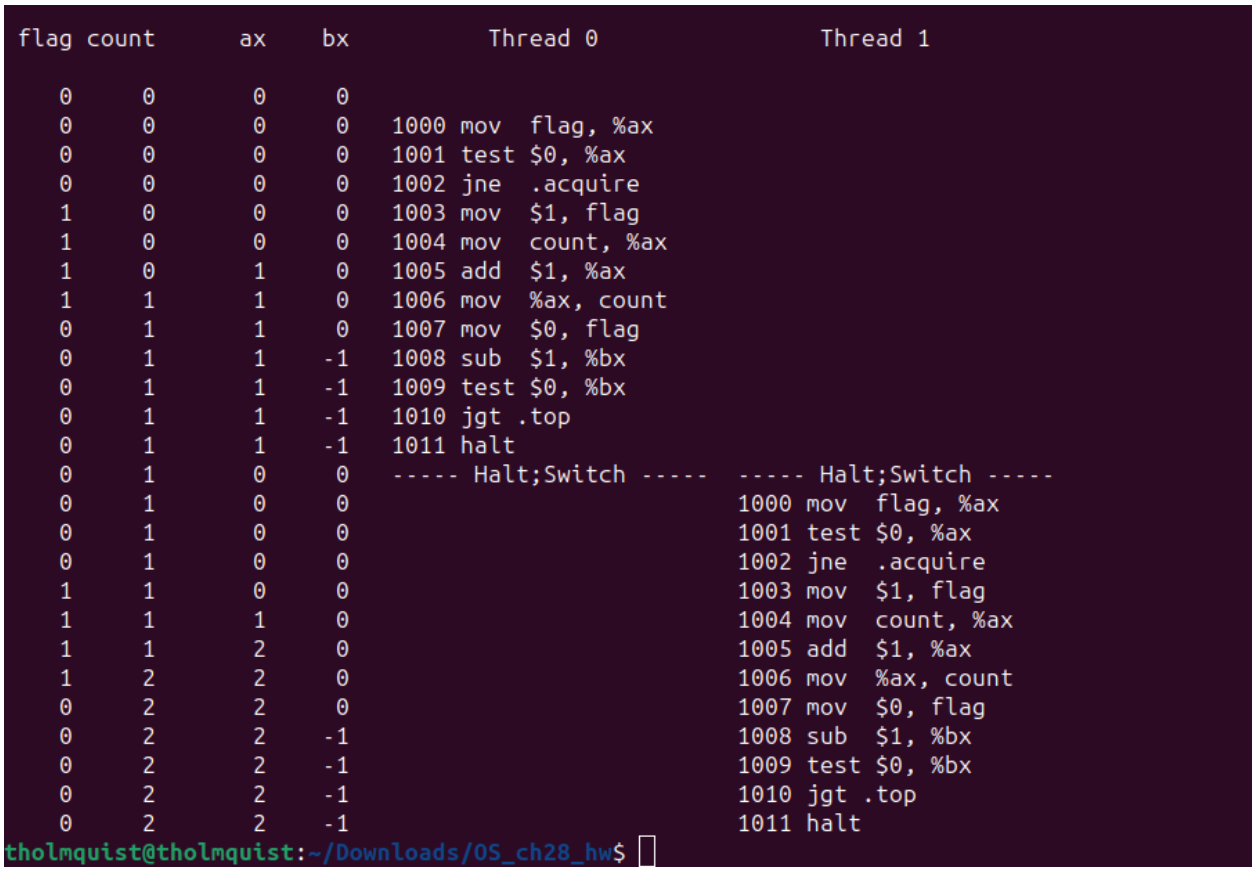
Ch.28 Simulation hw
Tyler Holmquist

1. Yes the assembly is looping until the flag is 0 or the lock is available then it changes the flag to 1 taking the lock. Then it increments count and changes the flag back to 0 releasing the lock, and finally checks to see if it is still looping.
2.

