

本科生实验报告

实验课程:\_\_\_\_\_\_操作系统\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 实验名称:\_\_\_\_\_\_Lab3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

专业名称:\_\_\_\_\_\_信息与计算科学\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生姓名:\_\_\_\_\_\_张文沁\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生学号:\_\_\_\_\_\_20337268\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

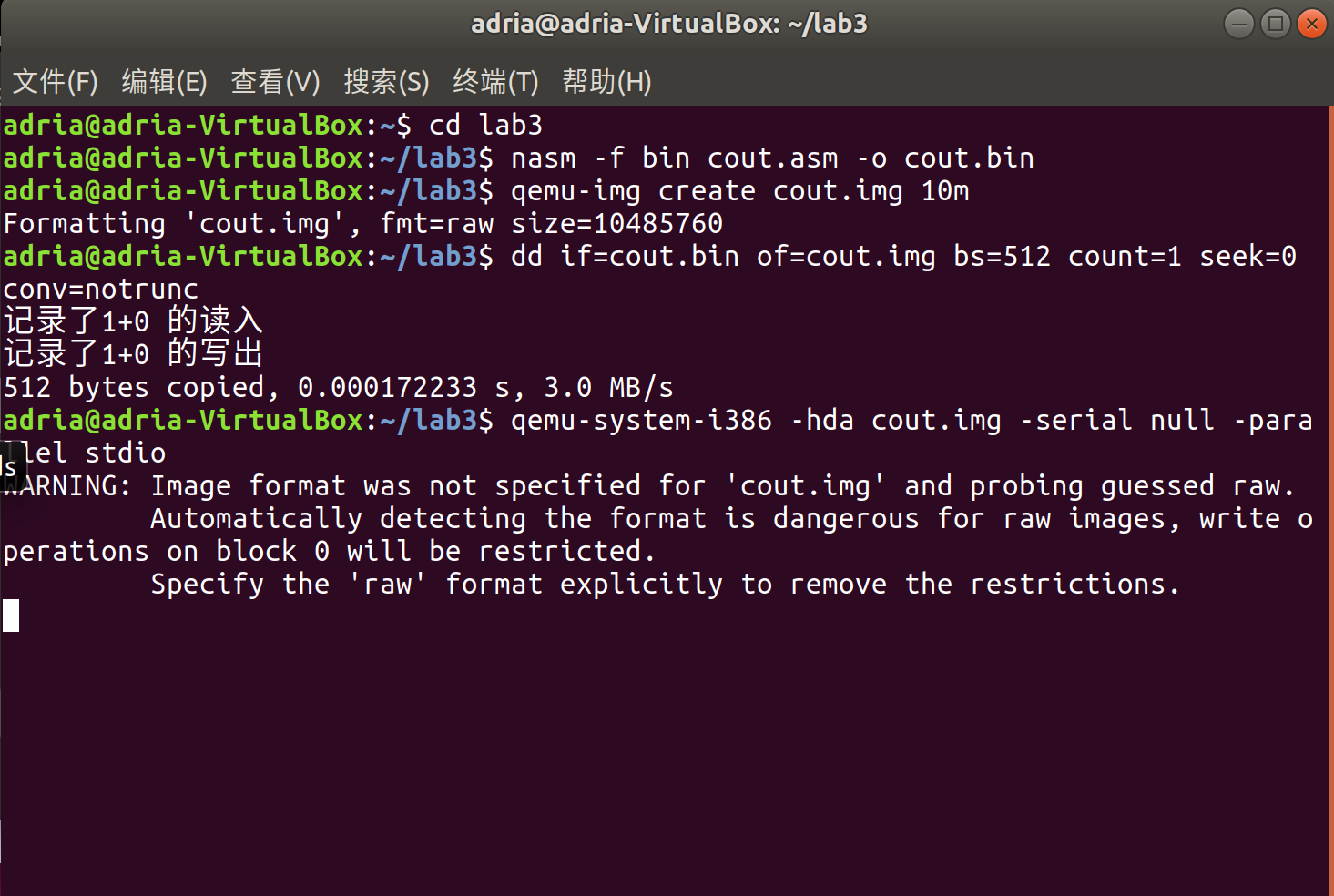
实验地点:\_\_\_\_\_无固定地点\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验成绩:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

报告时间:\_\_\_\_\_\_2022.3.12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **实验要求**
   1. **实验不限语言， C/C++/Rust都可以。**
   2. **实验不限平台， Windows、Linux和MacOS等都可以。**
   3. **实验不限CPU， ARM/Intel/Risc-V都可以。**
2. **实验过程**
   1. **Assignment 1 MBR**
      1. **1.1:复现example 1。说说你是怎么做的，并将结果截图。**

