Chapter 28

1.
Line 4-5 marks the main entry and the top

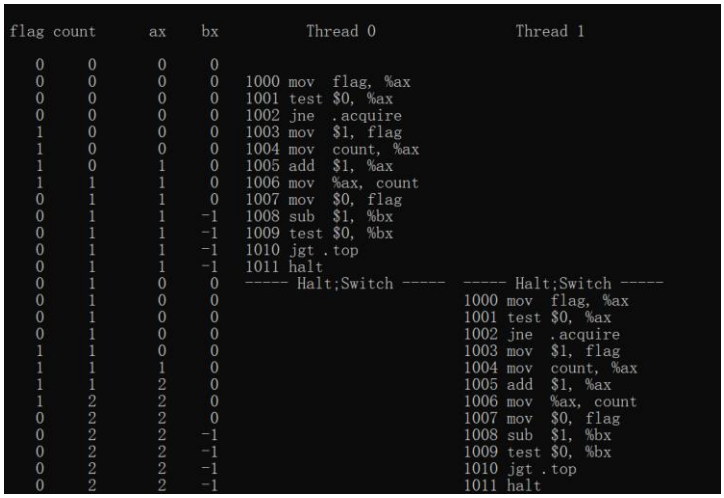
Line 7 marks acquire
Line 8 put the value of flag on %ax
Line 9 compare 0 with the value of %ax
Line 10 jump to tag .acquire if %ax != 0
Line 11 set the value of flag to 1

Line 14-16 increase the value of a count by 1

Line 19 set the value of flag to 0 (release lock)

Line 22-24 subtract %bx by 1, jump to .top tag if %bx != 0

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

3. Setting each %bx to 2 make each thread loops twice.
The flag is 0 finally.

4.
The correct result of count (critical section) should be 20 if set -a bx=10,bx=10.
So the following works correctly: -I 11,21,30…

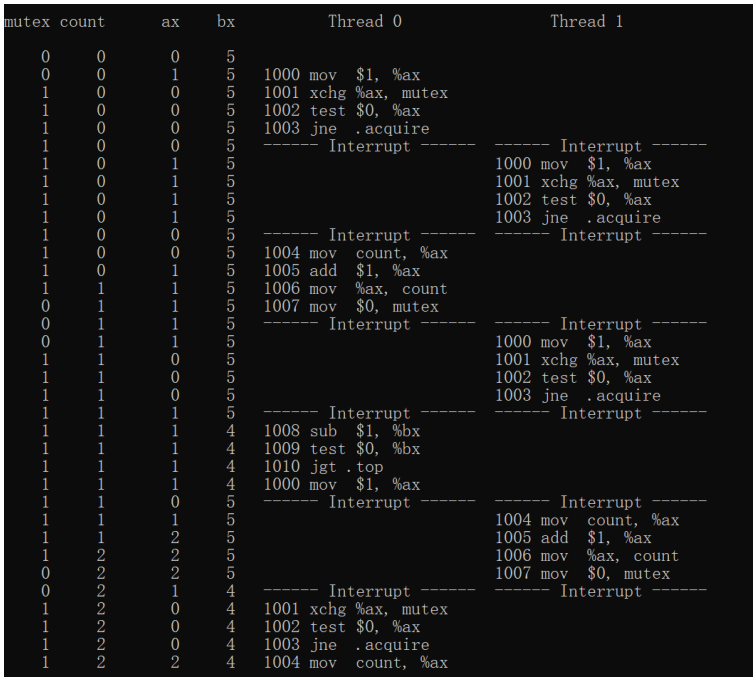
The following don't: -I 3,10,13,18,20…

5.
Lock acquire: xchg %ax, mutex (mutex = 1 now)
Lock release: mov $0, mutex

6. python x86.py -p test-and-set.s -M mutex,count -R ax,bx -a bx=10,bx=10 -i 3 -c
No matter how -I changes, the count is correct.

In some cases the cpu is not used efficiently, e.g. when a thread holds the lock and the other one wait for the release to acquire the lock.

7. e.g.
python x86.py -p test-and-set.s -M mutex,count -R ax,bx -a bx=5,bx=5 -i 4 -c



```
mutex count      ax    bx         Thread 0               Thread 1

   0     0       0     5
   0     0       1     5   1000 mov  $1, %ax
   1     0       0     5   1001 xchg %ax, mutex
   1     0       0     5   1002 test $0, %ax
   1     0       0     5   1003 jne  .acquire
   1     0       0     5   ------ Interrupt ------  ------ Interrupt ------
   1     0       1     5                           1000 mov  $1, %ax
   1     0       1     5                           1001 xchg %ax, mutex
   1     0       1     5                           1002 test $0, %ax
   1     0       1     5                           1003 jne  .acquire
   1     0       0     5   ------ Interrupt ------  ------ Interrupt ------
   1     0       0     5   1004 mov  count, %ax
   1     0       1     5   1005 add  $1, %ax
   1     1       1     5   1006 mov  %ax, count
   0     1       1     5   1007 mov  $0, mutex
   0     1       1     5   ------ Interrupt ------  ------ Interrupt ------
   0     1       1     5                           1000 mov  $1, %ax
   1     1       0     5                           1001 xchg %ax, mutex
   1     1       0     5                           1002 test $0, %ax
   1     1       0     5                           1003 jne  .acquire
   1     1       1     5   ------ Interrupt ------  ------ Interrupt ------
   1     1       1     4   1008 sub  $1, %bx
   1     1       1     4   1009 test $0, %bx
   1     1       1     4   1010 jgt .top
   1     1       1     4   1000 mov  $1, %ax
   1     1       0     5   ------ Interrupt ------  ------ Interrupt ------
   1     1       1     5                           1004 mov  count, %ax
   1     1       2     5                           1005 add  $1, %ax
   1     2       2     5                           1006 mov  %ax, count
   0     2       2     5                           1007 mov  $0, mutex
   0     2       1     4   ------ Interrupt ------  ------ Interrupt ------
   1     2       0     4   1001 xchg %ax, mutex
   1     2       0     4   1002 test $0, %ax
   1     2       0     4   1003 jne  .acquire
   1     2       2     4   1004 mov  count, %ax
```

In this case thread 1 waits for thread 0 to release the lock, so that thread 1 can acquire the lock. But the count is correct at the end.

