

**汇编终结报告**

学院： 计算机科学与技术学院

专业： 计算机科学与技术

班级： 1617101

学号： 011730128

姓名： 邹 博 士

## 一、文件清单：

|  |  |  |
| --- | --- | --- |
| 文件名称 | 文件内容 | 备注 |
| loader.asm | 程序加载器 |  |
| Loader.com | 加载器转换的二进制文件 | 放在虚拟硬盘0号扇区 |
| Div\_Err.asm | 除法出错中断处理程序 |  |
| Div\_Err.com | 除法出错程序二进制文件 | 放在虚拟硬盘10号扇区 |
| KeyBoard\_Int.asm | 键盘中断处理程序 |  |
| KeyBoard\_Int.com | 键盘中断二进制文件 | 放在虚拟硬盘20号扇区 |
| Extend\_IO.asm | 扩展IO中断处理程序 |  |
| Extend\_IO.com | 二进制文件 | 放在虚拟硬盘100号扇区 |
| Program.asm | 第八章程序 |  |
| Program.com | 二进制文件 | 放在虚拟硬盘200号扇区 |
| Game.asm | “模拟枪战”小游戏 |  |
| Game.com | 小游戏对应的二进制文件 | 放在虚拟硬盘50号扇区 |

## 二、本次完成的功能：

1. 中断系统程序：
   1. 键盘中断处理程序：通过按下”R”，”G”，”B”，可以分别在DOS界面的中央处输出两行红色、绿色、蓝色的字符串

”Welcom To Exploring Assembler World !”,

”Believe Me,You Will Enjoy It!”

* 1. 除法中断处理程序：显示除法溢出的字符串，并返回到溢出之后的下一条指令继续执行。
  2. 扩展显示I/O程序：由自己手动设置更改90H号中断向量。实现在屏幕中央显示两行不同彩色背景的字符。

1. 加载第八章程序：

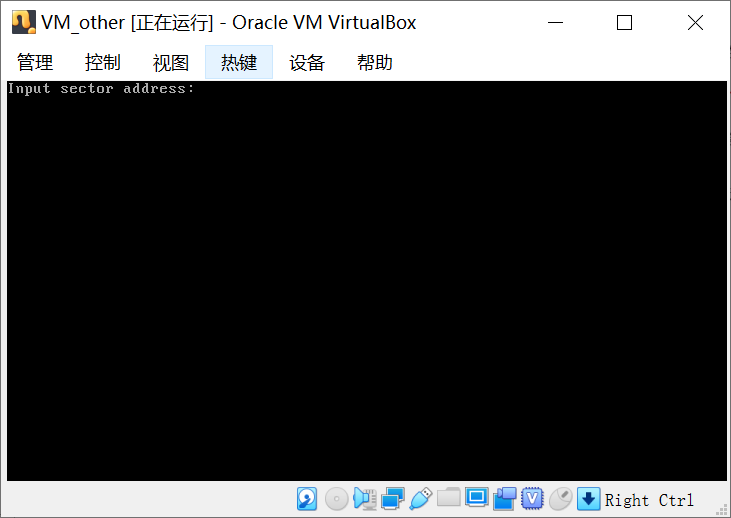
读取CMOS时钟模块信息，将系统时间实时显示在屏幕中央，并保持与系统时间同步，等待用户按下”#”键，退出该程序并返回加载器。

1. 用汇编语言编写的“模拟战争”小游戏

## 三、功能解释：

# 1.界面含义：

程序开始时，界面上会提示输入程序被写入的扇区号，从而跳转到对应的程序进行执行。

在刚开始编写课程设计的时候，也是做了一个菜单，通过输入数字来选择跳转的程序位置。由于在编写代码过程中，需要事先在加载器中固定数字与扇区号的对应关系，而每次修改文件或者增加文件都要重新修改一次加载器的代码，然后将加载器写入磁盘。因此，在调试过程中我将界面改为了输入扇区号来控制执行的程序，同时在这份报告的开头写明了各个程序对应的扇区号。

