# 《汇编语言》实验报告03

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 班级 | 2022秋 | 实验日期 | 2022.10.14 | 实验成绩 |  |
| 姓名 | 黄勖 | 学号 | 22920212204392 | | |
| 实验名称 | 汇编语言第三次实验 | | | | |
| 实  验  目  的  、  要  求 | 1. 熟练使用Debug（更加推荐采用程序形式），理解数据在内存中的存放，并理解并练习各种寻址方式。 2. 学习汇编语言的基本指令，学会阅读汇编代码并动手尝试编写一些算法 3. 学习虚拟机的使用方法，了解BIOS的基础运行方式，尝试破解BIOS密码 | | | | |
| 实  验  内  容  、  步  骤  及  结  果 | 1. **在数据段中依次存入10H,11H,12H,13H,14H,15H,16H,17H，将其相加，并将结果存入DX寄存器。** 2. **在code（lab3.asm）中编写代码**    1. **首先在数据段中依次存入10H,11H,12H,13H,14H,15H,16H,17H**      * 1. **用SI源变址寄存器存放相对于DS段之源变址指针，初始化指向第一个数据（10H），CX存放循环次数（8），DX,AH初始化为0（AH是ax的高8位，而AL是ax的低8位）**      * 1. **用AL接收字节型数据（AH保持为0），然后将DX和AX相加，SI指向下一个数据，循环执行，最后DX即为累加的结果**      * 1. **最后编写退出程序指令**      1. **（编译连接过程省略）在debug中实际调试程序**     **得到答案DX = 009C**   1. **练习使用debug命令破解BIOS密码，写出自己对破解密码的理解。**   **1）首先为了保证实验的安全性，我在网络上查找到win7的32系统可以直接在命令行运行 debug，于是搜索下载了相关的镜像并在vmware中安装了虚拟机系统**  **[注]笔者使用32位旗舰版镜像:** [**http://www.msdnwogaosuni.com/win7/13765.html**](%20http:/www.msdnwogaosuni.com/win7/13765.html)    ***图2-1 安装系统镜像的截图***    ***图2-2 安装win7x32完毕（好古早）***  **2）启动win7系统后，再选择顶部菜单启动按键，向下的三角形--打开电源时进入固件，即可自动启动虚拟机并进入VMware虚拟机BIOS设置界面。**    ***图2-3 vmware进入BIOS的操作图示***    ***图2-4 BIOS界面***  **3）用键盘操作在BIOS中的Security→Set Password设置管理员密码与普通用户密码**    ***图2-5 BIOS设置密码的界面***  **4）重启系统并再次进入BIOS界面，提示要输入密码（此时假设不知道密码，则无法进入BIOS系统）**    ***图2-6 进入BIOS提示输入密码***  **5）经过查阅资料，此时需要进入纯DOS状态利用debug代码破解BIOS代码，在此处我利用maxdos工具箱软件在系统引导中添加进入纯dos系统环境的选项**  **[注]笔者使用工具地址**[**https://www.xitongzhijia.net/soft/94105.html?1643103248**](https://www.xitongzhijia.net/soft/94105.html?1643103248)    ***图2-7 MaxDOS控制台界面***    ***图2-8 利用vmtools将文件传入虚拟机***  **6）此时需要进入纯DOS系统**    ***图2-9 系统引导界面-进入MaxDOS***    ***图2-10 选择进入MaxDOS模式***    ***图2-11 现在就可以选择进入纯DOS环境了***    ***图2-12 纯正DOS界面***  **7）此时启动debug并运行破解BIOS密码指令**  **输入以下破解程序：**  **debug**  **- O 70 10**  **- O 71 ff**  **- q**    ***图2-13 输入指令***  **8）重启，进入BIOS界面，发现密码已被破解，可以直接进入并修改BIOS配置！**    ***图2-14 破解成功画面***   1. **（选做题） 在长度为8的字节数组（无符号数）中，查找大于42H的无符号数的个数，存放在字节单元up中；等于42H的无符号数的个数，存放在字节单元equa中；小于42H的无符号数的个数，存放在字节单元down中。程序显示up equa down的值。