8-9
python x86.py -p peterson.s -a bx=0,bx=1 -M count,flag -R ax,cx -c -i 3,6,9,12

11
python x86.py -p ticket.s -M ticket,count,turn -R ax,bx,cx -a bx=1000,bx=1000 -c

12. add more threads using -t

```
-t NUMTHREADS, --threads=NUMTHREADS
                        number of threads
```

The program will check more times about if it's your turn.
13. python x86.py -p yield.s -M count,mutex -R ax,bx -a bx=5,bx=5 -c -i 4

```
count mutex      ax    bx          Thread 0                Thread 1

    0     0       0     5
    0     0       1     5   1000 mov  $1, %ax
    0     1       0     5   1001 xchg %ax, mutex
    0     1       0     5   1002 test $0, %ax
    0     1       0     5   1003 je .acquire_done
    0     1       0     5   ------ Interrupt ------  ------ Interrupt ------
    0     1       1     5                           1000 mov  $1, %ax
    0     1       1     5                           1001 xchg %ax, mutex
    0     1       1     5                           1002 test $0, %ax
    0     1       1     5                           1003 je .acquire_done
    0     1       0     5   ------ Interrupt ------  ------ Interrupt ------
    0     1       0     5   1006 mov  count, %ax
    0     1       1     5   1007 add  $1, %ax
    1     1       1     5   1008 mov  %ax, count
    1     0       1     5   1009 mov  $0, mutex
    1     0       1     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       1     5                           1004 yield
    1     0       1     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       1     4   1010 sub  $1, %bx
    1     0       1     4   1011 test $0, %bx
    1     0       1     4   1012 jgt .top
    1     0       1     4   1000 mov  $1, %ax
    1     0       1     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       1     5                           1005 j .acquire
    1     0       1     5                           1000 mov  $1, %ax
    1     1       0     5                           1001 xchg %ax, mutex
    1     1       0     5                           1002 test $0, %ax
    1     1       1     4   ------ Interrupt ------  ------ Interrupt ------
    1     1       1     4   1001 xchg %ax, mutex
    1     1       1     4   1002 test $0, %ax
    1     1       1     4   1003 je .acquire_done
    1     1       1     4   1004 yield
    1     1       0     5   ------ Interrupt ------  ------ Interrupt ------
    1     1       0     5                           1003 je .acquire_done
    1     1       1     5                           1006 mov  count, %ax
```

Compared to the image (test-set.s) in question 7, when -I 4, yield.s effectively uses cpu by letting a thread yielding the cpu when it starts to wait.

Scenario: a thread needs to wait for another thread to release the lock, when the thread starts to wait it releases the cpu.

14. python x86.py -p test-and-test-and-set.s -M mutex,count -R ax,bx -c -a bx=5,bx=5 -i 4

```
mutex count      ax    bx          Thread 0                Thread 1

    0     0       0     5
    0     0       0     5   1000 mov  mutex, %ax
    0     0       0     5   1001 test $0, %ax
    0     0       0     5   1002 jne .acquire
    0     0       1     5   1003 mov  $1, %ax
    0     0       0     5   ------ Interrupt ------  ------ Interrupt ------
    0     0       0     5                           1000 mov  mutex, %ax
    0     0       0     5                           1001 test $0, %ax
    0     0       0     5                           1002 jne .acquire
    0     0       1     5                           1003 mov  $1, %ax
    0     0       1     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       0     5   1004 xchg %ax, mutex
    1     0       0     5   1005 test $0, %ax
    1     0       0     5   1006 jne .acquire
    1     0       0     5   1007 mov  count, %ax
    1     0       1     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       1     5                           1004 xchg %ax, mutex
    1     0       1     5                           1005 test $0, %ax
    1     0       1     5                           1006 jne .acquire
    1     0       1     5                           1000 mov  mutex, %ax
    1     0       0     5   ------ Interrupt ------  ------ Interrupt ------
    1     0       1     5   1008 add  $1, %ax
    1     1       1     5   1009 mov  %ax, count
    0     1       1     5   1010 mov  $0, mutex
    0     1       1     4   1011 sub  $1, %bx
    0     1       1     5   ------ Interrupt ------  ------ Interrupt ------
    0     1       1     5                           1001 test $0, %ax
    0     1       1     5                           1002 jne .acquire
    0     1       0     5                           1000 mov  mutex, %ax
    0     1       0     5                           1001 test $0, %ax
    0     1       1     4   ------ Interrupt ------  ------ Interrupt ------
    0     1       1     4   1012 test $0, %bx
    0     1       1     4   1013 jgt .top
    0     1       0     4   1000 mov  mutex, %ax
    0     1       0     4   1001 test $0, %ax
    0     1       0     5   ------ Interrupt ------  ------ Interrupt ------
```