作业四

理论题

习题4.1

答：是。在同一个进程中，线程之间有共享的内存空间，它们之间切换时要交换的信息所占用的储存空间少，所以开销少、速度快；而在不同进程中的线程做切换时，要交换的状态信息更多，它所消耗的储存空间也更多，所以不同进程间的线程切换开销要更大

习题4.2

答：因为用户级线程，其结构是对操作系统不可见的，而操作系统的调度是以进程为单位，操作系统不知道是哪一个进程中的线程在做系统调用。

习题4.7

答：

a. 该函数的功能是，计算链表l中存储的元素值，有多少个正数。

b. 有可能出现一个线程的计算结果被另一个线程的结果覆盖的情况。比如，在一个线程对全局变量global_positives进行加1操作，但内存中的值还没来得及更改时，另一个进程也进行了加1操作，这导致和正确结果相差1，如果链表中正数很多的话，最终结果将会出错，小于真实值。

习题4.9

答：

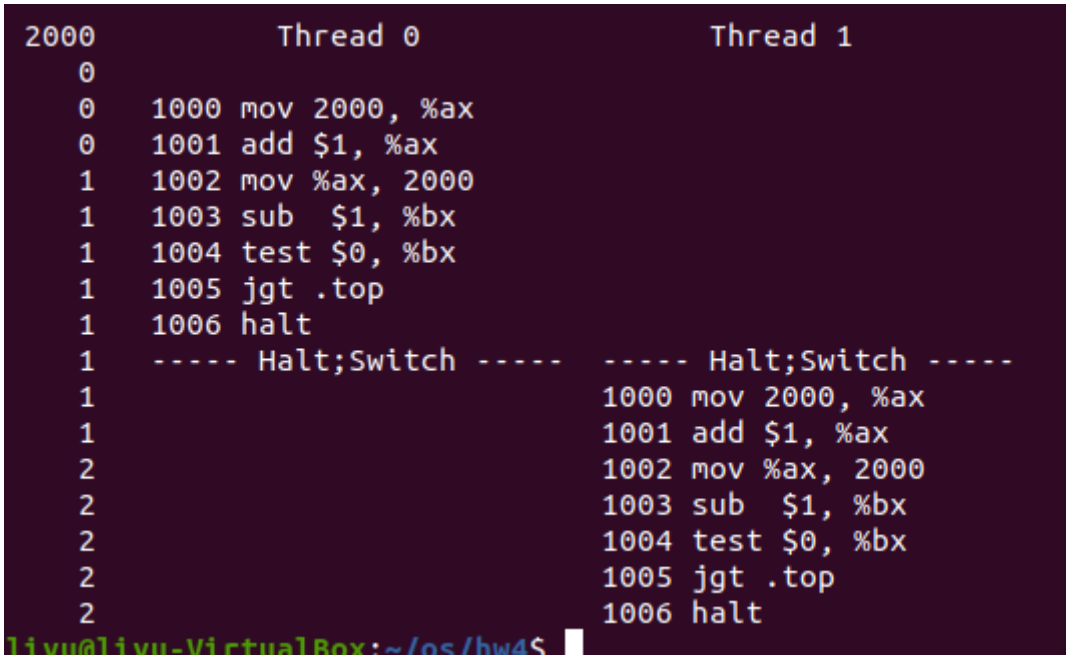
a. 高程序实现的功能是主程序创建一个新线程，两个线程每个对myglobal变量加20，并且主程序每进行一次加1操作就输出“o”，另一个线程每进行一次加一操作就输出“.”。

b. 不是期望的。两个线程对全局变量可以同时进行读写操作，但这是容易出现错误的。在一个线程的计算结果来不及更新时另一个线程的计算结果就将内存中的数读出加一之后再写回，这样就导致了最后结果要小于期望值的情况。我们可以设置临界区，使两个线程产生互斥，对全局变量的读写操作加锁，同一时间只允许一个线程进入并修改它的值。

