实验报告

史子凡 151180115

邮箱：903498500@qq.com

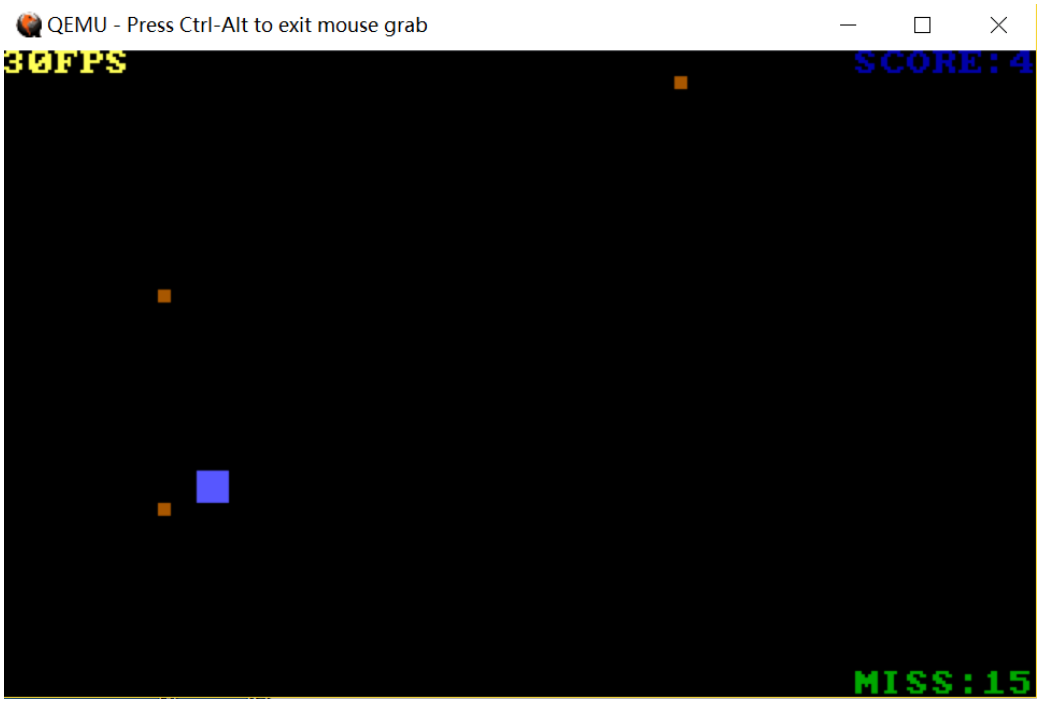
一.实验进度。

我已完成了 lab 的全部内容，并用 git 进行了记录。

gcc 版本是4.9.2.

```
gcc version 4.9.2 (Debian 4.9.2-10)
```

游戏是一个类似于接水果的小游戏（代码里面的变量名设成了飞机碰石头……）。玩家通过上下左右四个键来控制篮子的位置，水果从上方随机位置落下，如果接到就加一分，如果没接到错过的数量就加一。游戏界面如下图所示：



二.实验遇到的问题。

1.bootloader 里踩的坑：

Bootloader 部分主要参考的是 jos 的框架，然后在 main.c 里面就被坑到了。

```
for (; ph < eph; ph++)
    // p_pa is the load address of this segment (as well
    // as the physical address)
    readseg(ph->p_pa, ph->p_memsz, ph->p_offset);
```

从这段代码可以看出它把整个段都读了，而之前在 PA 中学到的应该是只读 p_offset~p_offset+p_filesz 这一部分，然后把 p_offset+p_filesz~p_offset+p_memsz 这一部分清

零。之前没发现这个问题，导致.data 节中的变量没有按照要求清零，被赋了别的值，然后在后面就出现各种奇怪的错误，最后发现都是初始化的过。于是乎，就再读之前写的代码，才找到这边的问题。然后修改为：

```
for(; ph < eph; ph ++) {
    pa = (unsigned char*)ph->p_pa;
    readseg(pa, ph->p_filesz, ph->p_offset);
    for (i = pa + ph->p_filesz; i < pa + ph->p_memsz; *i ++ = 0);
}
```

2.Makefile 中的问题：

我用的 Makefile 是网站上提供参考的 Makefile 文件，然后就遇到了问题。在 kernel 的 irq 文件夹下写了一个 do_irq.S 的文件，用于处理中断和异常。然而在 make 的时候，每次都报错说 vec0、irq0之类的没有被定义，也就是说 do_irq.S 这个文件并没有被编译进去从而产生相应的依赖关系。本来以为是某部分的函数没有写所以导致缺少依赖关系，但后来发现，原来问题是出在 Makefile 文件中。在下图中，我们可以看到 kernel 底下寻找.S 结尾的文件时用的是 wildcard。后来上网搜了一波，才知道 wildcard 寻找.S 结尾的文件时不会继续到子文件夹里找，所以 kernel 的子文件夹 irq 里不会被检测。

```
48 KERNEL_C := $(shell find $(KERNEL_DIR) -name "*.c")
49 KERNEL_S := $(wildcard $(KERNEL_DIR) -name "*.S")
```

因此，要将 wildcard 改写为与上一句寻找.c 文件一样的 shell find。

```
48 KERNEL_C := $(shell find $(KERNEL_DIR) -name "*.c")
49 KERNEL_S := $(shell find $(KERNEL_DIR) -name "*.S")
```

然后，编译就很顺利的通过了。

3.无法找到游戏的入口

之前就有这个疑问，如果游戏中的函数名没有为 main 的，那么运行时机器知道要从哪里开始吗？果真，机器不知道。于是乎，在 kernel 链接的时候加入了一句“-e game_init”，显式地告诉了机器应该从 game_init()这个函数开始运行（如下图所示），于是，游戏就可以运行了。

```
$(KERNEL): $(KERNEL_O) $(LIB_O)
    $(LD) -m elf_i386 -e game_init -T $(LD_SCRIPT) -nostdlib -o $@ $^ $(shel
l $(CC) $(CFLAGS) -print-libgcc-file-name)
```

4.第一部分

看到第一部分的要求时不知道要从哪里下手，基本上都参考了 jos 和2012级的 oslab 框架，深深的体会到了汇编的魅力。还必须感谢下上学期的 PA，虽然写的很苦很累，但是帮我们了解了很多知识，所以到 oslab 时就减少了很多麻烦，比如保护模式、ELF 之类的。