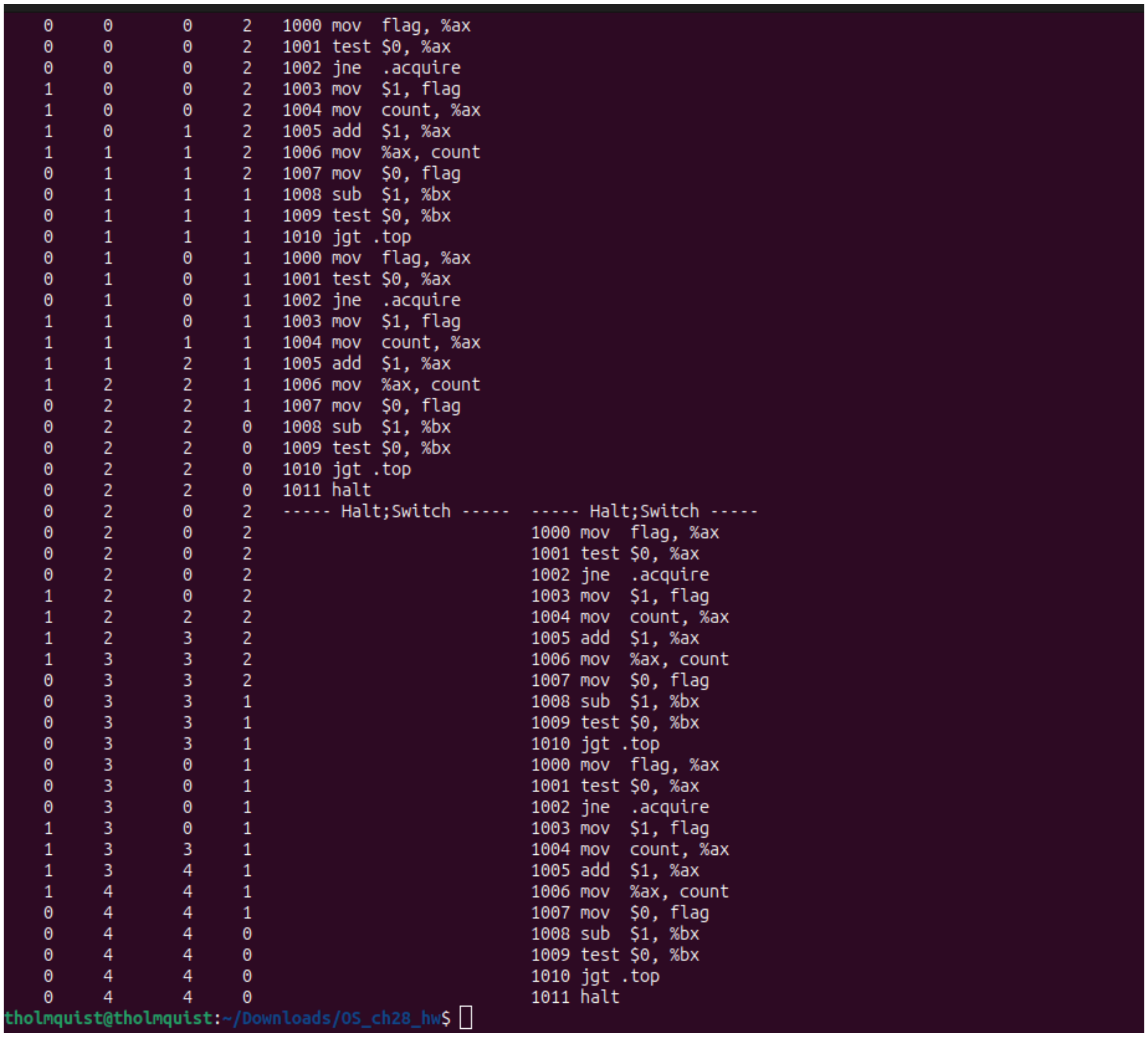


```
 flag count      ax     bx          Thread 0                Thread 1

    0     0       0      0
    0     0       0      0   1000 mov  flag, %ax
    0     0       0      0   1001 test $0, %ax
    0     0       0      0   1002 jne  .acquire
    1     0       0      0   1003 mov  $1, flag
    1     0       0      0   1004 mov  count, %ax
    1     0       1      0   1005 add  $1, %ax
    1     1       1      0   1006 mov  %ax, count
    0     1       1      0   1007 mov  $0, flag
    0     1       1     -1   1008 sub  $1, %bx
    0     1       1     -1   1009 test $0, %bx
    0     1       1     -1   1010 jgt .top
    0     1       1     -1   1011 halt
    0     1       0      0   ----- Halt;Switch -----  ----- Halt;Switch -----
    0     1       0      0                            1000 mov  flag, %ax
    0     1       0      0                            1001 test $0, %ax
    0     1       0      0                            1002 jne  .acquire
    1     1       0      0                            1003 mov  $1, flag
    1     1       1      0                            1004 mov  count, %ax
    1     1       2      0                            1005 add  $1, %ax
    1     2       2      0                            1006 mov  %ax, count
    0     2       2      0                            1007 mov  $0, flag
    0     2       2     -1                            1008 sub  $1, %bx
    0     2       2     -1                            1009 test $0, %bx
    0     2       2     -1                            1010 jgt .top
    0     2       2     -1                            1011 halt
tholmquist@tholmquist:~/Downloads/OS_ch28_hw$
```

