OS 2022fall 10.8 hw4

20307140008 施君豪

运行环境：Windows 10 (pycharm)

1. **第二十八章**
2. **通过flag是否置为1表示锁是否正被占用（取锁：flag=1，解锁：flag=0）**

**在上锁的时候对count进行加一操作，通过%bx的值判断循环是否结束。**

**（下为flag.s的代码及注释）**

.main

.top

.acquire

mov flag, %ax # 取flag的值

test $0, %ax # 若值为0，意味着未上锁

jne .acquire # 若不为零，循环等待

mov $1, flag # 将flag值置为1（上锁）

# critical section

mov count, %ax # 取count的值

add $1, %ax # 加一

mov %ax, count # 存回

# release lock

mov $0, flag # 将flag值置为0（解锁）

# see if we're still looping

sub $1, %bx

test $0, %bx

jgt .top

1. **正常工作，count最终为正确的结果(count=2)，flag最终为0（没有进程占用锁）**

**执行代码为x86.py -p flag.s -M flag,count -c，以下为结果：**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p flag.s -M flag,count -c

flag count Thread 0 Thread 1

0 0

0 0 1000 mov flag, %ax

0 0 1001 test $0, %ax

0 0 1002 jne .acquire

1 0 1003 mov $1, flag

1 0 1004 mov count, %ax

1 0 1005 add $1, %ax

1 1 1006 mov %ax, count

0 1 1007 mov $0, flag

0 1 1008 sub $1, %bx

0 1 1009 test $0, %bx

0 1 1010 jgt .top

0 1 1011 halt

0 1 ----- Halt;Switch ----- ----- Halt;Switch -----

0 1 1000 mov flag, %ax

0 1 1001 test $0, %ax

0 1 1002 jne .acquire

1 1 1003 mov $1, flag

1 1 1004 mov count, %ax

1 1 1005 add $1, %ax

1 2 1006 mov %ax, count

0 2 1007 mov $0, flag

0 2 1008 sub $1, %bx

0 2 1009 test $0, %bx

0 2 1010 jgt .top

1. 2 1011 halt

**3、 -a bx=2,bx=2指将thread0，thread1中%bx的值置为2，即两个进程中count的加法均执行两次。以下为x86.py中对-a功能的说明：**

parser.add\_option('-a', '--argv', default='', help='comma-separated per-thread args (e.g., ax=1,ax=2 sets thread 0 ax reg to 1 and thread 1 ax reg to 2); specify multiple regs per thread via colon-separated list (e.g., ax=1:bx=2,cx=3 sets thread 0 ax and bx and just cx for thread 1)', action='store', type='string', dest='argv')

**正常工作，count最终为正确的结果(count = 4)，flag最终为0（没有进程占用锁）**

**执行代码为x86.py -p flag.s -a bx=2,bx=2 -M flag,count -c，以下为结果：**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p flag.s -a bx=2,bx=2 -M flag,count -c

flag count Thread 0 Thread 1

0 0

0 0 1000 mov flag, %ax

0 0 1001 test $0, %ax

0 0 1002 jne .acquire

1 0 1003 mov $1, flag

1 0 1004 mov count, %ax

1 0 1005 add $1, %ax

1 1 1006 mov %ax, count

0 1 1007 mov $0, flag

0 1 1008 sub $1, %bx

0 1 1009 test $0, %bx

0 1 1010 jgt .top

0 1 1000 mov flag, %ax

0 1 1001 test $0, %ax

0 1 1002 jne .acquire

1 1 1003 mov $1, flag

1 1 1004 mov count, %ax

1 1 1005 add $1, %ax

1 2 1006 mov %ax, count

0 2 1007 mov $0, flag

0 2 1008 sub $1, %bx

0 2 1009 test $0, %bx

0 2 1010 jgt .top

0 2 1011 halt

0 2 ----- Halt;Switch ----- ----- Halt;Switch -----

0 2 1000 mov flag, %ax

0 2 1001 test $0, %ax

0 2 1002 jne .acquire

1 2 1003 mov $1, flag

1 2 1004 mov count, %ax

1 2 1005 add $1, %ax

1 3 1006 mov %ax, count

0 3 1007 mov $0, flag

0 3 1008 sub $1, %bx

0 3 1009 test $0, %bx

0 3 1010 jgt .top

0 3 1000 mov flag, %ax

0 3 1001 test $0, %ax

0 3 1002 jne .acquire

1 3 1003 mov $1, flag

1 3 1004 mov count, %ax

1 3 1005 add $1, %ax

1 4 1006 mov %ax, count

0 4 1007 mov $0, flag

0 4 1008 sub $1, %bx

0 4 1009 test $0, %bx

0 4 1010 jgt .top

1. 4 1011 halt

**4、 当i的值较小时，可能出现下面情况：**

**线程0执行 mov flag, %ax 完后,时钟终端,切到线程1执行,**

**而线程1在执行 mov %ax, count 中断,切到线程0,此时线程1是拥有锁的,**

**线程0继续执行 test $0, %ax ,这时ax的值是0, 所以线程1也获得了锁，导致最终得到错误的结果。**

**（下图为i = 4 时的情形，i = 1,2,3,4,6,7,9均可导致错误结果，其余值结果正常）**

Flag count ax bx Thread 0 Thread 1

0 1 0 1 1000 mov flag, %ax

0 1 0 2 ------ Interrupt ------ ------ Interrupt ------

0 1 0 2 1002 jne .acquire

1 1 0 2 1003 mov $1, flag

1 1 1 2 1004 mov count, %ax

1 1 2 2 1005 add $1, %ax

1 1 0 1 ------ Interrupt ------ ------ Interrupt ------

1 1 0 1 1001 test $0, %ax

1 1 0 1 1002 jne .acquire

1 1 0 1 1003 mov $1, flag

**执行代码为x86.py -p flag.s -a bx=2,bx=2 -M flag,count -c -R ax,bx -i 4 ，以下为结果：**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p flag.s -a bx=2,bx=2 -M flag,count -c -R ax,bx -i 4

flag count ax bx Thread 0 Thread 1

0 0 0 2

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 ------ Interrupt ------ ------ Interrupt ------

1 0 1 2 1000 mov flag, %ax

1 0 1 2 1001 test $0, %ax

1 0 1 2 1002 jne .acquire

1 0 1 2 1000 mov flag, %ax

1 0 0 2 ------ Interrupt ------ ------ Interrupt ------

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 2 ------ Interrupt ------ ------ Interrupt ------

0 1 1 2 1001 test $0, %ax

0 1 1 2 1002 jne .acquire

0 1 0 2 1000 mov flag, %ax

0 1 0 2 1001 test $0, %ax

0 1 1 2 ------ Interrupt ------ ------ Interrupt ------

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 2 ------ Interrupt ------ ------ Interrupt ------

0 1 0 2 1002 jne .acquire

1 1 0 2 1003 mov $1, flag

1 1 1 2 1004 mov count, %ax

1 1 2 2 1005 add $1, %ax

1 1 0 1 ------ Interrupt ------ ------ Interrupt ------

1 1 0 1 1001 test $0, %ax

1 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 2 ------ Interrupt ------ ------ Interrupt ------

1 2 2 2 1006 mov %ax, count

0 2 2 2 1007 mov $0, flag

0 2 2 1 1008 sub $1, %bx

0 2 2 1 1009 test $0, %bx

0 2 1 1 ------ Interrupt ------ ------ Interrupt ------

0 2 2 1 1005 add $1, %ax

0 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 1 ------ Interrupt ------ ------ Interrupt ------