实验题

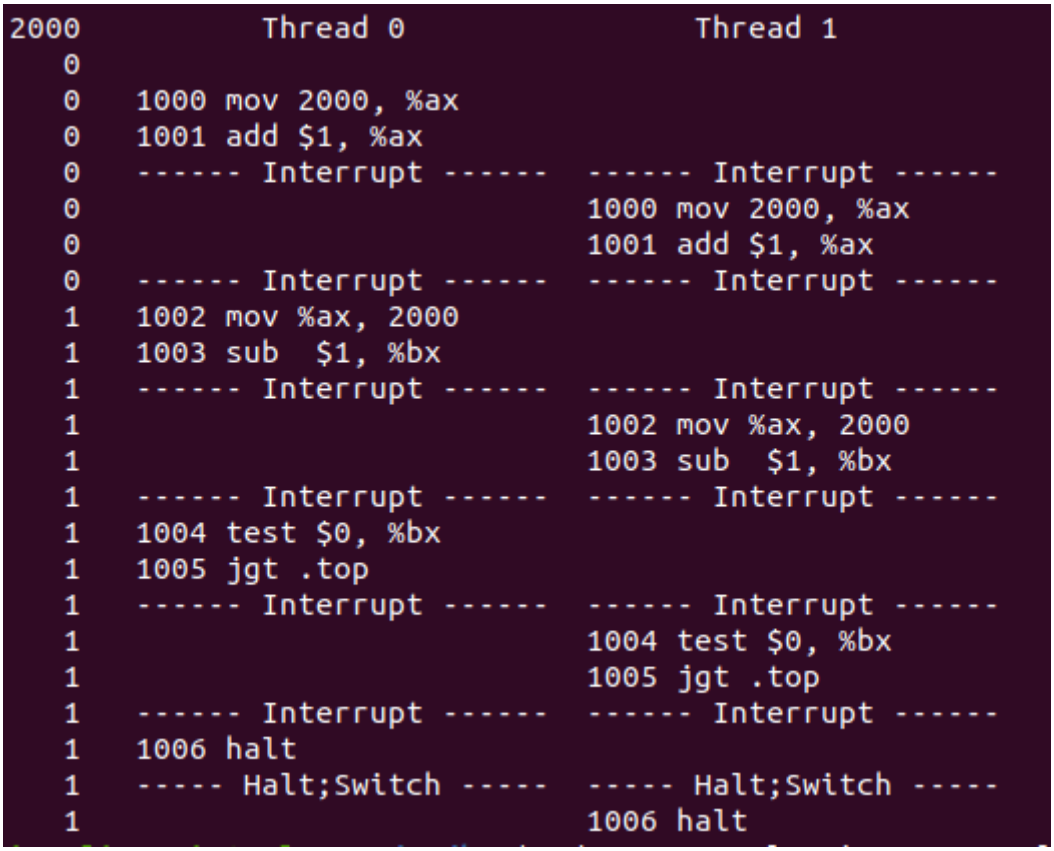
运行 looping-race-nolock.s 代码

结果显示



在没有竞争的情况下，两个线程应该是互不影响的访问内存。

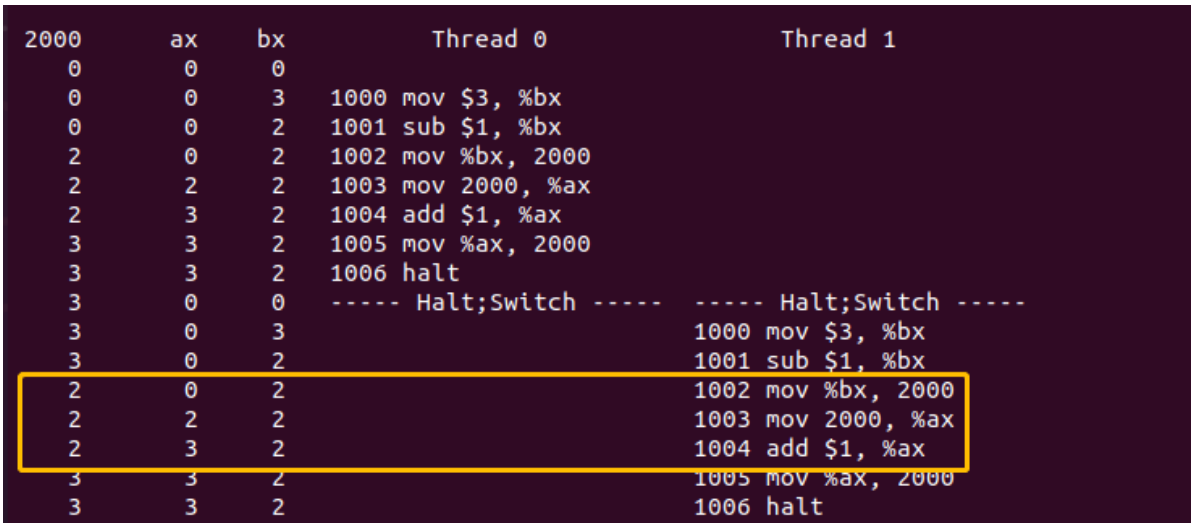
但是如果我们给它加了中断的话，因为两个线程之间存在竞争，从而导致对同一块内存地址的读写产生冲突，结果如下。最终结果为内存地址2000中存的数值是1，而如果串行运行的话最终结果应该为2.