yes it does work, and the flag will be 0 because both threads will be done with the lock leaving it open at then end.

3.



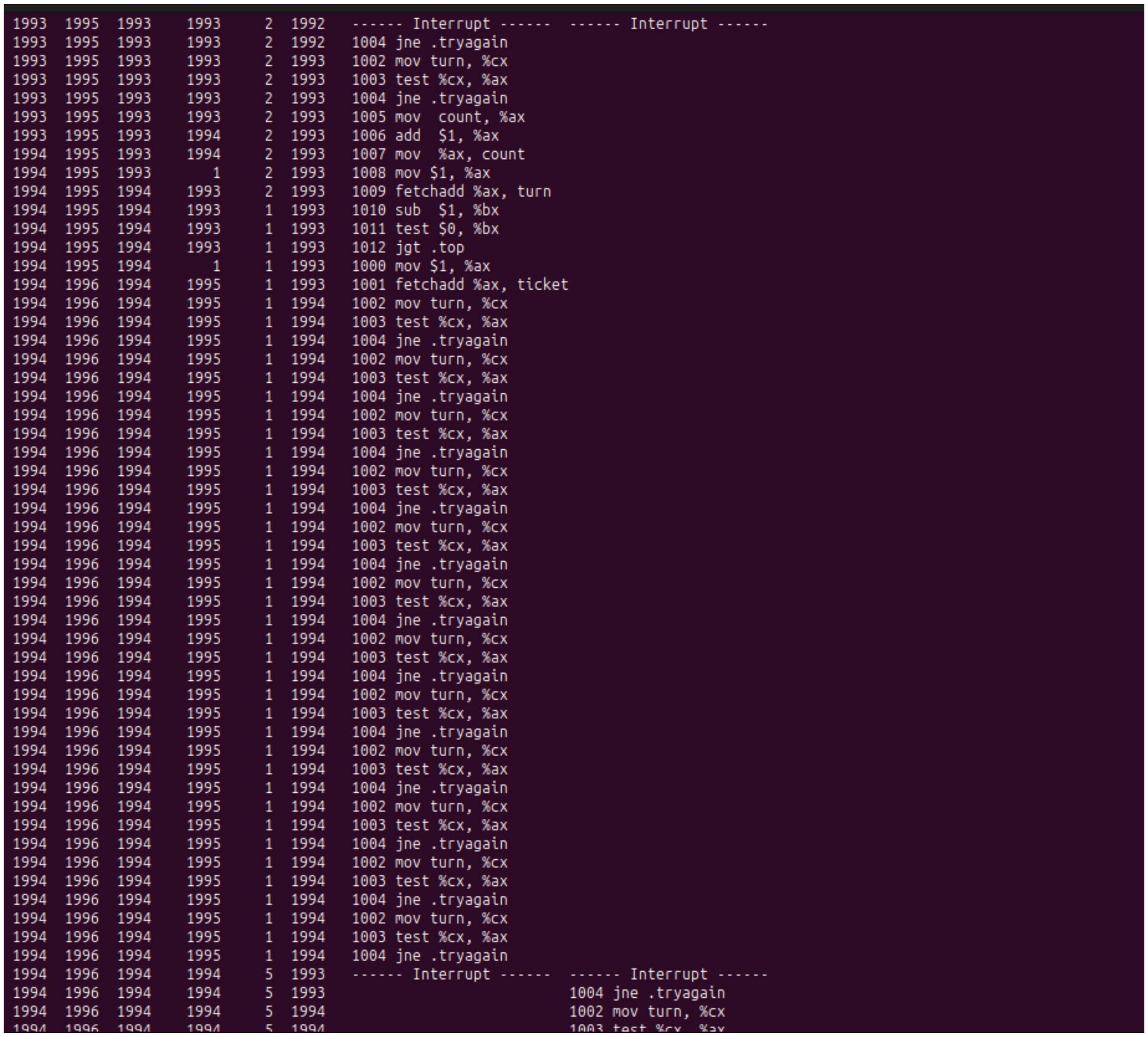
```
    0     0       0     2   1000 mov  flag, %ax
    0     0       0     2   1001 test $0, %ax
    0     0       0     2   1002 jne  .acquire
    1     0       0     2   1003 mov  $1, flag
    1     0       0     2   1004 mov  count, %ax
    1     0       1     2   1005 add  $1, %ax
    1     1       1     2   1006 mov  %ax, count
    0     1       1     2   1007 mov  $0, flag
    0     1       1     1   1008 sub  $1, %bx
    0     1       1     1   1009 test $0, %bx
    0     1       1     1   1010 jgt .top
    0     1       0     1   1000 mov  flag, %ax
    0     1       0     1   1001 test $0, %ax
    0     1       0     1   1002 jne  .acquire
    1     1       0     1   1003 mov  $1, flag
    1     1       1     1   1004 mov  count, %ax
    1     1       2     1   1005 add  $1, %ax
    1     2       2     1   1006 mov  %ax, count
    0     2       2     1   1007 mov  $0, flag
    0     2       2     0   1008 sub  $1, %bx
    0     2       2     0   1009 test $0, %bx
    0     2       2     0   1010 jgt .top
    0     2       2     0   1011 halt
    0     2       0     2   ----- Halt;Switch -----  ----- Halt;Switch -----
    0     2       0     2                            1000 mov  flag, %ax
    0     2       0     2                            1001 test $0, %ax
    0     2       0     2                            1002 jne  .acquire
    1     2       0     2                            1003 mov  $1, flag
    1     2       2     2                            1004 mov  count, %ax
    1     2       3     2                            1005 add  $1, %ax
    1     3       3     2                            1006 mov  %ax, count
    0     3       3     2                            1007 mov  $0, flag
    0     3       3     1                            1008 sub  $1, %bx
    0     3       3     1                            1009 test $0, %bx
    0     3       3     1                            1010 jgt .top
    0     3       0     1                            1000 mov  flag, %ax