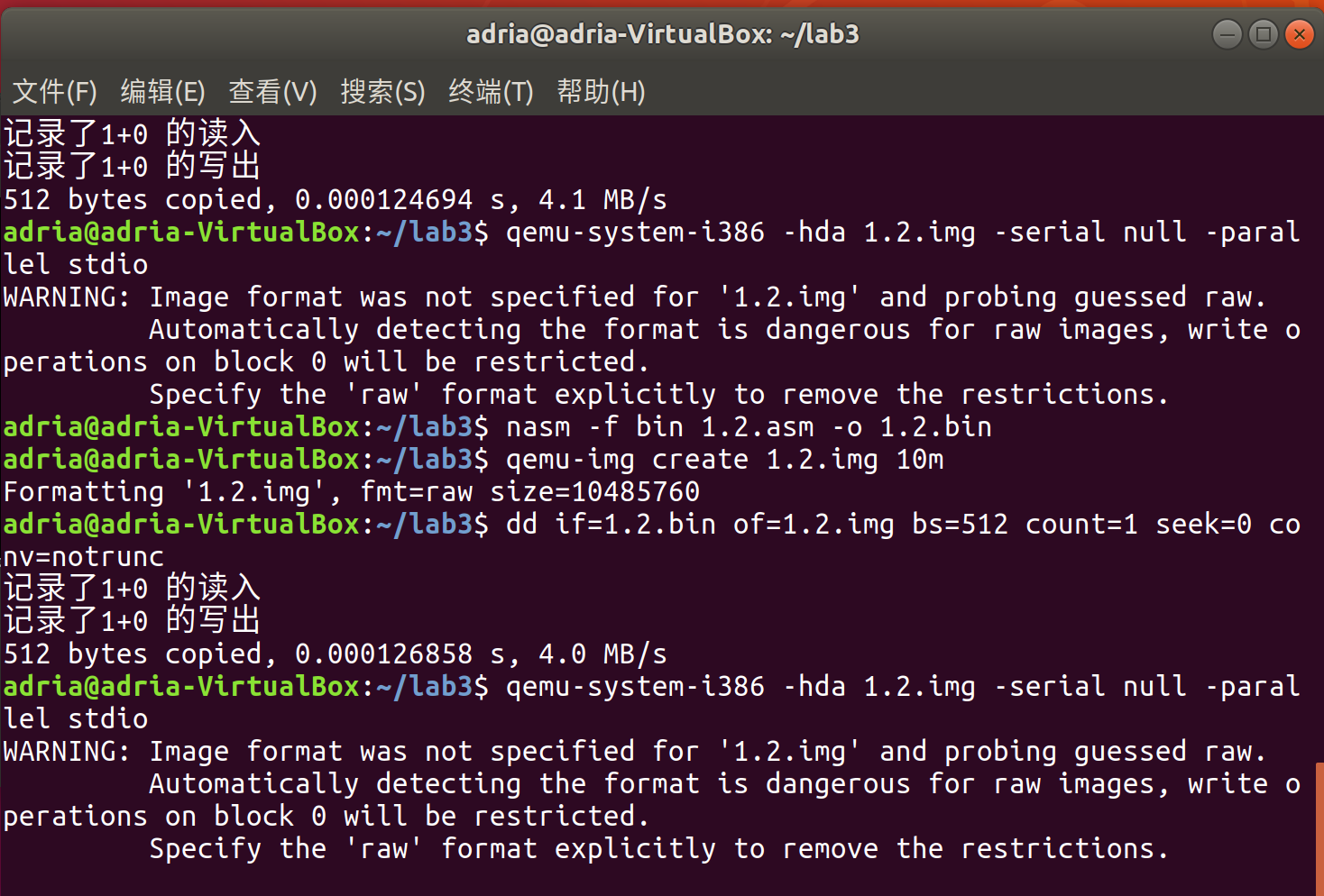
过程：根据实验指导进行一系列操作



* + 1. **1.2：请修改example 1的代码，使得MBR被加载到0x7C00后在(12,12)(12,12)处开始输出你的学号。注意，你的学号显示的前景色和背景色必须和教程中不同。**

过程：根据实验指导进行一系列操作并进行要求下的修改。根据显示位置公式计算初始位置。并改变显示颜色（0x07白色）。由qemu 显示屏是按 25x80 个字符来排列的矩阵可知（12,12）显存起始位置=0xB8000+2\*(80\*12+12)=0xB8000+2 ⋅972