若输入的扇区号不合法，也会做出提示。

# 2.加载第八章程序

读取CMOS时钟模块信息，将系统时间实时显示在屏幕中央，并保持与系统时间同步，等待用户按下’#’键，退出该程序并返回加载器。

# 3.中断处理程序

在初始界面输入扇区号，并输入回车键，即可到达对应的中断处理程序界面。

1. 键盘中断处理程序

首先设置9号中断向量指向新的由自己编写的键盘中断处理程序，然后接受用户按键，直到用户按下回车键为止。

新的键盘中断处理程序：通过判断按键的扫描码，转换得到其对应的ASCII码，显示输出用户按下的字符，并在DOS界面中央输出相应颜色的字符串。

1. 除法出错中断处理程序

首先开中断，接着保护通用寄存器和段寄存器DS，在除法中断发生后，显示提示信息；最后恢复被保护的寄存器，利用中断返回指令IRET实现中断返回。

1. 扩展显示I/O程序

修改90H号中断处理程序，其功能是以TTY方式显示带属性的字符。

演示过程：首先工作程序设置90H中断向量，使其指向扩展的显示I/O程序，然后为了体现演示效果，依次在屏幕中央输出两行不同颜色的字符串！

"No.90H handler is ready!"

"Now You Can Explore The World!"

# 4.模拟枪战小游戏

模拟小时候科技不发达的时候玩的小游戏，画面只提示目前所处的状态，以及分数，以及进行的操作！

1.  按键盘上的W、A、S、D分别控制角色的跳上、左移、跳下、右移，分别显示”Jump up…”, “Move left…” , ”Jump down…”, “Move right…”

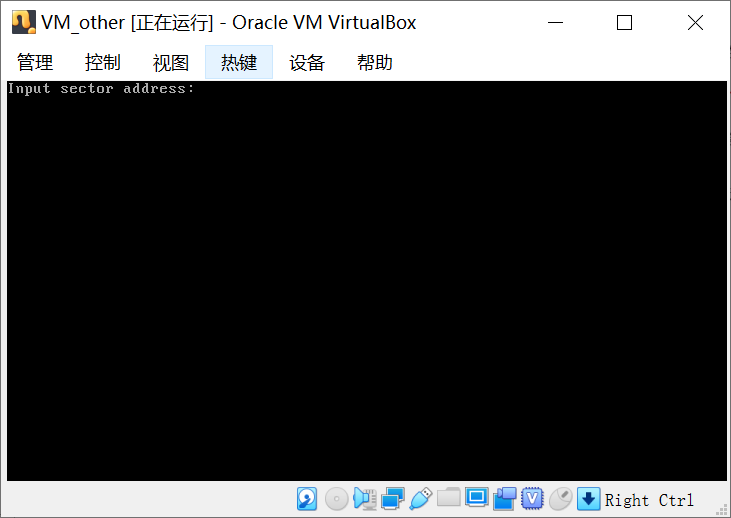
2.  按空格键开始射击, 显示”Shoot…”

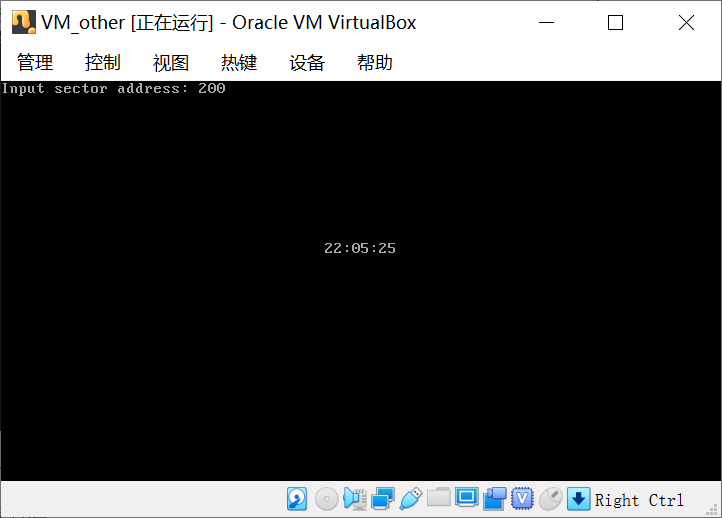
3. 按R键装填子弹，显示”Get bullet…”

