OSlab1实验报告

王亚楠 141220107

邮箱：[wyn1996@hotmail.com](mailto:wyn1996@hotmail.com)

# 实验进度

已完成所有试验要求，并设计了拼图小游戏。

使用的gcc版本：32位

游戏规则：w——上，s——下，a——左，d——右

# 试验心得

1. 原来实验使用的是标准vga显示，由于要实现拼图的功能，对显示的像素和色彩都有更高的要求，因此我采用了真彩色模式。在start.S中改动了切换到图形模式的汇编代码。如图所示，  
     
   其中通过’movw $4f02, %ax’来将显示模式变成所需要的真彩色模式，$0x0115是800\*600＊24bit对应的编号，’movw $0x2000, %di’用来设定ModeInfoBlock所在的内存地址。  
   设置完真彩色模式后，我通过直接将图片对应的像素点映射到显存地址来实现图片的显示，通过查看ModeInfoBlock数组，得知显存地址的偏移量是40bites，所以通过0x2000+40来得到显存的起始地址，然后便可以按照我的想法进行图像的显示了。
2. 关于将图片转换成像素点数组的过程中遇到了一些比较坑的问题。  
   原来我是通过写一个shell脚本来实现转换的，开始的代码如下：   
     
   后来发现，图片的显示存在问题，部分像素点对应的位置发生了缺失和移位，然后我查看了像素点数组，发现其中有很多’\*’，我一开始直接去掉了’\*’，然后发现问题依旧存在。后来通过请教欧先飞后知道，因为这个脚本会将很多0看作一个’\*’，所以缺失了很多像素点。然后将转换中最关键的一行代码化成了如下所示，  
     
   即通过整个图片大小size的转换来获得，而不是1个bite1个bite的获得，这样就可以很好的规避之前’\*’的问题。  
     
   另外，通过和助教王诲喆沟通，他教给了我一个更加容易的方法，就是通过imagemagick工具对图片进行转换，先将要转换的图片变成.bgr格式，然后通过xxd将.bgr文件写到一个数组中去。
3. 关于键盘中断的话也想说些什么。  
   当然，经过PA以后，设定键盘中断并不是什么难事。先注册中断号，然后在中断处理函数中对接受到的中断号(irq=1001，即键盘中断)进行相应的处理即可。  
   但是，运行打字小游戏的时候发现，我电脑的键盘扫描码和实验所给的不一致，应该说是和qemu中内定的键盘扫描码不一致吧，暂时还没找到处理的方法。
4. 实现串口输出  
   通过对输出的格式要求对%d,%x,%c,%s依次进行判断然后按照规则调用printer函数进行输出。  
   之前没有接触过void \*\*args这样的存放字符串的指针数组，所以一开始不知道怎么获得%d相对的字符信息，所以还是纠结了比较久的。
5. 关于拼图游戏  
   其实这个游戏实现的话还是比较简单的，我是直接通过一个二维数组来表示每张图的位置。在game.c中获得当前按键的扫描码，然后判断扫描码对数组进行更新。然后更新完后，对拼图改变的两个位置重新写显存即可。
6. 对lab1整个的理解  
   首先从start.S来说，先是关中断，设定真彩色显示模式、调用BIOS  
   ，然后设置段寄存器，打开A20地址线，接着对GDT进行设置，接着进入保护模式，对段寄存器和栈进行设置。  
   整个游戏的话，首先要对串号端口、时钟端口、中断符表以及我们所需要用到的时钟中断和键盘中断进行初始化，然后打开中断，进入游戏。  
   lab1加深了我对中断、模式切换的理解，同时自己实现一个小游戏也让人充满自豪感。