用到命令如下：



* 1. **Assignment 2 实模式中断**
     1. **2.1：请探索实模式下的光标中断，利用中断实现光标的位置获取和光标的移动。**

通过输出字符串的方式证明进行了光标的获取和移动，获取体现在可以输出到指定位置，移动体现在输出之后光标在字符串最后面表示光标随着字符串的输出进行了移位。

* + 1. **2.2：请修改1.2的代码，使用实模式下的中断来输出你的学号。**

实现方式类似实验2.1，因为都是中断进行的输出，所以在2.1的基础上进行了修改，使之输出学号，使用了 int 10h 中断的 13H 功能

* + 1. **2.3： 在2.1和2.2的知识的基础上，探索实模式的键盘中断，利用键盘中断实现键盘输入并回显。**

使用了int 16h中断的0号功能。它会将键盘输入的字符的 ASCII 码放在寄存器al中。

* 1. **Assignment 3 函数的实现**
     1. **3.1： 分支逻辑的实现**
     2. **3.2：循环逻辑的实现**
     3. **3.3：请编写函数your\_function并调用之，函数的内容是遍历字符数组string**

学习教程中所给关于汇编语言的内容完成了实验。debug过程遇到困难在最后阐述。

* 1. **Assignment 4 汇编小程序**

先要写一个函数来延时，控制画框显示速度和抖动速度；接下来写函数判 断字符该往哪个方向走并打印字符。具体实现见代码区。

1. **关键代码**
   1. **Assignment 1 MBR**
      1. **1.1:复现example 1。说说你是怎么做的，并将结果截图。**
2. org 0x7c00
3. [bits 16]
4. xor ax, ax ; eax = 0
5. ; 初始化段寄存器, 段地址全部设为0
6. mov ds, ax
7. mov ss, ax
8. mov es, ax
9. mov fs, ax
10. mov gs, ax
11. ; 初始化栈指针
12. mov sp, 0x7c00
13. mov ax, 0xb800
14. mov gs, ax
15. ;输出
16. mov ah, 0x01 ;蓝色
17. mov al, 'H'
18. mov [gs:2 \* 0], ax
19. ...
20. jmp $ ; 死循环
21. times 510 - ($ - $$) db 0
22. db 0x55, 0xaa
    * 1. **1.2：请修改example 1的代码，使得MBR被加载到0x7C00后在(12,12)处开始输出你的学号。注意，你的学号显示的前景色和背景色必须和教程中不同。说说你是怎么做的，并将结果截图。**