    0     3       0     1                            1001 test $0, %ax
    0     3       0     1                            1002 jne  .acquire
    1     3       0     1                            1003 mov  $1, flag
    1     3       3     1                            1004 mov  count, %ax
    1     3       4     1                            1005 add  $1, %ax
    1     4       4     1                            1006 mov  %ax, count
    0     4       4     1                            1007 mov  $0, flag
    0     4       4     0                            1008 sub  $1, %bx
    0     4       4     0                            1009 test $0, %bx
    0     4       4     0                            1010 jgt .top
    0     4       4     0                            1011 halt
tholmquist@tholmquist:~/Downloads/OS_ch28_hw$ 
```

switching bx to 2 makes it so each thread loops twice and the flag is still 0 at the end because each thread finishes its process and releases the lock.

4. After setting bx to 10, I found that the good outcomes occur when -i is set to 11,15, or 16. The rest of 1-20 are bad outputs and I tested no farther than those variants. This is because for these three good inputs it allows each thread to either finish its process before switching or switches threads at a time that doesn't mess with the parameters used such as flag in an out of order way.
5. The lock acquire is written by swapping the mutex with the ax register which holds one in order to keep it's state if the lock is taken or take the lock if it isn't. If the lock is not available then it tries again until it is. The release is done by just switching the mutex from 1 to 0 showing that the lock is free again.
6. It performs as expected and correctly with all intervals tested 1-20 since the mutex always ended at 0. It may have been inefficient though for some cases because the threads would have to switch more frequently causing more overhead rather than letting

a job finish then switching. Also if the intervals are too large then there is an excessive wait time that could cause inefficiency as well.