0 2 2 1 1010 jgt .top

0 2 0 1 1000 mov flag, %ax

0 2 0 1 1001 test $0, %ax

0 2 0 1 1002 jne .acquire

0 2 2 0 ------ Interrupt ------ ------ Interrupt ------

0 2 2 0 1009 test $0, %bx

0 2 2 0 1010 jgt .top

0 2 2 0 1011 halt

0 2 0 1 ----- Halt;Switch ----- ----- Halt;Switch -----

1 2 0 1 1003 mov $1, flag

1 2 0 1 ------ Interrupt ------ ------ Interrupt ------

1 2 2 1 1004 mov count, %ax

1 2 3 1 1005 add $1, %ax

1 3 3 1 1006 mov %ax, count

0 3 3 1 1007 mov $0, flag

0 3 3 1 ------ Interrupt ------ ------ Interrupt ------

0 3 3 0 1008 sub $1, %bx

0 3 3 0 1009 test $0, %bx

0 3 3 0 1010 jgt .top

0 3 3 0 1011 halt

**5、通过1和mutex互换的结果是否为1表示锁是否正被占用**

**未锁：返回0，同时mutex置为1，已取得锁，**

**已锁：返回1，mutex仍然为1，不会影响目前的锁**

**同时通过%bx的值判断循环是否结束。**

**（下为test-and-set.s的代码及注释）**

.main

.top

.acquire

mov $1, %ax

xchg %ax, mutex # 原子交换1与mutex

test $0, %ax # 若值为0，意味着未上锁

jne .acquire # 若不为零，循环等待

# critical section

mov count, %ax # 取count的值

add $1, %ax # 加一

mov %ax, count # 存回

# release lock

mov $0, mutex

# see if we're still looping

sub $1, %bx

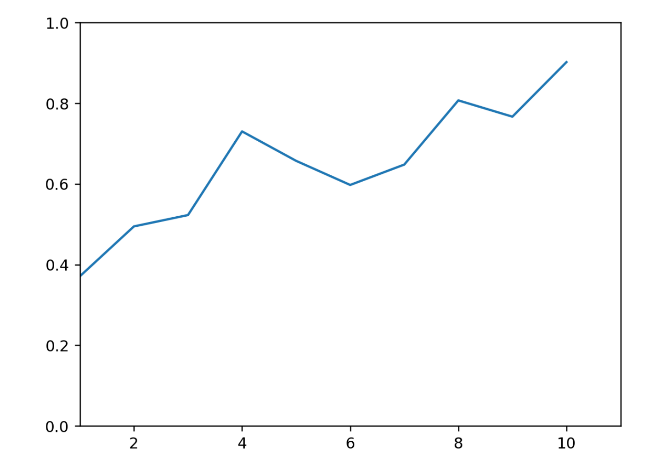
test $0, %bx

jgt .top

**6、总按预期工作，结果均与预期相符。若i过小（中断过于频繁），会产生当一个线程持有锁进入临界区时被抢占的情形, 抢占的线程将会自旋一个时间片,导致CPU的使用率相对较低。**

**量化标准：CPU使用率 = 有效使用占比 = 1 - （无效自旋占用时间/总时间）**

**结果图如下：（横轴为i，纵轴为CPU使用率）**

**结果为折线的原因为由于i的间隔不同，触发线程持有锁进入临界区时被抢占的情形的次数不同，总体趋势为i越小CPU使用率越低。**

**7、无论是何时尝试获取锁，一旦一个进程获取了锁另一个进程则必然无法成功。**

**执行代码为x86.py -p test-and-set.s -i 10 -R ax,bx -M mutex,count -a bx=3 -P 1100111000111000111000 -c，以下为结果：**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p test-and-set.s -i 10 -R ax,bx -M mutex,count -a bx=3 -P 1100111000111000111000 -c