答：和上文实验一致，只是修改了

mov ax,0xb8ac（起始位置，但是似乎并不精确，第二次直接在代码中加入了mov[gs:2\*972],ax的字样）

mov al,0x07（显示颜色）

* 1. **Assignment 2 实模式中断**
     1. **2.1：请探索实模式下的光标中断，利用中断实现光标的位置获取和光标的移动。**

通过输出字符串的方式证明进行了光标的获取和移动，获取体现在可以输出，移动体现在输出之后光标在字符串最后面表示光标随着字符串的输出进行了移位。

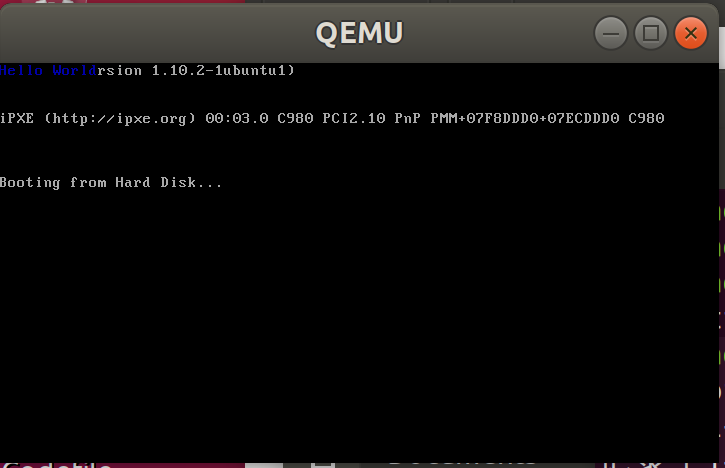
1. SECTION MBR vstart=0x7c00
2. ;将起始地址设置为0x7c00——因为BIOS会将MBR程序加载到0x7c00处
4. mov    sp, 0x7c00
5. ;清空屏幕，使用BIOS提供的中断
6. mov    ax, 0x600
7. mov    bx, 0x700
8. mov    cx, 0x0
9. mov    dx, 0x184f
10. **int**    0x10
11. ;下面获取当前的光标位置
12. ;    **INT** 0x10;    功能号:0x03    功能描述:获取当前光标位置
13. ;    输入:
14. ;        AH--功能号:    0x03
15. ;        BH--带获取光标的页码号
16. ;    输出:
17. ;        CH--光标开始行
18. ;        CL--光标结束行
19. ;        DH--光标所在行号
20. ;        DL--光标所在列号
22. ;调用功能号为0x3的BIOS中断，获取当前光标位置的相关信息，并将相关信息保存在对应的寄存器中
24. mov    ah, 0x03
25. mov    bx, 0
26. **int**    0x10
28. ;    下面进行打印字符串
29. ;    **INT** 0x10;    功能号:0x13    功能描述:打印出字符串
30. ;    输入:
31. ;        ES:BP--字符串地址
32. ;        AH--功能号    0x13
33. ;        AL--设置写字符串方式 1表示光标跟随移动
34. ;        CX--字符串长度(不包括最后的0)
35. ;        BH--设置要显示的页号
36. ;        BL--设置字符属性 0x2表示黑底绿字
38. mov    ax, cs
39. mov    es, ax
40. mov    ax, String
41. mov    bp, ax
42. mov    ax, 0x1301
43. mov    bx, 0x2
44. mov    cx, 0x12
45. **int**    0x10
46. ;调用0x13的BIOS中断，将0:String地址处，长度为0x12（后期随长度进行修改）的字符串进行了输出，并且光标跟随移动
47. jmp    $
48. String db "Cursor behind me."
49. times 510 - ($ - $$) db 0
50. db    0x55, 0xaa
    * 1. **2.2：请修改1.2的代码，使用实模式下的中断来输出你的学号。**

实现方式类似实验2.1，因为都是中断进行的输出，所以在2.1的基础上进行了修改，使之输出学号

* + 1. **2.3： 在2.1和2.2的知识的基础上，探索实模式的键盘中断，利用键盘中断实现键盘输入并回显**

使用了int 16h中断的0号功能。它会将键盘输入的字符的 ASCII 码放在寄存器al中，设置寄存器ah的颜色，将ax复制给0xB8000，就会在qemu屏幕的第一个位置显示输入的字符。结果如下：输入字符 a，qemu 屏幕上显示 a

1. mov gs,ax
2. mov sp,0x7c00
3. mov ax,0xb800
4. mov gs,ax
6. start:
7. ;清屏
8. mov ah,6
9. mov al,0
10. mov ch,0
11. mov cl,0
12. mov dh,25
13. mov dl,80
14. **int** 10h
15. mov ah,0;调用16号中断的0号功能
16. **int** 16h
17. mov ah,0xcf
18. mov [gs:2\*0],ax
19. ;死循环
20. jmp $
21. times 510 - ($ - $$) db 0
22. db 0x55,0xaa
    1. **Assignment 3 函数的实现**
       1. **3.1： 分支逻辑的实现**
23. %include "head.include"
25. your\_if:
26. mov edx,[a1]
27. cmp edx,12
28. jl case1
29. cmp edx,24
30. jl case2
31. jl case3
33. case1:
34. mov eax,[a1]
35. mov ebx,2
36. idiv ebx
37. inc eax
38. mov [if\_flag],eax
39. jmp end
41. case2:
42. mov eax,24
43. sub eax,[a1]
44. imul eax,edx
45. mov [if\_flag],eax
46. jmp end
48. case3:
49. mov eax,[a1]
50. shl eax,4
51. mov [if\_flag],eax
52. jmp end
54. end:
    * 1. **3.2：循环逻辑的实现**
55. your\_while:
56. loop1:
57. mov edx,[a2]
58. cmp edx,12
59. jl end\_loop
60. call my\_random
61. mov ebx,[while\_flag]
62. mov edx,[a2]
63. mov byte[ebx+(edx-12)],a1
64. dec edx
65. mov [a2],edx
66. jmp loop1
68. end\_loop:
    * 1. **3.3：请编写函数your\_function并调用之，函数的内容是遍历字符数组string**
69. your\_function:
70. xor eax,eax  ;清零操作
71. xor ecx,ecx
72. mov ebx,[your\_string]
73. loop2:
74. mov cl,byte[eax+ebx] ;注意是字节寄存器
75. inc eax
76. cmp ecx,0
77. je end\_for
78. pushad
79. push ecx
80. call print\_a\_char
81. pop ecx
82. popad
83. jmp loop2
85. end\_for:
86. ret
    1. **Assignment 4 汇编小程序**
       1. **延时程序**
87. Dn\_Rt equ 1
88. Up\_Rt equ 2
89. Up\_Lt equ 3
90. Dn\_Lt equ 4
91. delay equ 1000
92. ddelay equ 100          ; 计时器延迟计数,用于控制画框的速度
    * 1. **判断走向**
93. mov al,1
94. cmp al,byte[rdul]
95. jz  DnRt  ;跳转到对应的区块进行处理
96. mov al,2
97. cmp al,byte[rdul]
98. jz  UpRt
99. mov al,3
100. cmp al,byte[rdul]
101. jz  UpLt
102. mov al,4
103. cmp al,byte[rdul]
104. jz  DnLt
     * 1. **打印字符**
105. show:
106. xor ax,ax
107. ; 计算显存地址
108. mov ax,word[x]
109. mov bx,80
110. mul bx
111. add ax,word[y]
112. mov bx,2
113. mul bx
114. mov bp,ax
115. mov ah,[curcolor2]
116. ; 弹字符的背景色和前景色
117. inc byte[curcolor2]
118. cmp byte[curcolor2], 0fh
119. jnz skip
120. skip:
121. mov al,byte[**char**]       ; AL = 显示字符值（默认值为20h=空格符）
122. mov word[gs:bp],ax      ; 显示字符的ASCII码值
124. mov si, myinfo
125. mov di, 2
126. mov cx, word[infolen]
127. loop2:                      ; 显示myinfo中的每个字符
128. mov al, byte[ds:si]
129. inc si
130. mov ah, [curcolor]      ; 背景色和前景色
131. add byte[curcolor], 12h ; 变色，数字可随意调节
132. mov word [gs:di],ax
133. add di,2
134. loop loop2
135. jmp loop1
136. **实验结果**
     1. **Assignment 1 MBR**
        1. **1.1:复现example 1。说说你是怎么做的，并将结果截图。**

