**实验报告**

**实验序号： 实验项目名称： 编写0号中断的处理程序**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学　　号 |  | 姓 名 |  | 班 级 |  |
| 实验地点 |  | 指导教师 |  | 实验成绩 |  |
| **一、实验目的**  1）学会使用标志寄存器；  2）熟练使用通过检测比较结果来转移的转移指令。  **二、实验环境(设备**)  编辑器：Visual Studio 2022  汇编程序：MASM.EXE  调试程序：DEBUG.EXE  连接装配程序：LINK.EXE  **三、实验内容**  编写0号中断处理程序，使得在除法溢出时，在屏幕中间显示字符串'divide error!'，然后返回到DOS。  **四、实验结果及其分析**  **1.程序分析**  **（1）中断向量表**  **① 结构**  内存0000：0000~0000：03FF（大小为1KB）的空间是系统存放中断处理程序**入口地址**的中断向量表。8086CPU支持256个中断，但系统实际上要处理的中断事件远没有达到256个，所以在向量列表中有许多单元是空的。  **② 应用**  中断向量表只用来存放中断处理程序的入口地址，DOS系统和其他应用程序都不会随便使用这段空间。一般情况下，从**0000：0200~0000：02FF**的256个字节的空间所对应的中断向列表项都是**空**的，操作系统和其他应用都不会占用，因此可以利用中断向量表中的空闲单元来存放我想要运行的程序，使do0成为0号中断的处理程序。  **③ 程序实践**  为了实现该程序功能，我需要编写一个do0程序作为0号中断的中断处理程序并放到000：0200后。若要使得CPU在遇到除法溢出时就转去执行do0，必须将do0的入口地址0000：0200登记在中断向量表的对应表项中。因为除法溢出对应的中断类型码为0，中断处理程序的入口地址应该从0\*4地址单元开始存放（段地址存放在0\*4+2字单元中，偏移地址存放在0\*4字单元中）；也就说要将do0的**段地址0存放在0000：0002字单元**中，将**偏移地址200H存放在0000：0000字单元**中。  **（2）rep movsb指令**  在代码中，我使用了rep movsb指令把 ds:si 所指地址的一个字节搬移到 es:di 所指的地址上。其中，movsb是字符串传送指令，这条指令按字节传送数据，通过SI和DI这两个寄存器控制字符串的源地址和目标地址，比如DS:SI这段地址的N个字节复制到ES:DI指向的地址，复制后DS:SI的内容保持不变。而REP是重复前缀指令，因为既然是传递字符串，则不可能一个字（节）一个字（节）地传送，所以需要有一个寄存器（即CX）来控制串长度。  因此，在这段程序中，我通过设置CX的值为传输长度offset do0end - offset do0，指令每次执行前都会判断CX的值是否为0（为0结束重复，不为0，CX的值减1），以此来设定重复执行的次数,完成把 ds:si 所指地址的一个字节搬移到 es:di 所指的地址上。  **（3）cld指令**  CLD（CLear Direction flag）是清方向标志位，也就是使DF的值为0，在执行串操作时，使地址按递增的方式变化，这样便于调整相关段的的当前指针。  **2.代码与结果调试**  **（1）程序代码**  **1）编写可以显示divide error!的中断处理程序：do0;**    **2）将do0送入内存0000：0200处；**    **3）将do0的入口地址0000：0200存储在中断向量表0号表项中**    **（2）结果调试**  **1）DEBUG跟踪调试——验证0号中断处理程序是否成功编写**  ① 如下图，在程序运行前分别使用d 0:0 和 d 0:0200命令查看内存内容，用于和运行后的结果进行对比。    ② 如下图，首先使用G命令运行程序。之后，再分别使用d 0:0 和 d 0:0200命令查看内存内容，并与运行前的结果进行对比，可以看到：此时0号中断向量的指向改变了，指向了0000:0200H，且0000:0200H中存储的就是预期的“divide error！”    **2）编写测试程序——检查在除法溢出时，屏幕中间是否显示字符串‘divide error!'**  ① 程序代码如下：  该程序实现的是1000H/1H的除法计算，由于商1000H超过256，放进al寄存器时会产生溢出效果，如果前一步的0号中断处理程序正确编写运行，则会在屏幕中间显示字符串‘divide error!’。    ② 结果测试：  首先在程序运行前使用d 0:0 和 d 0:0200命令查看内存内容。可以看到，这两处的内存中正确显示预期内容：    使用G命令执行程序，可以看到由于程序中出现了出发错误，此时在屏幕中间显示了字符串‘divide error!’：    **五、实验心得与讨论**  通过这次实验，亲手尝试编写0号中断处理程序实现特定的功能，更好地理解了课本上的关于中断的知识点还有联系。尤其是比如，在上课的时候对于书上说的中断过程和程序安装有机联系起来，但是通过实际实操、DEBUG调试查看内存中的内容等，很好地帮助我了解中断处理程序运行机制，理解各种知识点之间的联系。 | | | | | |
| **评语：**  1．按时完成实验；  2．实验内容和过程记录完整；  3．有实验的心得或讨论；  **老师：xx** | | | | | |