Mutex count ax bx Thread 0 Thread 1

0 0 0 3

0 0 1 3 1000 mov $1, %ax

1 0 0 3 1001 xchg %ax, mutex

1 0 0 3 ------ Interrupt ------ ------ Interrupt ------

1 0 1 3 1000 mov $1, %ax

1 0 1 3 1001 xchg %ax, mutex

1 0 0 3 ------ Interrupt ------ ------ Interrupt ------

1 0 0 3 1002 test $0, %ax

1 0 0 3 1003 jne .acquire

1 0 0 3 1004 mov count, %ax

1 0 1 3 ------ Interrupt ------ ------ Interrupt ------

1 0 1 3 1002 test $0, %ax

1 0 1 3 1003 jne .acquire

1 0 1 3 1000 mov $1, %ax

1 0 0 3 ------ Interrupt ------ ------ Interrupt ------

1 0 1 3 1005 add $1, %ax

1 1 1 3 1006 mov %ax, count

0 1 1 3 1007 mov $0, mutex

0 1 1 3 ------ Interrupt ------ ------ Interrupt ------

1 1 0 3 1001 xchg %ax, mutex

1 1 0 3 1002 test $0, %ax

1 1 0 3 1003 jne .acquire

1 1 1 3 ------ Interrupt ------ ------ Interrupt ------

1 1 1 2 1008 sub $1, %bx

1 1 1 2 1009 test $0, %bx

1 1 1 2 1010 jgt .top

1 1 0 3 ------ Interrupt ------ ------ Interrupt ------

1 1 1 3 1004 mov count, %ax

1 1 2 3 1005 add $1, %ax

1 2 2 3 1006 mov %ax, count

1 2 1 2 ------ Interrupt ------ ------ Interrupt ------

1 2 1 2 1000 mov $1, %ax

1 2 1 2 1001 xchg %ax, mutex

1 2 2 3 ------ Interrupt ------ ------ Interrupt ------

0 2 2 3 1007 mov $0, mutex

0 2 2 2 1008 sub $1, %bx

0 2 1 2 ------ Interrupt ------ ------ Interrupt ------

0 2 1 2 1002 test $0, %ax

0 2 1 2 1003 jne .acquire

0 2 1 2 1000 mov $1, %ax

0 2 2 2 ------ Interrupt ------ ------ Interrupt ------

0 2 2 2 1009 test $0, %bx

0 2 2 2 1010 jgt .top

0 2 1 2 1000 mov $1, %ax

0 2 1 2 ------ Interrupt ------ ------ Interrupt ------

1 2 0 2 1001 xchg %ax, mutex

1 2 0 2 1002 test $0, %ax

1 2 0 2 1003 jne .acquire

1 2 1 2 ------ Interrupt ------ ------ Interrupt ------

1 2 1 2 1001 xchg %ax, mutex

1 2 1 2 1002 test $0, %ax

1 2 1 2 1003 jne .acquire

1 2 0 2 ------ Interrupt ------ ------ Interrupt ------

1 2 2 2 1004 mov count, %ax

1 2 3 2 1005 add $1, %ax

1 3 3 2 1006 mov %ax, count

1 3 1 2 ------ Interrupt ------ ------ Interrupt ------

1 3 1 2 1000 mov $1, %ax

1 3 1 2 1001 xchg %ax, mutex

1 3 1 2 1002 test $0, %ax

1 3 3 2 ------ Interrupt ------ ------ Interrupt ------

0 3 3 2 1007 mov $0, mutex

0 3 3 1 1008 sub $1, %bx

0 3 1 2 ------ Interrupt ------ ------ Interrupt ------

0 3 1 2 1003 jne .acquire

0 3 1 2 1000 mov $1, %ax

0 3 3 1 ------ Interrupt ------ ------ Interrupt ------

0 3 3 1 1009 test $0, %bx

0 3 3 1 1010 jgt .top

0 3 1 1 1000 mov $1, %ax

0 3 1 2 ------ Interrupt ------ ------ Interrupt ------

1 3 0 2 1001 xchg %ax, mutex

1 3 0 2 1002 test $0, %ax

1 3 0 2 1003 jne .acquire

1 3 1 1 ------ Interrupt ------ ------ Interrupt ------

1 3 1 1 1001 xchg %ax, mutex

1 3 1 1 1002 test $0, %ax

1 3 1 1 1003 jne .acquire

1 3 0 2 ------ Interrupt ------ ------ Interrupt ------

1 3 3 2 1004 mov count, %ax

1 3 4 2 1005 add $1, %ax

1 4 4 2 1006 mov %ax, count

1 4 1 1 ------ Interrupt ------ ------ Interrupt ------

1 4 1 1 1000 mov $1, %ax

1 4 1 1 1001 xchg %ax, mutex

1 4 1 1 1002 test $0, %ax

1 4 4 2 ------ Interrupt ------ ------ Interrupt ------

0 4 4 2 1007 mov $0, mutex

0 4 4 1 1008 sub $1, %bx

0 4 4 1 1009 test $0, %bx

0 4 1 1 ------ Interrupt ------ ------ Interrupt ------

0 4 1 1 1003 jne .acquire

0 4 1 1 1000 mov $1, %ax

0 4 4 1 ------ Interrupt ------ ------ Interrupt ------

0 4 4 1 1010 jgt .top

0 4 1 1 1000 mov $1, %ax

0 4 1 1 ------ Interrupt ------ ------ Interrupt ------

1 4 0 1 1001 xchg %ax, mutex

1 4 0 1 1002 test $0, %ax

1 4 0 1 1003 jne .acquire

1 4 1 1 ------ Interrupt ------ ------ Interrupt ------

1 4 1 1 1001 xchg %ax, mutex

1 4 1 1 1002 test $0, %ax

1 4 1 1 1003 jne .acquire

1 4 0 1 ------ Interrupt ------ ------ Interrupt ------

1 4 4 1 1004 mov count, %ax

1 4 5 1 1005 add $1, %ax

1 5 5 1 1006 mov %ax, count

1 5 1 1 ------ Interrupt ------ ------ Interrupt ------

1 5 1 1 1000 mov $1, %ax

1 5 1 1 1001 xchg %ax, mutex

1 5 1 1 1002 test $0, %ax

1 5 5 1 ------ Interrupt ------ ------ Interrupt ------

0 5 5 1 1007 mov $0, mutex

0 5 5 0 1008 sub $1, %bx

0 5 5 0 1009 test $0, %bx

0 5 1 1 ------ Interrupt ------ ------ Interrupt ------

0 5 1 1 1003 jne .acquire

0 5 1 1 1000 mov $1, %ax

1 5 0 1 1001 xchg %ax, mutex

1 5 5 0 ------ Interrupt ------ ------ Interrupt ------

1 5 5 0 1010 jgt .top

1 5 5 0 1011 halt

1 5 0 1 ----- Halt;Switch ----- ----- Halt;Switch -----

1 5 0 1 1002 test $0, %ax

1 5 0 1 1003 jne .acquire

1 5 5 1 1004 mov count, %ax

1 5 6 1 1005 add $1, %ax

1 6 6 1 1006 mov %ax, count

0 6 6 1 1007 mov $0, mutex

0 6 6 0 1008 sub $1, %bx

0 6 6 0 1009 test $0, %bx

0 6 6 0 1010 jgt .top

0 6 6 0 1011 halt

**8、Peterson.s即为书P222页上C语言代码的汇编实现，具体已在下方备注**

**（下为Peterosn.s的代码及注释）**

.main

# put address of flag into fx

lea flag, %fx

# assume thread ID is in bx (0 or 1, scale by 4 to get proper flag address)

mov %bx, %cx # bx为自己的id（self），复制进cx中

neg %cx # cx取反

add $1, %cx #（带上上一行）cx = 1- cx

.acquire

mov $1, 0(%fx,%bx,4) # flag[self] = 1

mov %cx, turn # turn = 1 - self

.spin1

mov 0(%fx,%cx,4), %ax # flag[1-self]

test $1, %ax

jne .fini # 如果 flag[1-self] != 1, 跳出循环至下方fini’处

.spin2 # just labeled for fun, not needed

mov turn, %ax

test %cx, %ax # compare 'turn' and '1 - self'

je .spin1 # if turn==1-self, go back and start spin again

# fall out of spin

.fini

# do critical section now

mov count, %ax

add $1, %ax

mov %ax, count

.release

mov $0, 0(%fx,%bx,4) # flag[self] = 0

# end case: make sure it's other's turn

mov %cx, turn # turn = 1 - self

**9、需要使用-a bx=0,bx=1对两个进程的bx赋初值，否则两个进程的均认为自己为0号位，检测1号位是否上锁，锁失效。除此以外，无论i取任何值，Peterson算法均能正常运行，得出正确结果。**

**10、分别使线程0和线程1循环等待，同时在另一个进程拥有锁时尝试获取锁，结果锁均运行完好，成功运行并得出正确结果。**

**执行代码分别为：**

**x86.py -p peterson.s -M turn,count -R bx -a bx=0,bx=1 -P 0000011111 -c，**

**x86.py -p peterson.s -M turn,count -R bx -a bx=0,bx=1 -P 00000011111 -c**

**以下为结果：**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p peterson.s -M turn,count -R bx -a bx=0,bx=1 -P 0000011111 -c

turn count bx Thread 0 Thread 1

0 0 0

0 0 0 1000 lea flag, %fx

0 0 0 1001 mov %bx, %cx

0 0 0 1002 neg %cx

0 0 0 1003 add $1, %cx

0 0 0 1004 mov $1, 0(%fx,%bx,4)

0 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1000 lea flag, %fx

0 0 1 1001 mov %bx, %cx

0 0 1 1002 neg %cx

0 0 1 1003 add $1, %cx

0 0 1 1004 mov $1, 0(%fx,%bx,4)

0 0 0 ------ Interrupt ------ ------ Interrupt ------

1 0 0 1005 mov %cx, turn

1 0 0 1006 mov 0(%fx,%cx,4), %ax

1 0 0 1007 test $1, %ax

1 0 0 1008 jne .fini

1 0 0 1009 mov turn, %ax

1 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1005 mov %cx, turn

0 0 1 1006 mov 0(%fx,%cx,4), %ax

0 0 1 1007 test $1, %ax

0 0 1 1008 jne .fini

0 0 1 1009 mov turn, %ax

0 0 0 ------ Interrupt ------ ------ Interrupt ------

0 0 0 1010 test %cx, %ax

0 0 0 1011 je .spin1

0 0 0 1006 mov 0(%fx,%cx,4), %ax

0 0 0 1007 test $1, %ax

0 0 0 1008 jne .fini

0 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1010 test %cx, %ax

0 0 1 1011 je .spin1

0 0 1 1006 mov 0(%fx,%cx,4), %ax

0 0 1 1007 test $1, %ax

0 0 1 1008 jne .fini

0 0 0 ------ Interrupt ------ ------ Interrupt ------

0 0 0 1009 mov turn, %ax

0 0 0 1010 test %cx, %ax

0 0 0 1011 je .spin1

0 0 0 1012 mov count, %ax

0 0 0 1013 add $1, %ax

0 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1009 mov turn, %ax

0 0 1 1010 test %cx, %ax

0 0 1 1011 je .spin1

0 0 1 1006 mov 0(%fx,%cx,4), %ax

0 0 1 1007 test $1, %ax

0 0 0 ------ Interrupt ------ ------ Interrupt ------

0 1 0 1014 mov %ax, count

0 1 0 1015 mov $0, 0(%fx,%bx,4)

