# 《汇编语言》实验报告2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 班级 | 2022秋 | 实验日期 | 2022.10.1 | 实验成绩 |  |
| 姓名 | 黄勖 | 学号 | 22920212204392 | | |
| 实验名称 | 汇编语言第二次实验 | | | | |
| 实  验  目  的  、  要  求 | 1）熟练掌握汇编语言程序的书写、汇编、连接等步骤  2）掌握基本的debug命令，并对程序进行基本的调试  3）学习32位的Intel x86的基本指令，学会阅读汇编代码并动手尝试编写一些算法  4）复习冒泡排序算法的程序结构并尝试运用 | | | | |
| 实  验  内  容  、  步  骤  及  结  果 | 1. **将给定程序输入，并汇编、连接后生成可执行文件lab2.exe**     *图1 – 输入的程序*    *图2 – 使用cd命令切换目录和MASM命令编译文件*    *图3 – 链接文件*    *图4 – 运行可执行文件*  **开始进行Debug lab2.exe**    *图5 – 执行debug指令*   1. **将内存中字符串“world”改写成“WORLD”，并显示修改后的结果**   ①使用D命令（Dump内存16进制显示）查看内存中的内容  查看指定地址的数据（默认显示128个内存单元）：d <段地址>:<偏移地址>    *图6 – 查看内存结果*  ②使用E命令（Enter修改内存字节）改写内存中字符串的内容    *图7 – 修改内存结果*  ③使用G命令（Go执行）继续执行程序，发现已经修改成功！    *图8 – 修改后运行结果*   1. **展示3F24+4A2B和3F24－4A2B的计算（用-h 和add/sub指令）**   **Ⅰ（方法1 – H指令）**H: 执行十六进制算术运算（Hexadecimal）    **格式：**H 值1 值2           值1、2为0—FFFFH范围内的任意十六进制数。    **用途：**用来求两个十六进制数的和、差，对结果显示为值1+值2及值1-值2。  如果值2 > 值1则显示其补码。    *图9 – 使用H命令得到的结果（前者为和后者为差）*  **Ⅱ（方法2 - add/sub指令）**A:输入汇编指令 T:执行CS:IP指向的指令   1. 利用R指令先查看各寄存器的内容     （可以看出原本AX内容为FFFF，BX内容为0000，IP指向0000）   1. 利用R指令修改ax,bx寄存器的内容，使得其中存储我们需要计算的数字     （如 -R ax可以显示ax当前值，在“：”后输入新值并回车，在“r ←”后查看修改结果，上图修改了ax与bx的值为需要计算的数字）   1. 利用A指令编写add指令，将ax与bx中的内容相加并存储到ax中      1. 利用T指令执行A指令编写的代码，可以查看到ax的内容变为894F，即相加的结果      1. 继续编写计算相减，首先继续重新赋值ax与bx      1. 利用A指令编写sub指令，将ax与bx中的内容相减并存储到ax中      1. 利用T指令执行A指令编写的代码，可以查看到ax的内容变为F4F9，即相减的结果     **(四)在内存中输入 MOV AX, 32H**  **ADD AX,AX**  **执行并查看AX的变化，修改AX的值为FFFF**   1. 首先利用A指令输入代码   （H表示16进制实际不需要输入，否则会报错）   1. 其次使用T指令执行两行代码 2. 第一次输入 mov指令将AX的值修改为32（十六进制）      1. 第二次输入 add指令将AX的值进行自增，结果为64（十六进制）      1. 最后利用R指令修改AX的值，实验结束     **（五）使用32位的Intel x86的指令，编写计算冒泡排序算法的程序（从小到大排序、从大到小排序）；并在32位的Intel x86汇编语言环境下运行通过**  算法设计：冒泡排序以往是在C语言中编写，当时的代码比较简单，编写出来也很易懂，只需要两层循环。在汇编中我的想法是使用loop对ecx进行操作外层循环，si=0作为每次内层循环的初始条件，cx初始值为内层的循环次数（即本轮的交换次数，即交换n-1次来做到内层依次排序）用si和cx值相等与否作为内层循环结束条件。   1. **从小到大**     *图9 – 从小到大的冒泡排序代码*  在debug前我们先需要编译和运行，由于报告前文以及简述其过程在这里便略过。  在debug中我们先查看初始的数据顺序（DS中）    *图10 – 从小到大的冒泡排序初始数据*  在运行程序后，再查看内存中的数据结果，发现已经从小到大排序完毕。    *图11 – 从小到大的冒泡排序结果*   1. 从大到小   方案与①类似，只需要将比较的条件转化为前面的数比后面小就交换。    *图12 – 从大到小的冒泡排序代码*  先查看初始的数据。    *图13 – 从大到小的冒泡排序初始数据*  排序完毕后再查看内存中的数据结果，发现已经从大到小排序完毕。    *图14 – 从大到小的冒泡排序结果*  **实验出现的问题：**   1. 主要出现的问题为在使用A指令时输入mov ax,32H发现报错     解决方法：H表示16进制实际不需要输入，否则会报错，最后只需要删除H即可成功输入汇编指令。  2）在设计冒泡排序算法的过程中，对于实际编写汇编语言代码遇到了很多困难，对于多种跳转和循环的指令还不是很熟悉使用方式，未来还需要多加练习。 | | | | |
| 总  结 | 在这一次实验中我对debug的使用有了更深的体会，并且我在实际操作中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，我对每一个指令的运行模式有了大致的框架，这让我受益匪浅；同时在编写冒泡排序的过程中我也对基本的汇编语言指令有了更深的认识，并且运用在具体的编码中；具体遇到问题的解决方案我已经附在实验内容中，在此就不多赘述；在未来我还要探索debug能够解决的更多问题，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！ | | | | |