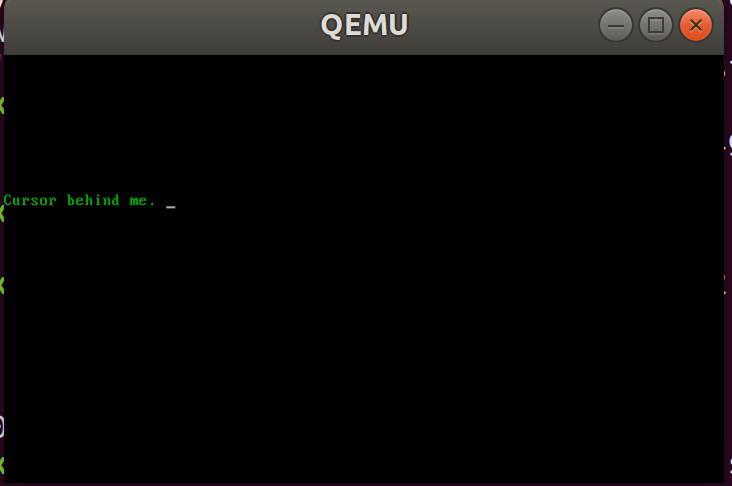
****

* + 1. **1.2：请修改example 1的代码，使得MBR被加载到0x7C00后在(12,12)处开始输出你的学号。注意，你的学号显示的前景色和背景色必须和教程中不同。说说你是怎么做的，并将结果截图。**



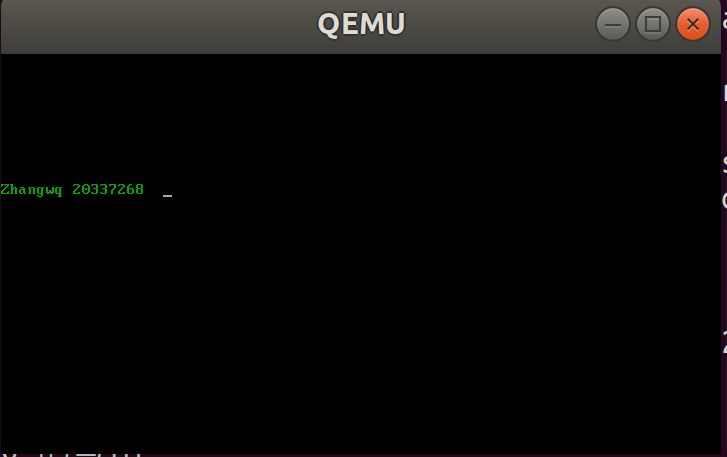
* 1. **Assignment 2 实模式中断**
     1. **2.1：请探索实模式下的光标中断，利用中断实现光标的位置获取和光标的移动。说说你是怎么做的，并将结果截图。**