1 1 0 1016 mov %cx, turn

1 1 0 1017 halt

1 1 1 ----- Halt;Switch ----- ----- Halt;Switch -----

1 1 1 1008 jne .fini

1 1 1 1009 mov turn, %ax

1 1 1 1010 test %cx, %ax

1 1 1 1011 je .spin1

1 1 1 1012 mov count, %ax

1 1 1 1013 add $1, %ax

1 2 1 1014 mov %ax, count

1 2 1 1015 mov $0, 0(%fx,%bx,4)

0 2 1 1016 mov %cx, turn

0 2 1 1017 halt

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p peterson.s -M turn,count -R bx -a bx=0,bx=1 -P 00000011111 -c

turn count bx Thread 0 Thread 1

0 0 0

0 0 0 1000 lea flag, %fx

0 0 0 1001 mov %bx, %cx

0 0 0 1002 neg %cx

0 0 0 1003 add $1, %cx

0 0 0 1004 mov $1, 0(%fx,%bx,4)

1 0 0 1005 mov %cx, turn

1 0 1 ------ Interrupt ------ ------ Interrupt ------

1 0 1 1000 lea flag, %fx

1 0 1 1001 mov %bx, %cx

1 0 1 1002 neg %cx

1 0 1 1003 add $1, %cx

1 0 1 1004 mov $1, 0(%fx,%bx,4)

1 0 0 ------ Interrupt ------ ------ Interrupt ------

1 0 0 1006 mov 0(%fx,%cx,4), %ax

1 0 0 1007 test $1, %ax

1 0 0 1008 jne .fini

1 0 0 1009 mov turn, %ax

1 0 0 1010 test %cx, %ax

1 0 0 1011 je .spin1

1 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1005 mov %cx, turn

0 0 1 1006 mov 0(%fx,%cx,4), %ax

0 0 1 1007 test $1, %ax

0 0 1 1008 jne .fini

0 0 1 1009 mov turn, %ax

0 0 0 ------ Interrupt ------ ------ Interrupt ------

0 0 0 1006 mov 0(%fx,%cx,4), %ax

0 0 0 1007 test $1, %ax

0 0 0 1008 jne .fini

0 0 0 1009 mov turn, %ax

0 0 0 1010 test %cx, %ax

0 0 0 1011 je .spin1

0 0 1 ------ Interrupt ------ ------ Interrupt ------

0 0 1 1010 test %cx, %ax

0 0 1 1011 je .spin1

0 0 1 1006 mov 0(%fx,%cx,4), %ax

0 0 1 1007 test $1, %ax

0 0 1 1008 jne .fini

0 0 0 ------ Interrupt ------ ------ Interrupt ------

0 0 0 1012 mov count, %ax

0 0 0 1013 add $1, %ax

0 1 0 1014 mov %ax, count

0 1 0 1015 mov $0, 0(%fx,%bx,4)

1 1 0 1016 mov %cx, turn

1 1 0 1017 halt

1 1 1 ----- Halt;Switch ----- ----- Halt;Switch -----

1 1 1 1009 mov turn, %ax

1 1 1 1010 test %cx, %ax

1 1 1 1011 je .spin1

1 1 1 1012 mov count, %ax

1 1 1 1013 add $1, %ax

1 2 1 1014 mov %ax, count

1 2 1 1015 mov $0, 0(%fx,%bx,4)

0 2 1 1016 mov %cx, turn

0 2 1 1017 halt

**11、tickets.s中的代码为书上代码的汇编实现，与书上代码相符**

**（下为tickets.s的代码与注释）**

.main

.top

.acquire

mov $1, %ax

fetchadd %ax, ticket # 取一张彩票（已有总数+1）

.tryagain

mov turn, %cx # 查看目前序列是否轮到自己

test %cx, %ax

jne .tryagain

# critical section

mov count, %ax #取count的值

add $1, %ax #加一

mov %ax, count #存回

# release lock

mov $1, %ax

fetchadd %ax, turn #turn加一，从而下一个ticket能够知道到自己了

# see if we're still looping

sub $1, %bx

test $0, %bx

jgt .top

**12、是的。随着时间推移，在每次中断前进程会先完成加一操作，然后由于对方进程持有下一张ticket，故只能一直无意义自旋等待对方释放ticket（即直到下次中断轮到自己前无法进行其他操作）**

**执行代码为：x86.py -p ticket.s -a bx=1000,bx=1000 -c -M count，以下为部分结果：**

count Thread 0 Thread 1

1989 1002 mov turn, %cx

1989 1003 test %cx, %ax

1989 1004 jne .tryagain

1989 1005 mov count, %ax

1989 1006 add $1, %ax

1990 1007 mov %ax, count

1990 1008 mov $1, %ax

1990 1009 fetchadd %ax, turn

1990 1010 sub $1, %bx

1990 1011 test $0, %bx

1990 1012 jgt .top

1990 1000 mov $1, %ax

1990 1001 fetchadd %ax, ticket

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

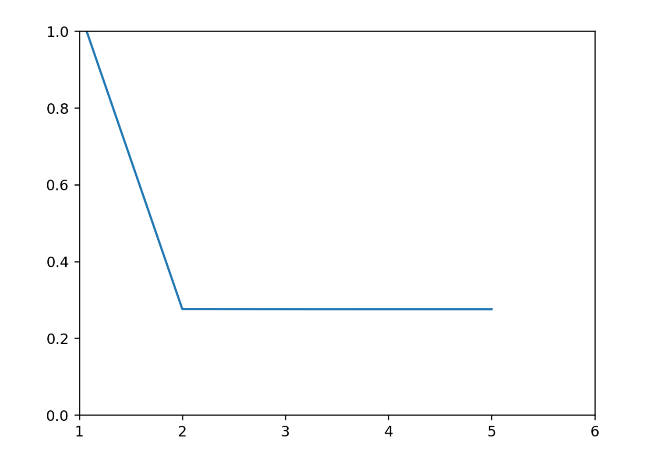
1990 1002 mov turn, %cx

1990 1003 test %cx, %ax

1990 1004 jne .tryagain

1990 1002 mov turn, %cx

**13、随着增加更多线程，线程仍然像上题一样花许多时间自旋等待锁。经过测试，除了单线程由于不用等待自旋CPU使用率最高，从双线程开始，线程开始花时间自旋CPU使用率降低。随着线程数的增多，CPU使用率没有发生改变。**

**下图横轴为CPU进程数，纵轴为CPU使用率。**

**14、节省了另一个进程持有锁时中断后多次自旋等待锁的时间。在yield.s中，单次循环发现锁目前被进程所占用后，即通过yield释放，省去了test-and-set.s中继续循环直至下一次中断的过程。**

**执行代码为：x86.py -p test-and-set.s -M count -a bx=10,bx=10 -i 5 -c，**

**x86.py -p yield.s -M count,mutex -a bx=10,bx=10 -i 5 -c**

**以下为部分结果：**

**（将i设置为更大的值会更为明显，但篇幅实在太大，还请谅解）**

**（yield.s节省处已用荧光标出）**

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p test-and-set.s -M count -a bx=10,bx=10 -i 5 -c