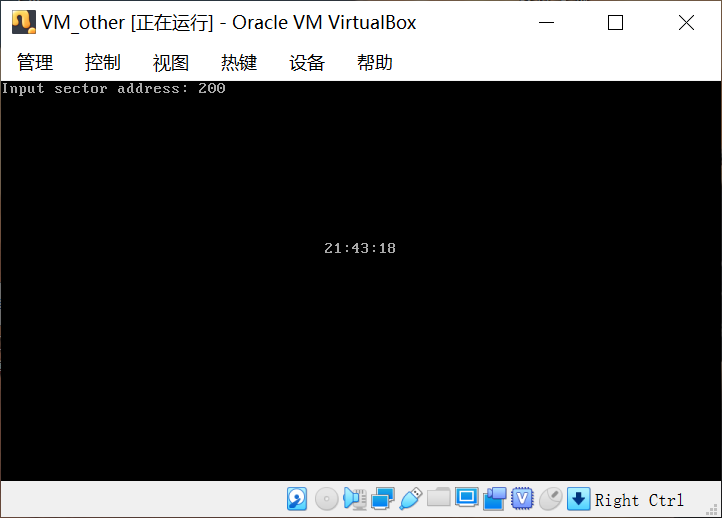
4.  按ESC键退出游戏，显示”Byebye…”,并退出程序，返回加载器

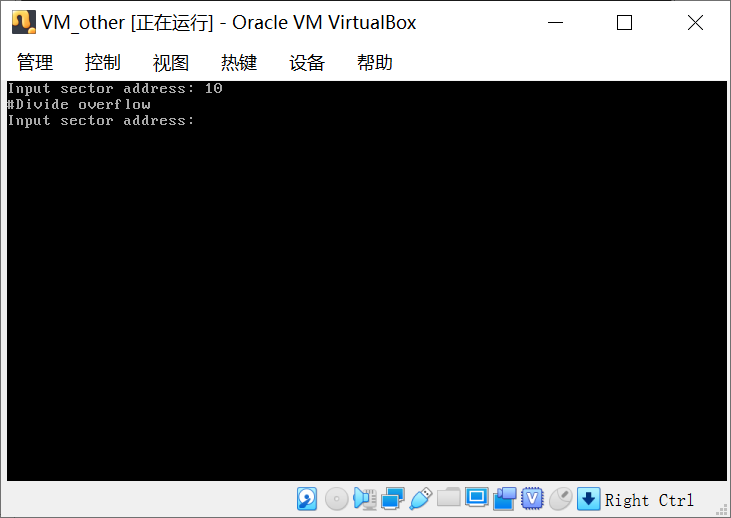
用户每次操作后会显示”Game is running…”，提示游戏正在进行

## 四、程序演示：

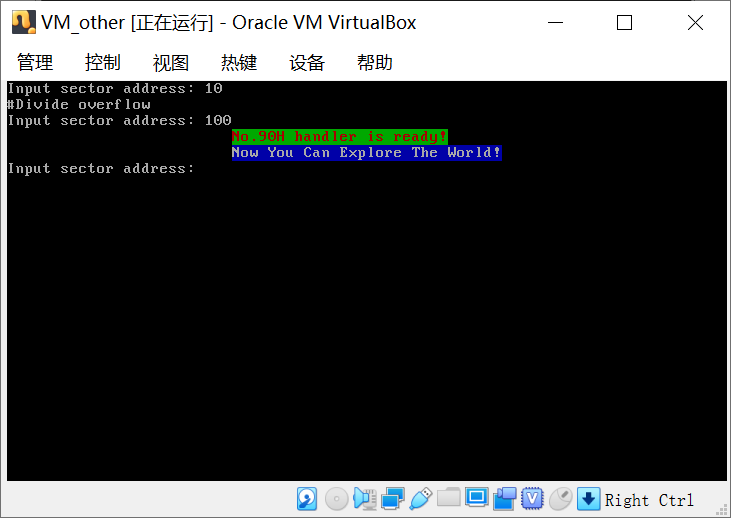
首先是初始界面：

1. 加载第八章程序，读取CMOS时钟模块信息，将系统时间实时显示在屏幕中央