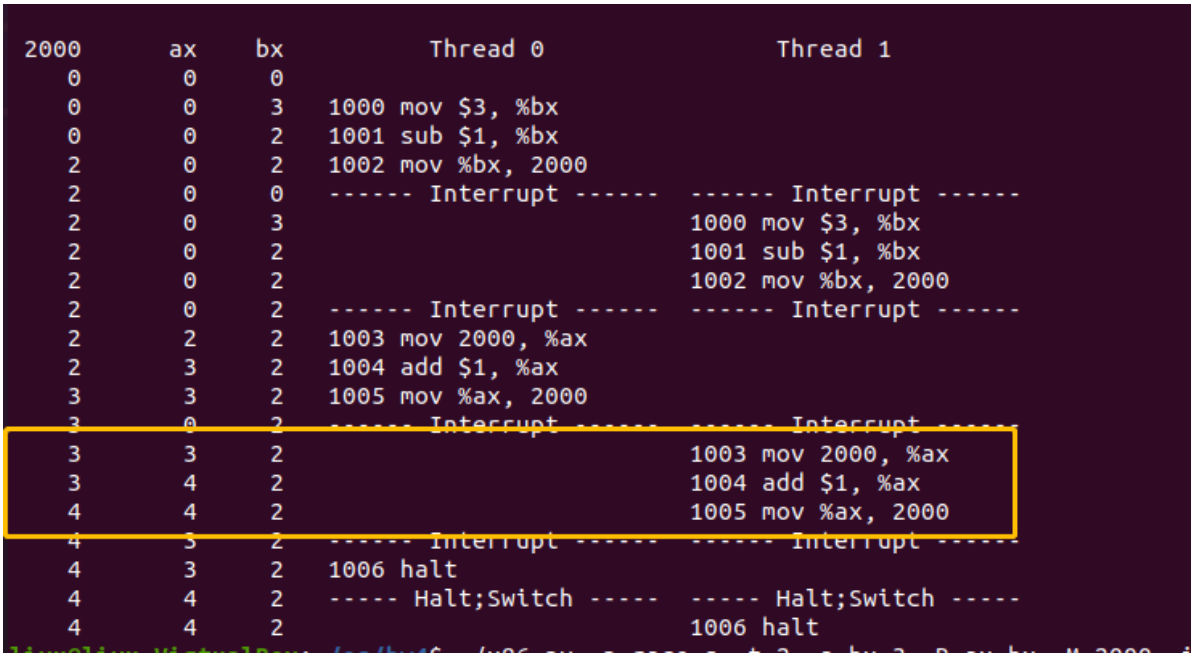
自己编写代码

```
# %bx中的值初始化为3

.main
.top
mov $3, %bx
sub $1, %bx     # 给bx寄存器中的值减1
mov %bx, 2000   # 将bx寄存器中的值存入内存地址为2000的内存单元
mov 2000, %ax   # get 'value' at address 2000
add $1, %ax     # increment it
mov %ax, 2000   # store it back

halt
```

2个线程串行结果



2个线程，每三个指令一次中断的并行结果



我们可以看到，在thread1把内存地址2000处的值移到ax并进行后续计算时，其值为3。但如果是串行计算的话，此时内存地址为2000处的值应该为2，所以这就导致了最终结果在内存地址为2000处储存的值并行比串行多了1。

由此，可以体现多线程在运行时的竞争结果