count Thread 0 Thread 1

0

0 1000 mov $1, %ax

0 1001 xchg %ax, mutex

0 1002 test $0, %ax

0 1003 jne .acquire

0 1004 mov count, %ax

0 ------ Interrupt ------ ------ Interrupt ------

0 1000 mov $1, %ax

0 1001 xchg %ax, mutex

0 1002 test $0, %ax

0 1003 jne .acquire

0 1000 mov $1, %ax

0 ------ Interrupt ------ ------ Interrupt ------

0 1005 add $1, %ax

1 1006 mov %ax, count

1 1007 mov $0, mutex

1 1008 sub $1, %bx

1 1009 test $0, %bx

1 ------ Interrupt ------ ------ Interrupt ------

1 1001 xchg %ax, mutex

1 1002 test $0, %ax

1 1003 jne .acquire

1 1004 mov count, %ax

1 1005 add $1, %ax

1 ------ Interrupt ------ ------ Interrupt ------

1 1010 jgt .top

1 1000 mov $1, %ax

1 1001 xchg %ax, mutex

1 1002 test $0, %ax

1 1003 jne .acquire

1 ------ Interrupt ------ ------ Interrupt ------

2 1006 mov %ax, count

2 1007 mov $0, mutex

2 1008 sub $1, %bx

2 1009 test $0, %bx

2 1010 jgt .top

2 ------ Interrupt ------ ------ Interrupt ------

2 1000 mov $1, %ax

2 1001 xchg %ax, mutex

2 1002 test $0, %ax

2 1003 jne .acquire

2 1004 mov count, %ax

2 ------ Interrupt ------ ------ Interrupt ------

2 1000 mov $1, %ax

2 1001 xchg %ax, mutex

2 1002 test $0, %ax

2 1003 jne .acquire

2 1000 mov $1, %ax

2 ------ Interrupt ------ ------ Interrupt ------

2 1005 add $1, %ax

3 1006 mov %ax, count

3 1007 mov $0, mutex

3 1008 sub $1, %bx

3 1009 test $0, %bx

3 ------ Interrupt ------ ------ Interrupt ------

3 1001 xchg %ax, mutex

3 1002 test $0, %ax

3 1003 jne .acquire

3 1004 mov count, %ax

3 1005 add $1, %ax

3 ------ Interrupt ------ ------ Interrupt ------

3 1010 jgt .top

3 1000 mov $1, %ax

3 1001 xchg %ax, mutex

3 1002 test $0, %ax

3 1003 jne .acquire

3 ------ Interrupt ------ ------ Interrupt ------

4 1006 mov %ax, count

4 1007 mov $0, mutex

4 1008 sub $1, %bx

4 1009 test $0, %bx

4 1010 jgt .top

4 ------ Interrupt ------ ------ Interrupt ------

4 1000 mov $1, %ax

4 1001 xchg %ax, mutex

4 1002 test $0, %ax

4 1003 jne .acquire

4 1004 mov count, %ax

4 ------ Interrupt ------ ------ Interrupt ------

4 1000 mov $1, %ax

4 1001 xchg %ax, mutex

4 1002 test $0, %ax

4 1003 jne .acquire

4 1000 mov $1, %ax

4 ------ Interrupt ------ ------ Interrupt ------

4 1005 add $1, %ax

5 1006 mov %ax, count

5 1007 mov $0, mutex

5 1008 sub $1, %bx

5 1009 test $0, %bx

5 ------ Interrupt ------ ------ Interrupt ------

5 1001 xchg %ax, mutex

5 1002 test $0, %ax

5 1003 jne .acquire

5 1004 mov count, %ax

5 1005 add $1, %ax

5 ------ Interrupt ------ ------ Interrupt ------

5 1010 jgt .top

5 1000 mov $1, %ax

5 1001 xchg %ax, mutex

5 1002 test $0, %ax

5 1003 jne .acquire

5 ------ Interrupt ------ ------ Interrupt ------

6 1006 mov %ax, count

6 1007 mov $0, mutex

6 1008 sub $1, %bx

6 1009 test $0, %bx

6 1010 jgt .top

6 ------ Interrupt ------ ------ Interrupt ------

6 1000 mov $1, %ax

6 1001 xchg %ax, mutex

6 1002 test $0, %ax

6 1003 jne .acquire

6 1004 mov count, %ax

6 ------ Interrupt ------ ------ Interrupt ------

6 1000 mov $1, %ax

6 1001 xchg %ax, mutex

6 1002 test $0, %ax

6 1003 jne .acquire

6 1000 mov $1, %ax

6 ------ Interrupt ------ ------ Interrupt ------

6 1005 add $1, %ax

7 1006 mov %ax, count

7 1007 mov $0, mutex

7 1008 sub $1, %bx

7 1009 test $0, %bx

7 ------ Interrupt ------ ------ Interrupt ------

7 1001 xchg %ax, mutex

7 1002 test $0, %ax

7 1003 jne .acquire

7 1004 mov count, %ax

7 1005 add $1, %ax

7 ------ Interrupt ------ ------ Interrupt ------

7 1010 jgt .top

7 1000 mov $1, %ax

7 1001 xchg %ax, mutex

7 1002 test $0, %ax

7 1003 jne .acquire

7 ------ Interrupt ------ ------ Interrupt ------

8 1006 mov %ax, count

8 1007 mov $0, mutex

8 1008 sub $1, %bx

8 1009 test $0, %bx

8 1010 jgt .top

8 ------ Interrupt ------ ------ Interrupt ------

8 1000 mov $1, %ax

8 1001 xchg %ax, mutex

8 1002 test $0, %ax

8 1003 jne .acquire

8 1004 mov count, %ax

8 ------ Interrupt ------ ------ Interrupt ------

8 1000 mov $1, %ax

8 1001 xchg %ax, mutex

8 1002 test $0, %ax

8 1003 jne .acquire

8 1000 mov $1, %ax

8 ------ Interrupt ------ ------ Interrupt ------

8 1005 add $1, %ax

9 1006 mov %ax, count

9 1007 mov $0, mutex

9 1008 sub $1, %bx

9 1009 test $0, %bx

9 ------ Interrupt ------ ------ Interrupt ------

9 1001 xchg %ax, mutex

9 1002 test $0, %ax

9 1003 jne .acquire

9 1004 mov count, %ax

9 1005 add $1, %ax

9 ------ Interrupt ------ ------ Interrupt ------

9 1010 jgt .top

9 1000 mov $1, %ax

9 1001 xchg %ax, mutex

9 1002 test $0, %ax

9 1003 jne .acquire

9 ------ Interrupt ------ ------ Interrupt ------

10 1006 mov %ax, count

10 1007 mov $0, mutex

10 1008 sub $1, %bx

10 1009 test $0, %bx

10 1010 jgt .top

10 ------ Interrupt ------ ------ Interrupt ------

10 1000 mov $1, %ax

10 1001 xchg %ax, mutex

10 1002 test $0, %ax

10 1003 jne .acquire