**   **八个数：31H,21H,42H,52H,87H,23H,98H,01H**   * 1. **在数据段中存入八个数，为up,equa,down分配内存并初始化为0**      * 1. **初始化，把数组数据和42H进行比较，如果相等就转移到EQUAL\_NUMBER，如果小于42H就转移到DOWN\_NUMBER，如果大于42H就转移到UP\_NUMBER，并利用count变量判断是否已经计数完毕（如果count为0，说明比较完转移到RESULT，否则转移到COMPARE）**     **对每次的比较结果将每次的计数结果变量对应加一**       * 1. **输出结果，显示字符串upString，调用功能号02H，显示字符**      * 1. **退出程序**      * 1. **连接编译运行查看结果，得到up为3，equa为1，down为4，结果正确！**     **\*(四)遇到的问题**  **1）在尝试破解密码的时候，由于对教程的错误理解在win7的cmd窗口中执行dos指令，最后没有效果。（花了很长时间）**  **错误操作如下图：**          **解决办法：在实际实验过程中，进入纯dos系统操作则成功。**  **2）汇编语言中,为什么SI和DI不能同时使用汇编**  **解决：见总结。**   1. **破解BIOS密码的原理？**   **解决：见总结。** | | | | |
| 总  结 | 这一次实验我对汇编语言指令和BIOS有了更深的理解，并且这一次实验的实践操作颇丰，在练习编码和输入指令的过程中我对每一个指令的用途和用法有了更深的认识，通过一步步地解决问题，我的实践能力提高了，这让我受益匪浅；具体遇到问题的解决方案我在后文做了更详细的总结，在此就不多赘述；在未来我还要探索汇编语言的更多应用方面，寻找更多问题，并在发现问题的过程中继续提高我对汇编语言的掌握能力，这是一次颇有意义的实验！  **问题总结：**  **一、破解BIOS密码的原理与理解？**  出现了-o 70 -o71，因为CMOS中数据访问是通过70和71这两个I/O端口来实现的。端口70H是一个字节的地址端口，用来设置CMOS中数据的地址，而端口71H则是用来读写端口70H设置CMOS地址中的数据单元内容。70写的是地址，71写的是数据，o是out，这两条指令会把20写到RTC地址为21的寄存器里面。RTC (Run Time Cloc)位于芯片，就是主板上的时钟，用一小块钮扣电池驱动，这块电池同时还一直在刷新一个256字节的小存储器，里面存放的就是CMOS里的数据。**这里往一个地址随意写一个值，会导致校验错误，BIOS在boot的时候遇到校验错误会load default，就是会恢复默认值，这样密码就没了**。同样的，拿掉CMOS电池并短接也可以导致RTC存储器里的数据丢失，校验基本上就错了，也会恢复默认值。  **二、汇编语言中,为什么SI和DI不能同时使用汇编**  这两个寄存器的意思，SI源变址寄存器，DI目地变址寄存器，既然是变址寄存器，那么他们肯定是在某个地址的基础上进行偏移变化，由此我们就得出了需要基址寄存器。  要是把这两个寄存器同时使用，那地址变化的基址都没有，该怎么变化呢？在谁的基础上变化（也就是地址偏移）？  对于汇编中的规定，其实有时并不需要书上详细的介绍，我们都应该可以从中推导出这些规则，书上的那些介绍只是用来验证我们的推测的。或是对我们所掌握的知识的进行检测，用来说明我们所掌握的是对的！  **三、寄存器的总结**  **1:数据寄存器,一般称之为通用寄存器组**  8086 有8个8位数据寄存器，  这些8位寄存器可分别组成16位寄存器：  AH&AL＝AX：累加寄存器，常用于运算；  BH&BL＝BX：基址寄存器，常用于地址索引；  CH&CL＝CX：计数寄存器，常用于计数；  DH&DL＝DX：数据寄存器，常用于数据传递。  **2:地址寄存器/段地址寄存器**  为了运用所有的内存空间，8086设定了四个段寄存器，专门用来保存段地址：  CS（Code Segment）：代码段寄存器；  DS（Data Segment）：数据段寄存器；  SS（Stack Segment）：堆栈段寄存器；  ES（Extra Segment）：附加段寄存器。  **3：特殊功能的寄存器**  IP（Instruction Pointer）：指令指针寄存器，与CS配合使用，可跟踪程序的执行过程；  SP（Stack Pointer）：堆栈指针，与SS配合使用，可指向目前的堆栈位置。  BP（Base Pointer）：基址指针寄存器，可用作SS的一个相对基址位置；  SI（Source Index）：源变址寄存器可用来存放相对于DS段之源变址指针；  DI（Destination Index）：目的变址寄存器，可用来存放相对于 ES 段之目的变址指针。 | | | | |