解释：通过输出字符串的方式证明进行了光标的获取和移动，获取体现在可以输出，移动体现在输出之后光标在字符串最后面表示光标随着字符串的输出进行了移位。

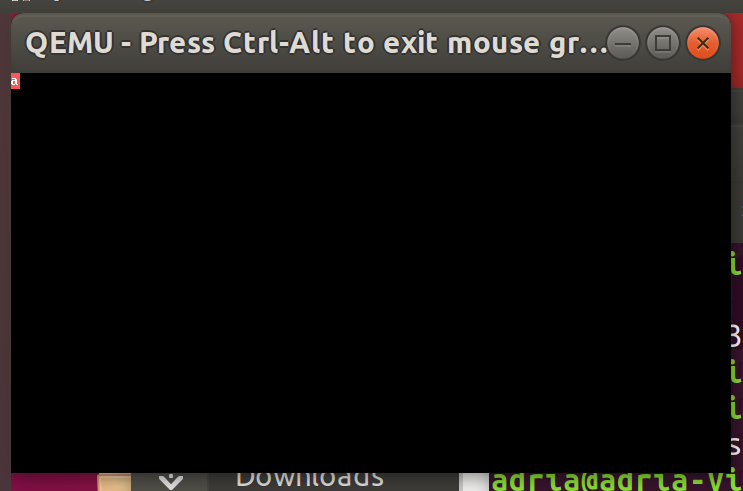
****

* + 1. **2.2：请修改1.2的代码，使用实模式下的中断来输出你的学号。说说你是怎么做的，并将结果截图。**

解释：因为原理相同，所以代码同上。

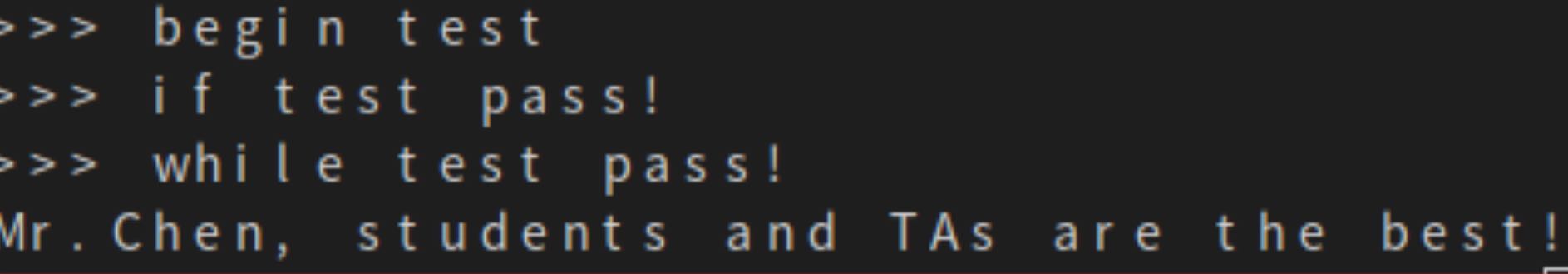