10 1004 mov count, %ax

10 ------ Interrupt ------ ------ Interrupt ------

10 1000 mov $1, %ax

10 1001 xchg %ax, mutex

10 1002 test $0, %ax

10 1003 jne .acquire

10 1000 mov $1, %ax

10 ------ Interrupt ------ ------ Interrupt ------

10 1005 add $1, %ax

11 1006 mov %ax, count

11 1007 mov $0, mutex

11 1008 sub $1, %bx

11 1009 test $0, %bx

11 ------ Interrupt ------ ------ Interrupt ------

11 1001 xchg %ax, mutex

11 1002 test $0, %ax

11 1003 jne .acquire

11 1004 mov count, %ax

11 1005 add $1, %ax

11 ------ Interrupt ------ ------ Interrupt ------

11 1010 jgt .top

11 1000 mov $1, %ax

11 1001 xchg %ax, mutex

11 1002 test $0, %ax

11 1003 jne .acquire

11 ------ Interrupt ------ ------ Interrupt ------

12 1006 mov %ax, count

12 1007 mov $0, mutex

12 1008 sub $1, %bx

12 1009 test $0, %bx

12 1010 jgt .top

12 ------ Interrupt ------ ------ Interrupt ------

12 1000 mov $1, %ax

12 1001 xchg %ax, mutex

12 1002 test $0, %ax

12 1003 jne .acquire

12 1004 mov count, %ax

12 ------ Interrupt ------ ------ Interrupt ------

12 1000 mov $1, %ax

12 1001 xchg %ax, mutex

12 1002 test $0, %ax

12 1003 jne .acquire

12 1000 mov $1, %ax

12 ------ Interrupt ------ ------ Interrupt ------

12 1005 add $1, %ax

13 1006 mov %ax, count

13 1007 mov $0, mutex

13 1008 sub $1, %bx

13 1009 test $0, %bx

13 ------ Interrupt ------ ------ Interrupt ------

13 1001 xchg %ax, mutex

13 1002 test $0, %ax

13 1003 jne .acquire

13 1004 mov count, %ax

13 1005 add $1, %ax

13 ------ Interrupt ------ ------ Interrupt ------

13 1010 jgt .top

13 1000 mov $1, %ax

13 1001 xchg %ax, mutex

13 1002 test $0, %ax

13 1003 jne .acquire

13 ------ Interrupt ------ ------ Interrupt ------

14 1006 mov %ax, count

14 1007 mov $0, mutex

14 1008 sub $1, %bx

14 1009 test $0, %bx

14 1010 jgt .top

14 ------ Interrupt ------ ------ Interrupt ------

14 1000 mov $1, %ax

14 1001 xchg %ax, mutex

14 1002 test $0, %ax

14 1003 jne .acquire

14 1004 mov count, %ax

14 ------ Interrupt ------ ------ Interrupt ------

14 1000 mov $1, %ax

14 1001 xchg %ax, mutex

14 1002 test $0, %ax

14 1003 jne .acquire

14 1000 mov $1, %ax

14 ------ Interrupt ------ ------ Interrupt ------

14 1005 add $1, %ax

15 1006 mov %ax, count

15 1007 mov $0, mutex

15 1008 sub $1, %bx

15 1009 test $0, %bx

15 ------ Interrupt ------ ------ Interrupt ------

15 1001 xchg %ax, mutex

15 1002 test $0, %ax

15 1003 jne .acquire

15 1004 mov count, %ax

15 1005 add $1, %ax

15 ------ Interrupt ------ ------ Interrupt ------

15 1010 jgt .top

15 1000 mov $1, %ax

15 1001 xchg %ax, mutex

15 1002 test $0, %ax

15 1003 jne .acquire

15 ------ Interrupt ------ ------ Interrupt ------

16 1006 mov %ax, count

16 1007 mov $0, mutex

16 1008 sub $1, %bx

16 1009 test $0, %bx

16 1010 jgt .top

16 ------ Interrupt ------ ------ Interrupt ------

16 1000 mov $1, %ax

16 1001 xchg %ax, mutex

16 1002 test $0, %ax

16 1003 jne .acquire

16 1004 mov count, %ax

16 ------ Interrupt ------ ------ Interrupt ------

16 1000 mov $1, %ax

16 1001 xchg %ax, mutex

16 1002 test $0, %ax

16 1003 jne .acquire

16 1000 mov $1, %ax

16 ------ Interrupt ------ ------ Interrupt ------

16 1005 add $1, %ax

17 1006 mov %ax, count

17 1007 mov $0, mutex

17 1008 sub $1, %bx

17 1009 test $0, %bx

17 ------ Interrupt ------ ------ Interrupt ------

17 1001 xchg %ax, mutex

17 1002 test $0, %ax

17 1003 jne .acquire

17 1004 mov count, %ax

17 1005 add $1, %ax

17 ------ Interrupt ------ ------ Interrupt ------

17 1010 jgt .top

17 1000 mov $1, %ax

17 1001 xchg %ax, mutex

17 1002 test $0, %ax

17 1003 jne .acquire

17 ------ Interrupt ------ ------ Interrupt ------

18 1006 mov %ax, count

18 1007 mov $0, mutex

18 1008 sub $1, %bx

18 1009 test $0, %bx

18 1010 jgt .top

18 ------ Interrupt ------ ------ Interrupt ------

18 1000 mov $1, %ax

18 1001 xchg %ax, mutex

18 1002 test $0, %ax

18 1003 jne .acquire

18 1004 mov count, %ax

18 ------ Interrupt ------ ------ Interrupt ------

18 1000 mov $1, %ax

18 1001 xchg %ax, mutex

18 1002 test $0, %ax

18 1003 jne .acquire

18 1000 mov $1, %ax

18 ------ Interrupt ------ ------ Interrupt ------

18 1005 add $1, %ax

19 1006 mov %ax, count

19 1007 mov $0, mutex

19 1008 sub $1, %bx

19 1009 test $0, %bx

19 ------ Interrupt ------ ------ Interrupt ------

19 1001 xchg %ax, mutex

19 1002 test $0, %ax

19 1003 jne .acquire

19 1004 mov count, %ax

19 1005 add $1, %ax

19 ------ Interrupt ------ ------ Interrupt ------

19 1010 jgt .top

19 1011 halt

19 ----- Halt;Switch ----- ----- Halt;Switch -----

20 1006 mov %ax, count

20 1007 mov $0, mutex

20 1008 sub $1, %bx

20 ------ Interrupt ------ ------ Interrupt ------

20 1009 test $0, %bx

20 1010 jgt .top

20 1011 halt

E:\ostep-homework-dev-zhongyl\threads-locks>x86.py -p yield.s -M count,mutex -a bx=10,bx=10 -i 5 -c