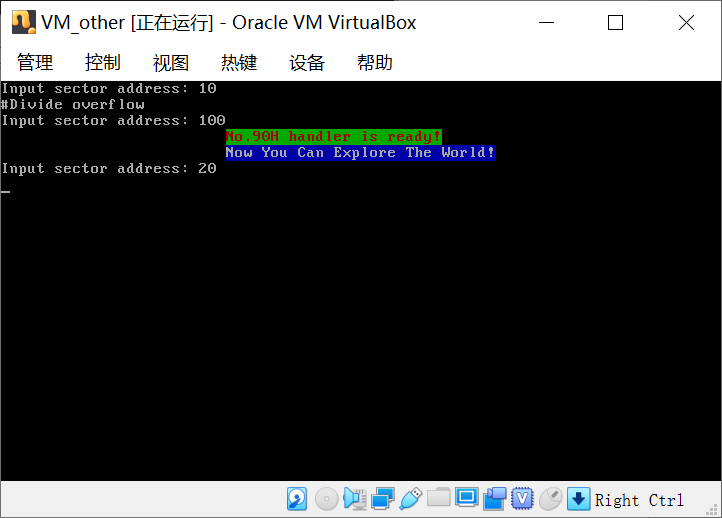


1. 演示中断处理程序
   1. 通过输入10，演示除法中断程序

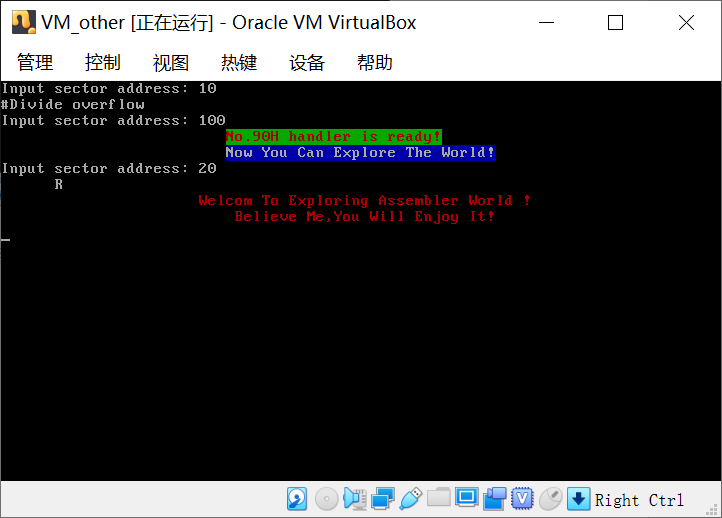
屏幕上输出 “Divide overflow”，并返回加载器

* 1. 接下来输入100，演示扩展IO中断程序

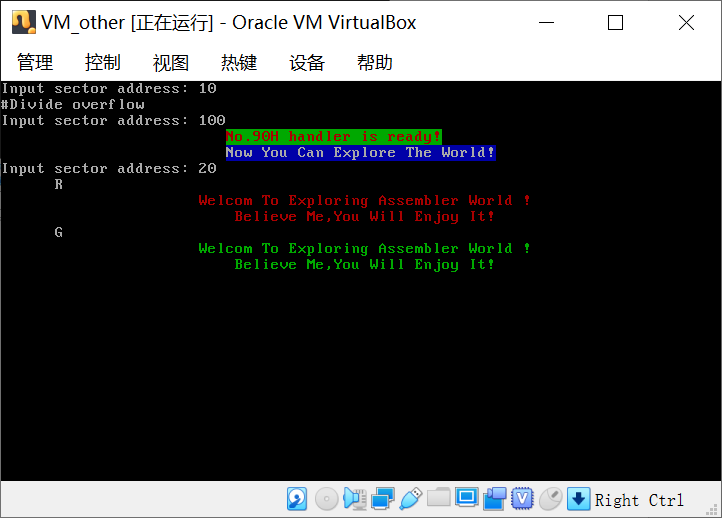
在屏幕的中央分别显示两行颜色不同的字符串