11.

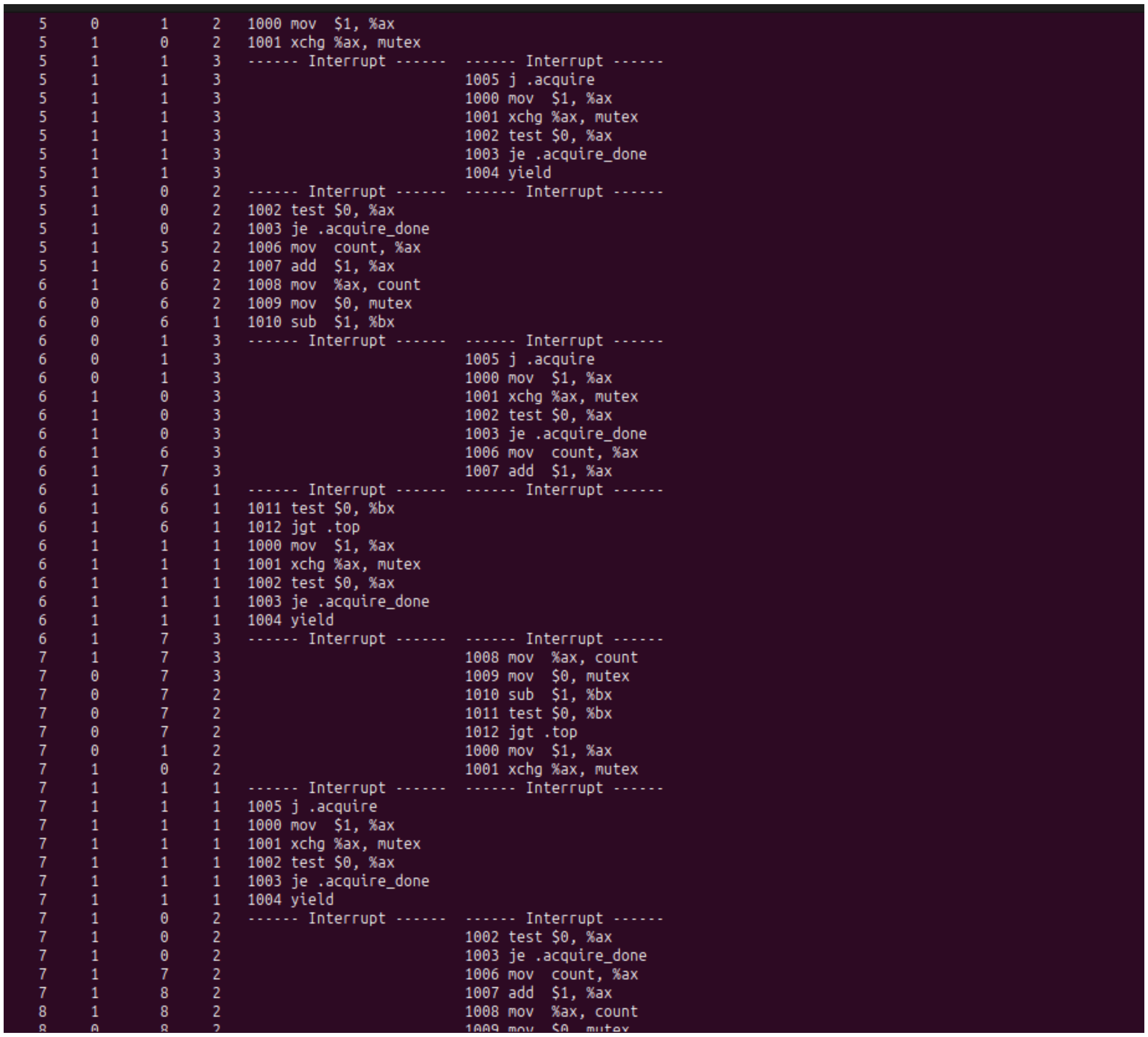


```
1993  1995  1993    1993     2  1992   ------ Interrupt ------  ------ Interrupt ------
1993  1995  1993    1993     2  1992   1004 jne .tryagain
1993  1995  1993    1993     2  1993   1002 mov turn, %cx
1993  1995  1993    1993     2  1993   1003 test %cx, %ax
1993  1995  1993    1993     2  1993   1004 jne .tryagain
1993  1995  1993    1993     2  1993   1005 mov  count, %ax
1993  1995  1993    1994     2  1993   1006 add  $1, %ax
1994  1995  1993    1994     2  1993   1007 mov  %ax, count
1994  1995  1993       1     2  1993   1008 mov $1, %ax
1994  1995  1994    1993     2  1993   1009 fetchadd %ax, turn
1994  1995  1994    1993     1  1993   1010 sub  $1, %bx
1994  1995  1994    1993     1  1993   1011 test $0, %bx
1994  1995  1994    1993     1  1993   1012 jgt .top
1994  1995  1994       1     1  1993   1000 mov $1, %ax
1994  1996  1994    1995     1  1993   1001 fetchadd %ax, ticket
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1995     1  1994   1002 mov turn, %cx
1994  1996  1994    1995     1  1994   1003 test %cx, %ax
1994  1996  1994    1995     1  1994   1004 jne .tryagain
1994  1996  1994    1994     5  1993   ------ Interrupt ------  ------ Interrupt ------
1994  1996  1994    1994     5  1993                            1004 jne .tryagain
1994  1996  1994    1994     5  1994                            1002 mov turn, %cx
1994  1996  1994    1994     5  1994                            1003 test %cx, %ax
```