count mutex Thread 0 Thread 1

0 0

0 0 1000 mov $1, %ax

0 1 1001 xchg %ax, mutex

0 1 1002 test $0, %ax

0 1 1003 je .acquire\_done

0 1 1006 mov count, %ax

0 1 ------ Interrupt ------ ------ Interrupt ------

0 1 1000 mov $1, %ax

0 1 1001 xchg %ax, mutex

0 1 1002 test $0, %ax

0 1 1003 je .acquire\_done

0 1 1004 yield

0 1 ------ Interrupt ------ ------ Interrupt ------

0 1 1007 add $1, %ax

1 1 1008 mov %ax, count

1 0 1009 mov $0, mutex

1 0 1010 sub $1, %bx

1 0 1011 test $0, %bx

1 0 ------ Interrupt ------ ------ Interrupt ------

1 0 1005 j .acquire

1 0 1000 mov $1, %ax

1 1 1001 xchg %ax, mutex

1 1 1002 test $0, %ax

1 1 1003 je .acquire\_done

1 1 ------ Interrupt ------ ------ Interrupt ------

1 1 1012 jgt .top

1 1 1000 mov $1, %ax

1 1 1001 xchg %ax, mutex

1 1 1002 test $0, %ax

1 1 1003 je .acquire\_done

1 1 ------ Interrupt ------ ------ Interrupt ------

1 1 1006 mov count, %ax

1 1 1007 add $1, %ax

2 1 1008 mov %ax, count

2 0 1009 mov $0, mutex

2 0 1010 sub $1, %bx

2 0 ------ Interrupt ------ ------ Interrupt ------

2 0 1004 yield

2 0 ------ Interrupt ------ ------ Interrupt ------

2 0 1011 test $0, %bx

2 0 1012 jgt .top

2 0 1000 mov $1, %ax

2 1 1001 xchg %ax, mutex

2 1 1002 test $0, %ax

2 1 ------ Interrupt ------ ------ Interrupt ------

2 1 1005 j .acquire

2 1 1000 mov $1, %ax

2 1 1001 xchg %ax, mutex

2 1 1002 test $0, %ax

2 1 1003 je .acquire\_done

2 1 ------ Interrupt ------ ------ Interrupt ------

2 1 1003 je .acquire\_done

2 1 1006 mov count, %ax

2 1 1007 add $1, %ax

3 1 1008 mov %ax, count

3 0 1009 mov $0, mutex

3 0 ------ Interrupt ------ ------ Interrupt ------

3 0 1004 yield

3 0 ------ Interrupt ------ ------ Interrupt ------

3 0 1010 sub $1, %bx

3 0 1011 test $0, %bx

3 0 1012 jgt .top

3 0 1000 mov $1, %ax

3 1 1001 xchg %ax, mutex

3 1 ------ Interrupt ------ ------ Interrupt ------

3 1 1005 j .acquire

3 1 1000 mov $1, %ax

3 1 1001 xchg %ax, mutex

3 1 1002 test $0, %ax

3 1 1003 je .acquire\_done

3 1 ------ Interrupt ------ ------ Interrupt ------

3 1 1002 test $0, %ax

3 1 1003 je .acquire\_done

3 1 1006 mov count, %ax

3 1 1007 add $1, %ax

4 1 1008 mov %ax, count

4 1 ------ Interrupt ------ ------ Interrupt ------

4 1 1004 yield

4 1 ------ Interrupt ------ ------ Interrupt ------

4 0 1009 mov $0, mutex

4 0 1010 sub $1, %bx

4 0 1011 test $0, %bx

4 0 1012 jgt .top

4 0 1000 mov $1, %ax

4 0 ------ Interrupt ------ ------ Interrupt ------

4 0 1005 j .acquire

4 0 1000 mov $1, %ax

4 1 1001 xchg %ax, mutex

4 1 1002 test $0, %ax

4 1 1003 je .acquire\_done

4 1 ------ Interrupt ------ ------ Interrupt ------

4 1 1001 xchg %ax, mutex

4 1 1002 test $0, %ax

4 1 1003 je .acquire\_done

4 1 1004 yield

4 1 ------ Interrupt ------ ------ Interrupt ------

4 1 1006 mov count, %ax

4 1 1007 add $1, %ax

5 1 1008 mov %ax, count

5 0 1009 mov $0, mutex

5 0 1010 sub $1, %bx

5 0 ------ Interrupt ------ ------ Interrupt ------

5 0 1005 j .acquire

5 0 1000 mov $1, %ax

5 1 1001 xchg %ax, mutex

5 1 1002 test $0, %ax

5 1 1003 je .acquire\_done

5 1 ------ Interrupt ------ ------ Interrupt ------

5 1 1011 test $0, %bx

5 1 1012 jgt .top

5 1 1000 mov $1, %ax

5 1 1001 xchg %ax, mutex

5 1 1002 test $0, %ax

5 1 ------ Interrupt ------ ------ Interrupt ------

5 1 1006 mov count, %ax

5 1 1007 add $1, %ax

6 1 1008 mov %ax, count

6 0 1009 mov $0, mutex

6 0 1010 sub $1, %bx

6 0 ------ Interrupt ------ ------ Interrupt ------

6 0 1003 je .acquire\_done

6 0 1004 yield

6 0 ------ Interrupt ------ ------ Interrupt ------

6 0 1011 test $0, %bx

6 0 1012 jgt .top

6 0 1000 mov $1, %ax

6 1 1001 xchg %ax, mutex

6 1 1002 test $0, %ax

6 1 ------ Interrupt ------ ------ Interrupt ------

6 1 1005 j .acquire

6 1 1000 mov $1, %ax

6 1 1001 xchg %ax, mutex

6 1 1002 test $0, %ax

6 1 1003 je .acquire\_done

6 1 ------ Interrupt ------ ------ Interrupt ------

6 1 1003 je .acquire\_done

6 1 1006 mov count, %ax

6 1 1007 add $1, %ax

7 1 1008 mov %ax, count

7 0 1009 mov $0, mutex

7 0 ------ Interrupt ------ ------ Interrupt ------

7 0 1004 yield

7 0 ------ Interrupt ------ ------ Interrupt ------

7 0 1010 sub $1, %bx

7 0 1011 test $0, %bx

7 0 1012 jgt .top

7 0 1000 mov $1, %ax

7 1 1001 xchg %ax, mutex

7 1 ------ Interrupt ------ ------ Interrupt ------

7 1 1005 j .acquire

7 1 1000 mov $1, %ax

7 1 1001 xchg %ax, mutex

7 1 1002 test $0, %ax

7 1 1003 je .acquire\_done

7 1 ------ Interrupt ------ ------ Interrupt ------

7 1 1002 test $0, %ax

7 1 1003 je .acquire\_done

7 1 1006 mov count, %ax

7 1 1007 add $1, %ax

8 1 1008 mov %ax, count

8 1 ------ Interrupt ------ ------ Interrupt ------

8 1 1004 yield

8 1 ------ Interrupt ------ ------ Interrupt ------

8 0 1009 mov $0, mutex

8 0 1010 sub $1, %bx

8 0 1011 test $0, %bx

8 0 1012 jgt .top

8 0 1000 mov $1, %ax

8 0 ------ Interrupt ------ ------ Interrupt ------

8 0 1005 j .acquire

8 0 1000 mov $1, %ax

8 1 1001 xchg %ax, mutex

8 1 1002 test $0, %ax

8 1 1003 je .acquire\_done

8 1 ------ Interrupt ------ ------ Interrupt ------

8 1 1001 xchg %ax, mutex

8 1 1002 test $0, %ax

8 1 1003 je .acquire\_done

8 1 1004 yield

8 1 ------ Interrupt ------ ------ Interrupt ------

8 1 1006 mov count, %ax

8 1 1007 add $1, %ax

9 1 1008 mov %ax, count

9 0 1009 mov $0, mutex

9 0 1010 sub $1, %bx

9 0 ------ Interrupt ------ ------ Interrupt ------

9 0 1005 j .acquire

9 0 1000 mov $1, %ax

9 1 1001 xchg %ax, mutex

9 1 1002 test $0, %ax

9 1 1003 je .acquire\_done

9 1 ------ Interrupt ------ ------ Interrupt ------

9 1 1011 test $0, %bx

9 1 1012 jgt .top

9 1 1000 mov $1, %ax

9 1 1001 xchg %ax, mutex

9 1 1002 test $0, %ax

9 1 ------ Interrupt ------ ------ Interrupt ------

9 1 1006 mov count, %ax

9 1 1007 add $1, %ax

10 1 1008 mov %ax, count

10 0 1009 mov $0, mutex

10 0 1010 sub $1, %bx

10 0 ------ Interrupt ------ ------ Interrupt ------