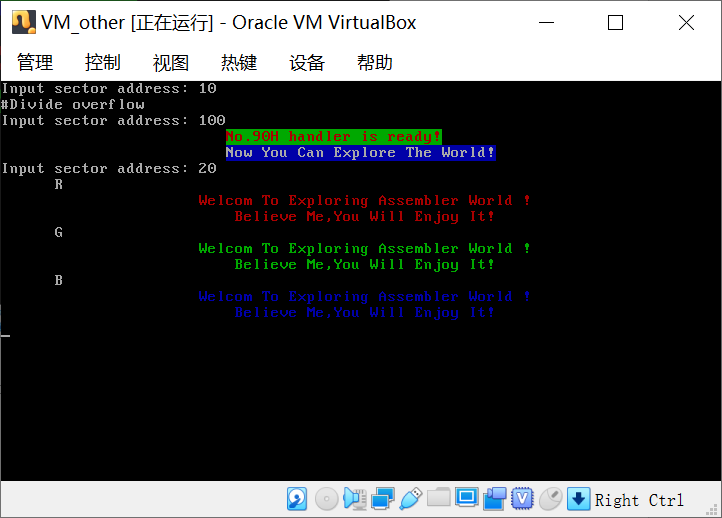
* 1. 接下来通过输入20，演示键盘中断程序

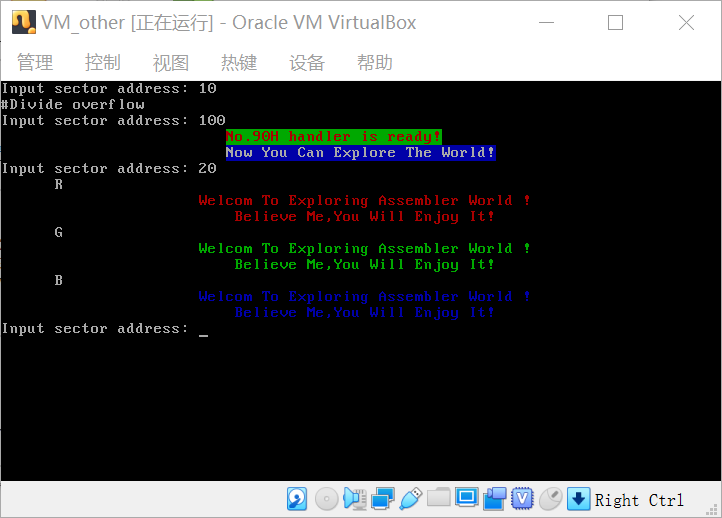
按下相应的键，根据按键扫描码确定输出何种颜色的字符串。当然也可以改为按下相应按键，根据扫描码修改DOS界面为对应的颜色，都是修改输出字符的属性值，只是显示的范围不同，原理是一样的，颜色值也可以调，其中02H代表黑底绿字、01H代表黑底蓝字、04H代表黑底红字。也试了修改为其他颜色或者背景色，发现不是很好看….就改回来了。

