Lab3

首先查看 Env 的数据结构，发现 Env 和 Page 一样，是通过双向链表管理的。有了 Page 链表的经验，对于 Env 链表的操作也是很顺手的。

在 i386_vm_init(void)中分配了 Env 的存放空间，映射至 UENV。然后在 env_init 中将每个 Env 逆序插入列表。

Env_setup_vm 为环境分配页目录的存放空间，然后将 env_cr3 指向该物理地址，Env_pgdir 指向逻辑地址，然后将 UTOP 以上的目录项全部拷贝进来。因为这部分对于所有环境都是一样的。

segment_alloc 是为环境分配内存。有了上个 lab 做 pmap.c 的经验，这个也是比较简单的。

Load_icode 是根据 bootmain 中的相应代码写的，基本没问题。就是最后要设置一下环境运行的入口 e->env_tf.tf_eip。

还有一些函数比较简单就不说了。做到这里，我没测试就直接写后面的代码了(但是没有完全实现)。然后测试下来只有 5 分，不过理当 15 分一起拿的呀。于是不断地看，后来才发现原来是这个原因。

中断向量的设置只要在汇编文件 trapentry.S 中运用宏定义就可以了。然后在 idt_init 里面运用 SETGATE 来设置中断门。权限的问题参考了一下资料，基本没什么问题。

然后写 alltraps，压入调用 trap 函数所需要的 trapframe 的一些信息包括寄存器。然后 Pop 出那些被保存的寄存器值，最后返回。这个代码不多，虽然写汇编比较麻烦，但是还好，没出什么错误。但是在写系统调用部分的汇编时就出了问题。

Syscall 的 handler 只要将参数压入栈就可以了直接调用 kern 里的 syscall 了调用返回后就 sysexit。因为不是特别了解嵌入汇编在 lib 的 syscall 里面犯了很多错。首先是重复定义的问题，在嵌入式汇编中写了个返回时跳转所用的标记（字母写的），但是编译时居然提示重复定义，但注释后通过编译，于是我查看 objdump 的汇编代码这才发现这段汇编果然出现了4次，立马想到是 inline 的问题，删掉 syscall 的 inline，编译就过了。后来又发现如果地址标记是数字，比如“1:”然后用到这个标记时运用“1f”来表示，这样不用删除 inline 也会过。不过我还是用了第一种方法。在嵌入式汇编的最后一个分号后面写上在汇编中改变的寄存器，这个试了很长时间。还有就是在汇编中一不小心保存了一些不应该保存的寄存器，比如 eax，这导致调用后返回了错误的值。

我在到了60分的时候，最后的5分老是拿不到，打印出来，原来是系统调用的次数多了1，于是想到应该再每次调用之后才将系统调用数累加，而不应该在调用之前就加1。

此次 lab 很多的 debug 时间都放在了很多细微的错误上，整体的思路是明晰的。