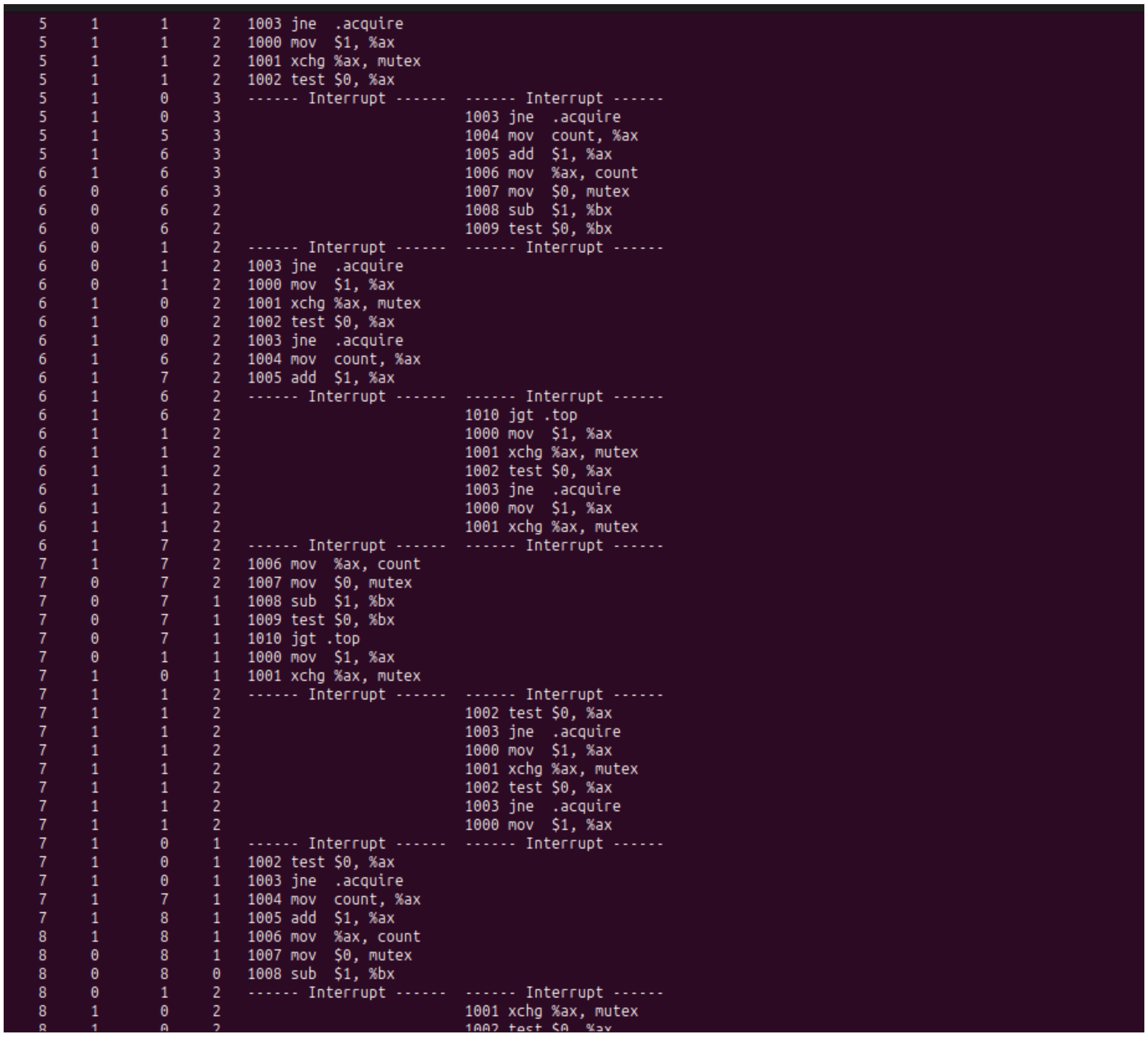
Yes ticket.s matches the code in the book. When run the threads spend a considerable amount of time waiting for the lock causing inefficiency in the cpu.

