显示一个字符串，就不要让用户自己来写代码了，直接调用操作系统的代码即可。

但操作系统和用户程序应当协商一种机制，让用户程序能够在使用这些例程时，不必考虑和关心它们的位置。

本章提供了一个小小的“操作系统”，因为当不起这么大的名称，所以叫“内核”或者“核心”。

即使是这样，它依然当不起，因为它实在是太简单了。

不过，也没有办法，就这么凑合着叫吧。

内核不能放到主引导扇区里，毕竟它都很大。

所以，计算机首先从主引导程序开始执行，主引导程序负责加载内核，并转交控制权。

然后，内核负责加载用户程序，并提供各种例程给用户程序调用。

提供给用户程序调用的例程也叫应用程序接口（Application Programming Interface，API），本章用简单的方法来允许用户程序使用API工作。

本章学习目标：1.了解保护模式是为操作系统提供的技术，并没有给普通应用程序的编程带来负担（这从本章的程序实例中就可以看出来）。

2.学习操作系统在保护模式下加载和重定位应用程序的一般原理，学习简单的内存动态分配，了解应用程序接口API的简单原理，学习字符串的比较算法。

3.学习若干x86处理器的新指令，包括bswap、cpuid、cmovcc、sgdt、movzx、movsx、cmpsb、cmpsw、cmpsd和xlat等。



编辑推荐

《x86汇编语言:从实模式到保护模式》主要讲述INTEL x86处理器的16位实模式、32位保护模式，至于虚拟8086模式，则是为了兼容传统的8086程序，现在看来已经完全过时，不再进行讲述。

《x86汇编语言:从实模式到保护模式》的特色之一是提供了大量典型的源代码，这些代码以及相配套的工具程序可以到书中指定的网站，或者电子工业出版社华信教育资源网搜索下载。

#### 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:<http://www.tushu007.com>