****

* + 1. **2.3： 在2.1和2.2的知识的基础上，探索实模式的键盘中断，利用键盘中断实现键盘输入并回显**

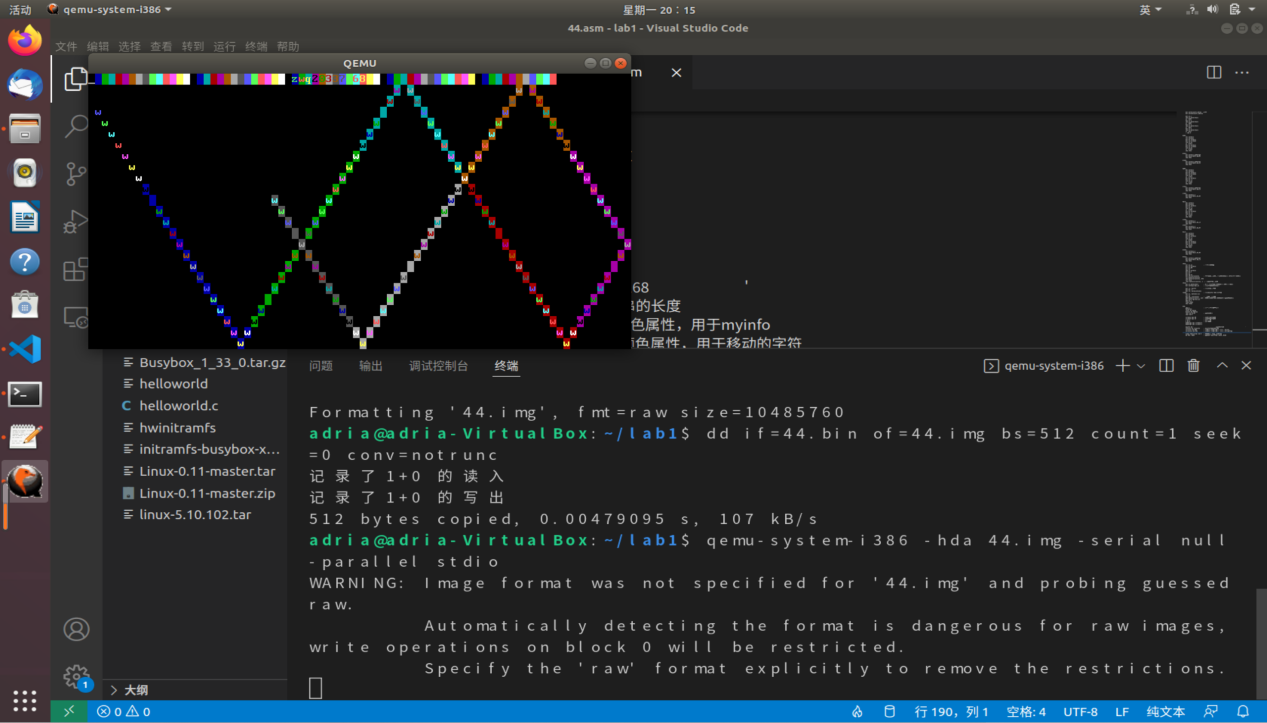
****结果如下：

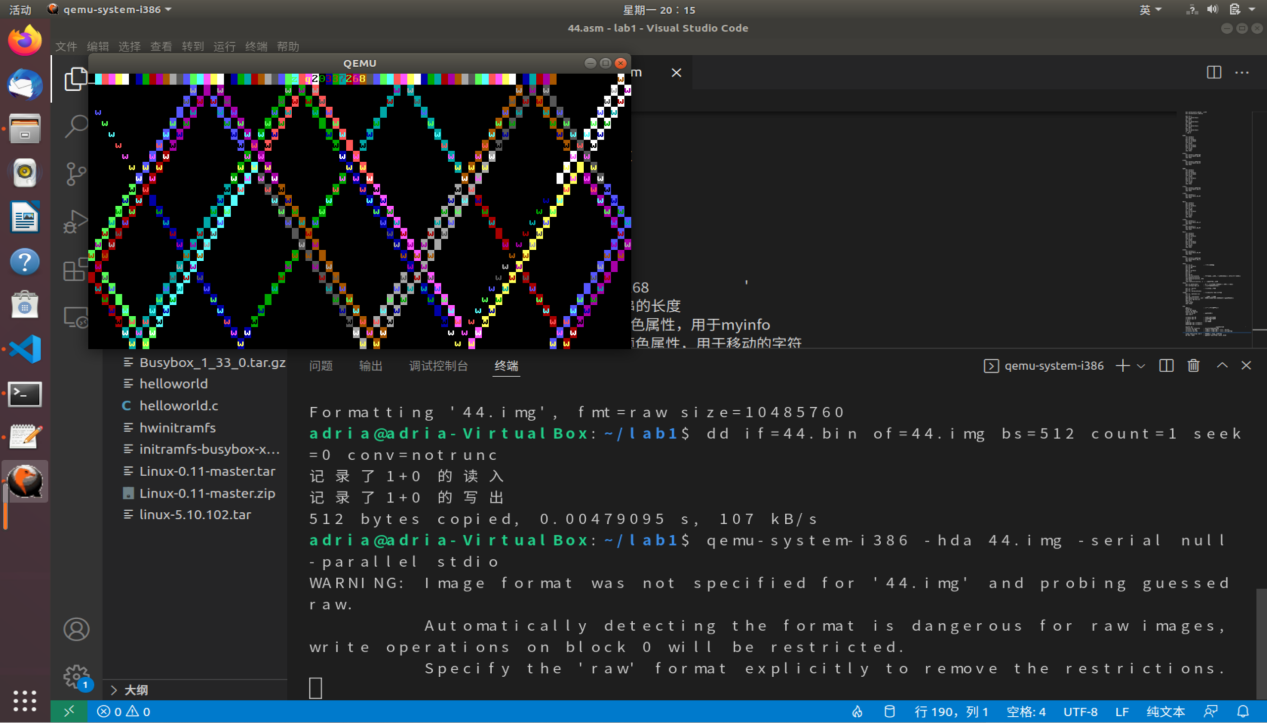
* 1. **Assignment 3 函数的实现**
     1. **3.1： 分支逻辑的实现**
     2. **3.2：循环逻辑的实现**
     3. **3.3：请编写函数your\_function并调用之，函数的内容是遍历字符数组string**