按下‘R’键，会在屏幕中央输出两行黑底红字的字符串

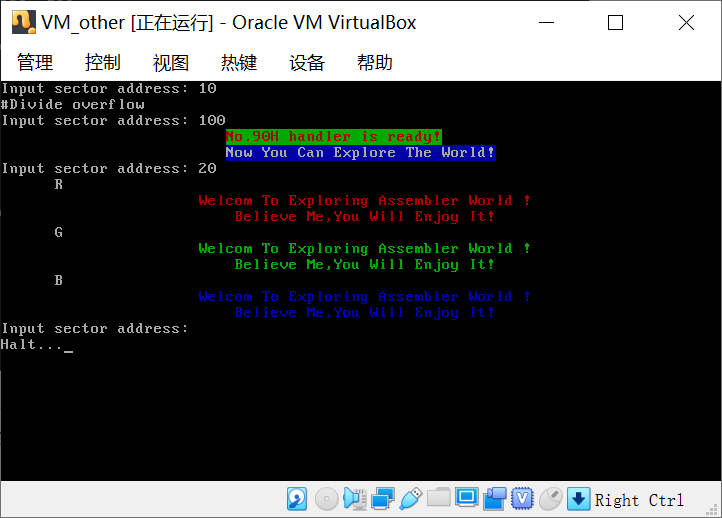
按下‘G’键，会在屏幕中央输出两行黑底绿字的字符串



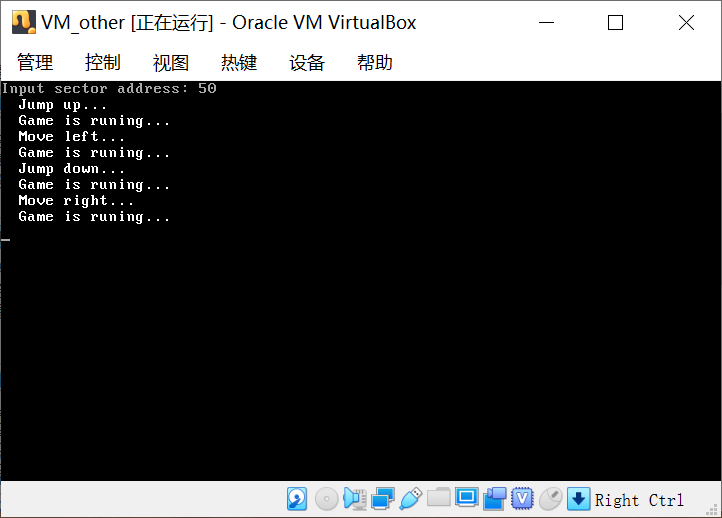
按下‘B’键，会在屏幕中央输出两行黑底蓝字的字符串

然后按回车键，可以返回加载器

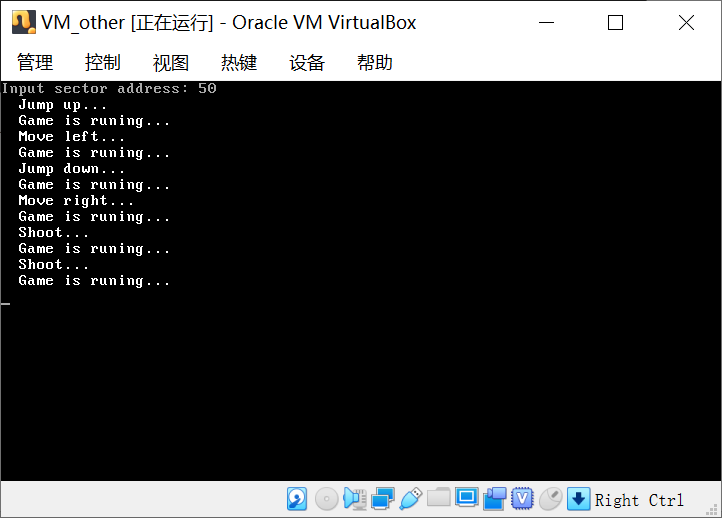
再次按下回车键即结束加载器



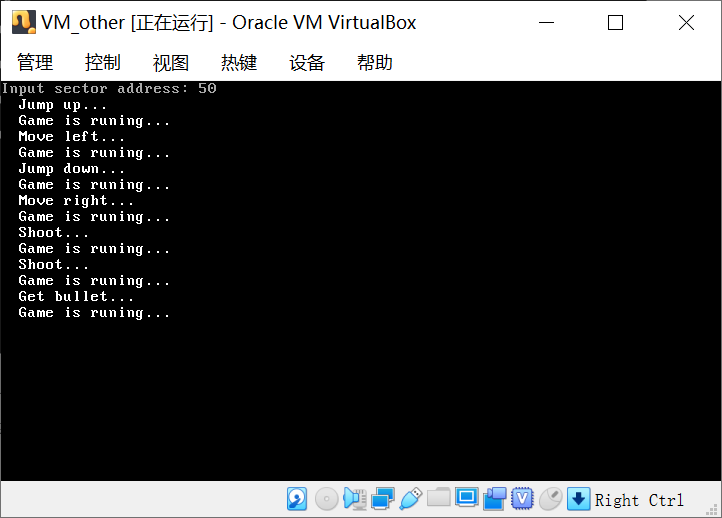
1. 汇编语言编写的模拟“枪战”小游戏

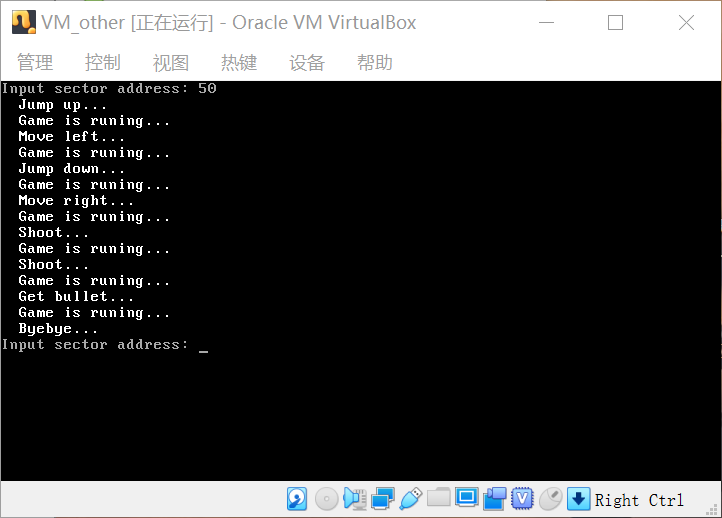
分别按下’W’, ’A’ ‘S’, ‘D’,控制虚拟任务的左右移动

按下空格键显示开枪射击



按下R键显示装填子弹



按下 ’ESC’ 键退出游戏，返回加载器

## 五、遇到的问题：

# 未将理论知识理解透彻便开始编写代码，走了很多弯路

首先，在写加载器代码的时候，看到书上加载器代码里的签名信息是”YANG”,于是我想都没想就改成了“ZOU”，然后开始了半个上午的查错之路…