12. As you add more threads each thread will be stuck spinning for longer as the queue size will grow. This will cause more inefficiency for the cpu as many threads will be stuck spinning taking up computational space.

13.



```
    5     0       1     2   1000 mov  $1, %ax
    5     1       0     2   1001 xchg %ax, mutex
    5     1       1     3   ------ Interrupt ------  ------ Interrupt ------
    5     1       1     3                            1005 j .acquire
    5     1       1     3                            1000 mov  $1, %ax
    5     1       1     3                            1001 xchg %ax, mutex
    5     1       1     3                            1002 test $0, %ax
    5     1       1     3                            1003 je .acquire_done
    5     1       1     3                            1004 yield
    5     1       0     2   ------ Interrupt ------  ------ Interrupt ------
    5     1       0     2   1002 test $0, %ax
    5     1       0     2   1003 je .acquire_done
    5     1       5     2   1006 mov  count, %ax
    5     1       6     2   1007 add  $1, %ax
    6     1       6     2   1008 mov  %ax, count
    6     0       6     2   1009 mov  $0, mutex
    6     0       6     1   1010 sub  $1, %bx
    6     0       1     3   ------ Interrupt ------  ------ Interrupt ------
    6     0       1     3                            1005 j .acquire
    6     0       1     3                            1000 mov  $1, %ax
    6     1       0     3                            1001 xchg %ax, mutex
    6     1       0     3                            1002 test $0, %ax
    6     1       0     3                            1003 je .acquire_done
    6     1       6     3                            1006 mov  count, %ax
    6     1       7     3                            1007 add  $1, %ax
    6     1       6     1   ------ Interrupt ------  ------ Interrupt ------
    6     1       6     1   1011 test $0, %bx
    6     1       6     1   1012 jgt .top
    6     1       1     1   1000 mov  $1, %ax
    6     1       1     1   1001 xchg %ax, mutex
    6     1       1     1   1002 test $0, %ax
    6     1       1     1   1003 je .acquire_done
    6     1       1     1   1004 yield
    6     1       7     3   ------ Interrupt ------  ------ Interrupt ------
    7     1       7     3                            1008 mov  %ax, count
    7     0       7     3                            1009 mov  $0, mutex
    7     0       7     2                            1010 sub  $1, %bx
    7     0       7     2                            1011 test $0, %bx
    7     0       7     2                            1012 jgt .top
    7     0       1     2                            1000 mov  $1, %ax
    7     1       0     2                            1001 xchg %ax, mutex
    7     1       1     1   ------ Interrupt ------  ------ Interrupt ------
    7     1       1     1   1005 j .acquire
    7     1       1     1   1000 mov  $1, %ax
    7     1       1     1   1001 xchg %ax, mutex
    7     1       1     1   1002 test $0, %ax
    7     1       1     1   1003 je .acquire_done
    7     1       1     1   1004 yield
    7     1       0     2   ------ Interrupt ------  ------ Interrupt ------
    7     1       0     2                            1002 test $0, %ax
    7     1       0     2                            1003 je .acquire_done
    7     1       7     2                            1006 mov  count, %ax
    7     1       8     2                            1007 add  $1, %ax
    8     1       8     2                            1008 mov  %ax, count
    8     0       8     2                            1009 mov  $0, mutex
```



```
    5     1       1     2   1003 jne  .acquire
    5     1       1     2   1000 mov  $1, %ax
    5     1       1     2   1001 xchg %ax, mutex
    5     1       1     2   1002 test $0, %ax
    5     1       0     3   ------ Interrupt ------  ------ Interrupt ------
