```
.var flag 2  => array of 2integers

.var turn ->

.var count ->

.main

                                    reg
lea flag, %fx  =>  flag -> fx
                   addr

mov %bx, %cx  =>  cx = bx -> self

neg %cx  =>  cx = - self

add $1, %cx   cx = 1+(-self)

.acquire
mov $1, 0(%fx, %bx, 4),  -> flag[self] = 1
        ↓
   disp -> 0
   base -> %fx   mem addr = %fx + (%bx x 4)
   index -> %bx
   scale -> 4
 mov %cx, turn -> turn = 1 - self
  .spin1
  mov 0(%fx, %cx, 4), %ax -> ax = flag[1-self]
   test $1, %ax
   jne .fini   , if flag[1-self] != 1, skip past loop
                                         to .fini
        meaning go out of spin, exec critical section
```

```
.spin2
 mov turn, %ax
  test %cx, %ax   → compare turn and 1-self
  je  .spin1 → if turn == 1-self , back to spinning

.fini
 mov count, %ax     ax = count   }
add $1, %ax          ax = 1+ ax   }  critical section
mov %ax, count      count = ax   }

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

mov %cx , turn     turn = 1-self
 halt
```

Let's assume thread 0 acquires lock 1st,

meaning self = 0

Dry run (thread 0)

cx = 1 , bx = 0 , flag[0] = 1 , turn = 1

ax = flag[1] , 0 != 1 jne .fini

ax = 1 (count)

flag[0] = 0

turn = 1

Dry run (thread 1)

bx = 1, cx = 0

flag[1] = 1, turn = 0

ax = flag[0]

1 == flag[0] → this is 1. since thread 0 is running with lock acquired

Now thread 1 was spinning when thread 0 releases the lock

Above now goes to critical section because the condition, $1 != \overset{\text{flag[0], lock released}}{0}$

Or it was executing spin 2 section, now $0 != \overset{\text{turn, updated by thread 0}}{1}$