中午吃饭的时候脑子里也都是汇编,觉得确实没有错误呀！开始怀疑是电脑的问题（hhh仿佛所有人实验做不出来都是说是设备的问题…）,然后突然意识到”YANG”的长度是4，因此书上代码是按预留4字节来计算的，而”ZOU”长度为3。想到这里便赶紧回教室打开了电脑，果然加载器就可以正确运行了，真的是令我哭笑不得的一个BUG…

# 在修改键盘中断的过程中遇到了问题

起初，我不太理解按键的扫描码与ASCII的真正区别。我想用IN 指令从键盘I/O接口（60H,64H）读取用户按键，然后根据按键的ASCII码，设置要输出的字符串的颜色。但是我在调试的过程中发现，无论我按什么键，都没有任何反应。

查阅资料后发现，IN指令从键盘缓冲区读到的是按键的扫描码，而系统会根据按键的扫描码（也可能同时有shift键或ctrl键等）再将其转换为相应的ASCII码。扫描码只跟按键的位置有关，而跟字母大小写等无关，于是我将判断条件改为了判断按键的扫描码，然后再转换为其对应的ACSII码，从而显示用户按下的字符;并判断按下的是 ’R’ 键时，将输出的字符串改为黑底红字;按下’G’ 键时，将输出的字符串改为黑底绿字;按下 ’B’ 键时，将输出字符串改为黑底蓝字;