    5     1       0     3                           1003 jne  .acquire
    5     1       5     3                           1004 mov  count, %ax
    5     1       6     3                           1005 add  $1, %ax
    6     1       6     3                           1006 mov  %ax, count
    6     0       6     3                           1007 mov  $0, mutex
    6     0       6     2                           1008 sub  $1, %bx
    6     0       6     2                           1009 test $0, %bx
    6     0       1     2   ------ Interrupt ------  ------ Interrupt ------
    6     0       1     2   1003 jne  .acquire
    6     0       1     2   1000 mov  $1, %ax
    6     1       0     2   1001 xchg %ax, mutex
    6     1       0     2   1002 test $0, %ax
    6     1       0     2   1003 jne  .acquire
    6     1       6     2   1004 mov  count, %ax
    6     1       7     2   1005 add  $1, %ax
    6     1       6     2   ------ Interrupt ------  ------ Interrupt ------
    6     1       6     2                           1010 jgt .top
    6     1       1     2                           1000 mov  $1, %ax
    6     1       1     2                           1001 xchg %ax, mutex
    6     1       1     2                           1002 test $0, %ax
    6     1       1     2                           1003 jne  .acquire
    6     1       1     2                           1000 mov  $1, %ax
    6     1       1     2                           1001 xchg %ax, mutex
    6     1       7     2   ------ Interrupt ------  ------ Interrupt ------
    7     1       7     2   1006 mov  %ax, count
    7     0       7     2   1007 mov  $0, mutex
    7     0       7     1   1008 sub  $1, %bx
    7     0       7     1   1009 test $0, %bx
    7     0       7     1   1010 jgt .top
    7     0       1     1   1000 mov  $1, %ax
    7     1       0     1   1001 xchg %ax, mutex
    7     1       1     2   ------ Interrupt ------  ------ Interrupt ------
    7     1       1     2                           1002 test $0, %ax
    7     1       1     2                           1003 jne  .acquire
    7     1       1     2                           1000 mov  $1, %ax
    7     1       1     2                           1001 xchg %ax, mutex
    7     1       1     2                           1002 test $0, %ax
    7     1       1     2                           1003 jne  .acquire
    7     1       1     2                           1000 mov  $1, %ax
    7     1       0     1   ------ Interrupt ------  ------ Interrupt ------
    7     1       0     1   1002 test $0, %ax
    7     1       0     1   1003 jne  .acquire
    7     1       7     1   1004 mov  count, %ax
    7     1       8     1   1005 add  $1, %ax
    8     1       8     1   1006 mov  %ax, count
    8     0       8     1   1007 mov  $0, mutex
    8     0       8     0   1008 sub  $1, %bx
    8     0       1     2   ------ Interrupt ------  ------ Interrupt ------
    8     1       0     2                           1001 xchg %ax, mutex
    8     1       0     2                           1002 test $0, %ax
```

When setting bx to 5 and intervals to 7 we can see better results in yield.s than in test-and-set.s. This can be seen in the pictures above where the first picture is yield and the second picture is test-and-set. As you can see in the test-and-set run, each interval would run 7 lines of code then go to the next thread but in yield it would run until it hit a yield call which would be less than seven and then give the other thread time to run. This resulted in less waiting time for each thread and more efficient use of the cpu therefore running in less time.

14. This lock checks to see if the lock is available before exchanging 1 with the mutex. This helps efficiency compared to a regular test-and-set because the code isn't doing the exchange every loop therefore using less of the cpu continuously.