Ticket Spinlock [ticket] - 3 балла

Ticket spinlock с помощью атомарного счетчика выстраивает потоки в очередь в порядке полученных ими номерков, т.е. ведет себя как простой планировщик. Такая логика работы может конфликтовать с планировщиком операционной системы, у которого могут быть свои взгляды на порядок исполнения потоков.

Рассмотрим простой сценарий: в системе исполняются n потоков, каждый из них в бесконечном цикле выполняет короткую критическую секцию. Под короткой критической секцией будем понимать такой фрагмент кода, который можно много раз исполнить в течение одного кванта времени.

Утверждается, что с ростом числа потоков в какой-то момент частота исполнения критических секций (т.е. число критических секций в секунду) в системе очень сильно упадет.

1) При каком числе потоков это случится и почему?

2) Почему test-and-set spinlock не подвержен этой проблеме?

Будем предполагать следующий алгоритм поведения планировщика: каждому потоку выдается фиксированный квант времени на исполнение на процессоре, после чего выполнение потока прекращается, планировщик помещает его в хвост очереди на исполнение, а на освободившемся процессоре запускает поток из головы очереди.

```
void ticket_spinlock::lock() {
  const ticket_t this_thread_ticket = next_free_ticket_.fetch_add(1);
  while (this_thread_ticket != owner_ticket_.load()) {
      // wait
  }
}

void tas_spinlock::lock() {
  while (locked_.exchange(true)) {
      // wait
  }
}
```

Подсказка: подумайте, что будет, если поток встанет в очередь на ticket спинлоке, но будет вытеснен планировщиком.