DOS的界面缓冲区地址空间为B8000H~BFFFFH,是80\*25彩色字符模式的显示缓冲区。与我之前理解不同的是一个字符是占两个字节空间的，其中，一个字节代表的是字符的值，而另一个字节代表的则是输出字符的属性，即闪烁、背景色、高亮颜色等信息，修改对应字符的属性值即可改变界面颜色。

# 体会到了使用低级语言开发的难度

在使用高级语言编程时，可以使用一系列现有的集成IDE进行开发，尤其最近编译器技术飞速发展，例如VS2019，极大的降低了程序员开发与调试的难度！但是在我编写本次课程设计的过程中，每次向虚拟机内存写入程序都要经过繁琐的编译⇨链接⇨写入，由于没有操作系统，不能下断点或者单步执行，只能依据自己的编程经验查错，然后根据写入后虚拟机屏幕上的显示输出来判断程序是否正确，花费的时间很多。

比较佩服前人的智慧，用汇编语言做出了很多令我们觉得不可思议的工作，给我们留下了宝贵的“财富”！

## 六、心得体会

我个人比较喜欢汇编语言这门课程，也比较喜欢老师您的讲课风格！从计算机组成原理到汇编语言，得益于老师的教学方式，让我理论知识与实践经验都有所提高！您在上课的时候总会提一些问题，比较具有发散性的思维，很锻炼学生的思维方式。做完这份作业后，我也想通了计算机科班出身的核心竞争力到底在哪里？我想，绝不是编程能力（很多非科班的编程能力也很强），而应该是系统思维能力与逻辑思维能力！像现在几乎所有专业都在学C语言，数据结构，数据库等；起初我也想用汇编语言写一个小游戏“飞机大战”，在网上也查询了相关的资料，在编码过程中使用了emu8086，在EMU编译通过之后基本可以运行，但是无法用NASM进行编译以及写入，而且语法错误巨多。因此改成了一个模拟射击小游戏。想法是来自小时候2G网络时代的一款游戏，界面上没有画面，只有游戏分数以及当前进行的操作比如跳跃、射击、装弹、目标等提示信息。用户可以通过按键盘进行相应的操作！调试完成后，我想过再增加一个简单的画面显示功能，最后却由于数据结构课程设计以及工研班的几门数学课不得不放弃了写界面的工作，因为还在补修几门计算机课程，我时间确实有点安排不过来。感觉也挺遗憾的，我想把实验做到很好，一方面对汇编语言和计算机底层科学很感兴趣；另一方面我也想让自己满意，感觉自己考试考的不太好。平常学汇编挺认真的，而且考试前也很认真的复习，感觉汇编学的还是蛮好的。但是考的不是很满意，下考场之后就打开电脑跑了第一题，自己确实写错了，直接10分没了。应该也是最近两学期考的最不满意的一门课了，所以也想借着这次课程设计的机会，综合复习一下汇编语言的知识，以及弥补一下自己考试没考好的遗憾。

写到这里，汇编语言的课程也算告一段落了！回想起我最初对计算机底层知识的了解，远没有做完这份作业后清楚。同时，我也弄清楚了BIOS的基本原理，以及中断的相关知识，对计算机科学的认知也加深了很多！很幸运，我在老师您的指导下完成了计算机组成原理的学习，我的第一门计算机专业课给了我一个很好的开始；再加上现在的微机原理、汇编语言，和良好的数电、模电基础，这几门课构建好了我的专业知识框架，感觉学校开设的这几门课程确实很有用处。

实践出真知，如果老师没有留这份作业的话，我可能永远都不会自己完成如此底层的工作，编译、链接、写入、加载一步一步进行，但是做完了之后我也学到了很多；比如计算机的底层运行机制，生成可执行文件、如何输入输出、BIOS的运行管理机制、中断活动的运行、NASM与MASM的使用方法及其区别等知识。也体会到了前人的智慧，给我们提供了各种开发工具，极大的提高了项目的开发效率！

非常感谢老师两学期以来富有激情的讲课风格以及在实验方面的悉心指导，让我收获了很多。也希望自己在之后的学习生活中能够保持积极学习的态度和满满的求知欲。

最后祝老师工作顺利，开开心心！