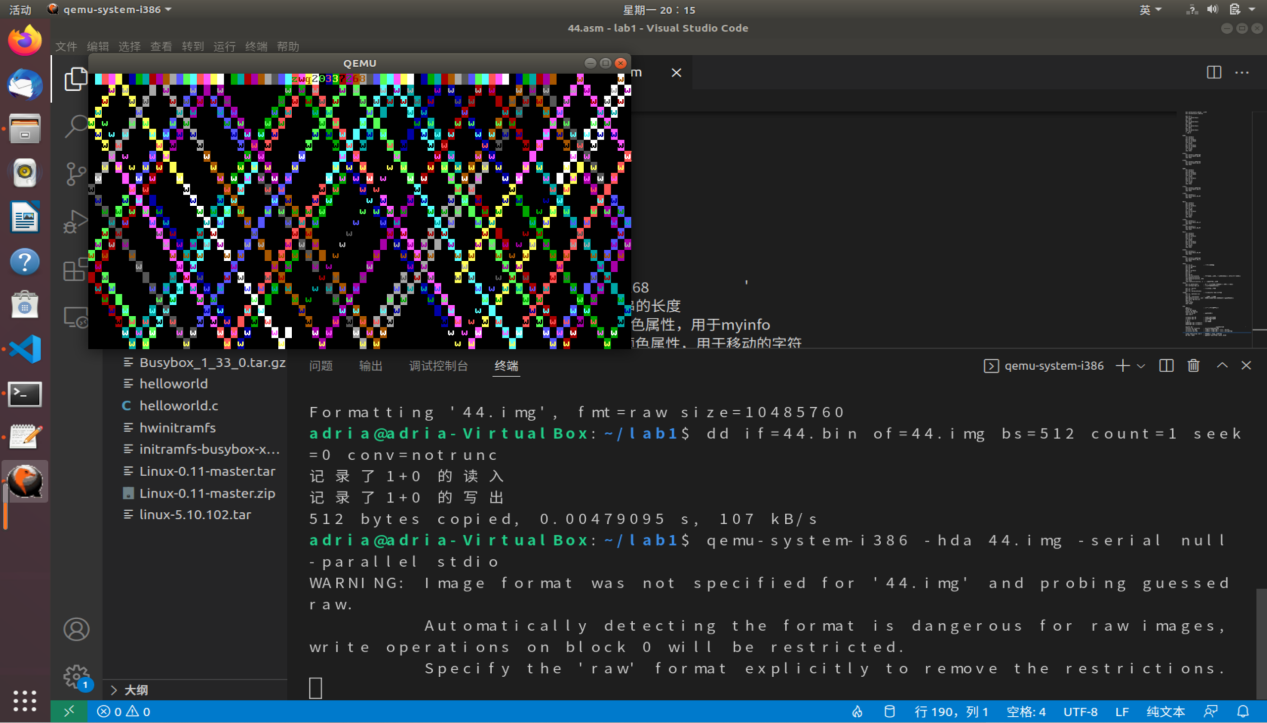
在终端进行运行会报错，但是放在vs code下面又可以正常运行

****

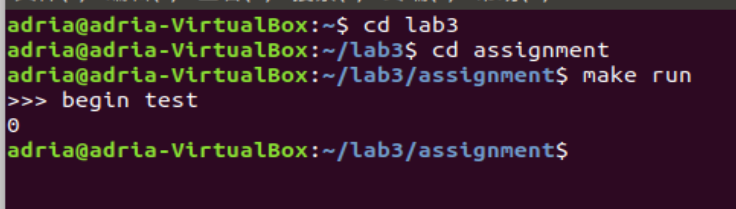
* 1. **Assignment 4 汇编小程序**

****

****

****

1. **总结**
   1. nasm汇编语言使用技能还需要提高
   2. 2.3中只能实现一个字符的输入和输出，无法完成字符串的功能
   3. mov eax,[a1] ;a1不能直接使用，而是要经过间接寻址才可以访问
   4. 使用了quit函数（自定义函数）导致只有一个输出



* 1. 分辨率怎么调试，在强制关闭虚拟机之后分辨率变差且无法恢复原样，附带使得终端无法执行命令，只得用vs code完成任务4
  2. 关于任务：1.2 按照计算的值来进行输出会出格子，取近似mov ax,0xb8ac或者0Xb8ab 0Xb8aa都相去甚远，最后查询得知可以使用mov[gs:2\*972],ax