10 0 1003 je .acquire\_done

10 0 1004 yield

10 0 ------ Interrupt ------ ------ Interrupt ------

10 0 1011 test $0, %bx

10 0 1012 jgt .top

10 0 1000 mov $1, %ax

10 1 1001 xchg %ax, mutex

10 1 1002 test $0, %ax

10 1 ------ Interrupt ------ ------ Interrupt ------

10 1 1005 j .acquire

10 1 1000 mov $1, %ax

10 1 1001 xchg %ax, mutex

10 1 1002 test $0, %ax

10 1 1003 je .acquire\_done

10 1 ------ Interrupt ------ ------ Interrupt ------

10 1 1003 je .acquire\_done

10 1 1006 mov count, %ax

10 1 1007 add $1, %ax

11 1 1008 mov %ax, count

11 0 1009 mov $0, mutex

11 0 ------ Interrupt ------ ------ Interrupt ------

11 0 1004 yield

11 0 ------ Interrupt ------ ------ Interrupt ------

11 0 1010 sub $1, %bx

11 0 1011 test $0, %bx

11 0 1012 jgt .top

11 0 1000 mov $1, %ax

11 1 1001 xchg %ax, mutex

11 1 ------ Interrupt ------ ------ Interrupt ------

11 1 1005 j .acquire

11 1 1000 mov $1, %ax

11 1 1001 xchg %ax, mutex

11 1 1002 test $0, %ax

11 1 1003 je .acquire\_done

11 1 ------ Interrupt ------ ------ Interrupt ------

11 1 1002 test $0, %ax

11 1 1003 je .acquire\_done

11 1 1006 mov count, %ax

11 1 1007 add $1, %ax

12 1 1008 mov %ax, count

12 1 ------ Interrupt ------ ------ Interrupt ------

12 1 1004 yield

12 1 ------ Interrupt ------ ------ Interrupt ------

12 0 1009 mov $0, mutex

12 0 1010 sub $1, %bx

12 0 1011 test $0, %bx

12 0 1012 jgt .top

12 0 1000 mov $1, %ax

12 0 ------ Interrupt ------ ------ Interrupt ------

12 0 1005 j .acquire

12 0 1000 mov $1, %ax

12 1 1001 xchg %ax, mutex

12 1 1002 test $0, %ax

12 1 1003 je .acquire\_done

12 1 ------ Interrupt ------ ------ Interrupt ------

12 1 1001 xchg %ax, mutex

12 1 1002 test $0, %ax

12 1 1003 je .acquire\_done

12 1 1004 yield

12 1 ------ Interrupt ------ ------ Interrupt ------

12 1 1006 mov count, %ax

12 1 1007 add $1, %ax

13 1 1008 mov %ax, count

13 0 1009 mov $0, mutex

13 0 1010 sub $1, %bx

13 0 ------ Interrupt ------ ------ Interrupt ------

13 0 1005 j .acquire

13 0 1000 mov $1, %ax

13 1 1001 xchg %ax, mutex

13 1 1002 test $0, %ax

13 1 1003 je .acquire\_done

13 1 ------ Interrupt ------ ------ Interrupt ------

13 1 1011 test $0, %bx

13 1 1012 jgt .top

13 1 1000 mov $1, %ax

13 1 1001 xchg %ax, mutex

13 1 1002 test $0, %ax

13 1 ------ Interrupt ------ ------ Interrupt ------

13 1 1006 mov count, %ax

13 1 1007 add $1, %ax

14 1 1008 mov %ax, count

14 0 1009 mov $0, mutex

14 0 1010 sub $1, %bx

14 0 ------ Interrupt ------ ------ Interrupt ------

14 0 1003 je .acquire\_done

14 0 1004 yield

14 0 ------ Interrupt ------ ------ Interrupt ------

14 0 1011 test $0, %bx

14 0 1012 jgt .top

14 0 1013 halt

14 0 ----- Halt;Switch ----- ----- Halt;Switch -----

14 0 1005 j .acquire

14 0 1000 mov $1, %ax

14 0 ------ Interrupt ------ ------ Interrupt ------

14 1 1001 xchg %ax, mutex

14 1 1002 test $0, %ax

14 1 1003 je .acquire\_done

14 1 1006 mov count, %ax

14 1 1007 add $1, %ax

14 1 ------ Interrupt ------ ------ Interrupt ------

15 1 1008 mov %ax, count

15 0 1009 mov $0, mutex

15 0 1010 sub $1, %bx

15 0 1011 test $0, %bx

15 0 1012 jgt .top

15 0 ------ Interrupt ------ ------ Interrupt ------

15 0 1000 mov $1, %ax

15 1 1001 xchg %ax, mutex

15 1 1002 test $0, %ax

15 1 1003 je .acquire\_done

15 1 1006 mov count, %ax

15 1 ------ Interrupt ------ ------ Interrupt ------

15 1 1007 add $1, %ax

16 1 1008 mov %ax, count

16 0 1009 mov $0, mutex

16 0 1010 sub $1, %bx

16 0 1011 test $0, %bx

16 0 ------ Interrupt ------ ------ Interrupt ------

16 0 1012 jgt .top

16 0 1000 mov $1, %ax

16 1 1001 xchg %ax, mutex

16 1 1002 test $0, %ax

16 1 1003 je .acquire\_done

16 1 ------ Interrupt ------ ------ Interrupt ------

16 1 1006 mov count, %ax

16 1 1007 add $1, %ax

17 1 1008 mov %ax, count

17 0 1009 mov $0, mutex

17 0 1010 sub $1, %bx

17 0 ------ Interrupt ------ ------ Interrupt ------

17 0 1011 test $0, %bx

17 0 1012 jgt .top

17 0 1000 mov $1, %ax

17 1 1001 xchg %ax, mutex

17 1 1002 test $0, %ax

17 1 ------ Interrupt ------ ------ Interrupt ------

17 1 1003 je .acquire\_done

17 1 1006 mov count, %ax

17 1 1007 add $1, %ax

18 1 1008 mov %ax, count

18 0 1009 mov $0, mutex

18 0 ------ Interrupt ------ ------ Interrupt ------

18 0 1010 sub $1, %bx

18 0 1011 test $0, %bx

18 0 1012 jgt .top

18 0 1000 mov $1, %ax

18 1 1001 xchg %ax, mutex

18 1 ------ Interrupt ------ ------ Interrupt ------

18 1 1002 test $0, %ax

18 1 1003 je .acquire\_done

18 1 1006 mov count, %ax

18 1 1007 add $1, %ax

19 1 1008 mov %ax, count

19 1 ------ Interrupt ------ ------ Interrupt ------

19 0 1009 mov $0, mutex

19 0 1010 sub $1, %bx

19 0 1011 test $0, %bx

19 0 1012 jgt .top

19 0 1000 mov $1, %ax

19 0 ------ Interrupt ------ ------ Interrupt ------

19 1 1001 xchg %ax, mutex

19 1 1002 test $0, %ax

19 1 1003 je .acquire\_done

19 1 1006 mov count, %ax

19 1 1007 add $1, %ax

19 1 ------ Interrupt ------ ------ Interrupt ------

20 1 1008 mov %ax, count

20 0 1009 mov $0, mutex

20 0 1010 sub $1, %bx

20 0 1011 test $0, %bx

20 0 1012 jgt .top

20 0 ------ Interrupt ------ ------ Interrupt ------

20 0 1013 halt

**15、相比test-and-set.s，test-and-test-and-set.s将判断锁是否已经被占用从xchg后挪到了一开始，即先判断锁是否已上，再进行原子交换，然后再次判断，若第二次结果仍然是未上锁，则继续运行关键内容。**

**与原来版本相比，test-and-test-and-set.s的优势在于自旋等待锁时仅需要运行mov mutex, %ax ；test $0, %ax；jne .acquire三步即可，省去了test-and-set.s中反复需要进行的原子交换，减少了对硬件的访问，同时安全性仍然有保障。**

**弊端在于若中断间隔过短则反而可能拖慢速度。（i=1，2，3这种极端情况）**

**（下为test-and-test-and-set.s的代码及注释）**

.main

.top

.acquire

mov mutex, %ax

test $0, %ax

jne .acquire

mov $1, %ax

xchg %ax, mutex # 对1与mutex进行原子交换

test $0, %ax # 若返回值为0，则锁未被占用

jne .acquire # 否则锁被占用，循环等待

# critical section

mov count, %ax # 取count的值

add $1, %ax # 加一

mov %ax, count # 存回count中

# release lock

mov $0, mutex

# see if we're still looping

sub $1, %bx

test $0, %bx

jgt .top