Теория и практика многопоточного программирования

Семинар 5

Неганов Алексей

Московский физико-технический институт (национальный исследовательский университет)
Кафедра теоретической и прикладной информатики

Москва 2020

Spinlock

```
class spin_lock_TAS
{
    atomic<unsigned int> m_spin ;
public:
    spin_lock_TAS(): m_spin(0) {}
    ~spin_lock_TAS() { assert( m_spin.load() == 0);}

    void lock() {
        unsigned int expected;
        do { expected = 0; }
        while ( !m_spin.compare_exchange_weak(expected, 1));
    }

    void unlock() {
        m_spin.store(0);
    }
};
```

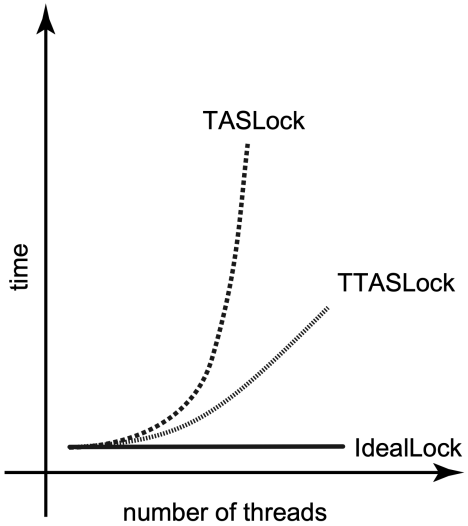
Вопрос

Как в атомарных операциях расставить `memory_order`?

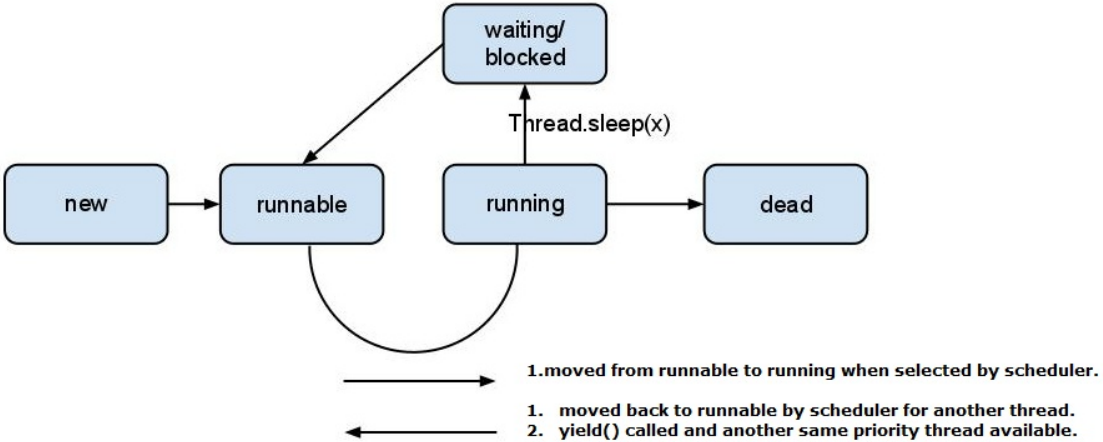
Spinlock

```
class spin_lock_TTAS
{
    atomic<unsigned int> m_spin ;
public:
    spin_lock_TTAS(): m_spin(0) {}
    ~spin_lock_TTAS() { assert( m_spin.load() == 0);}

    void lock() {
        unsigned int expected;
        do {
            while (m_spin.load());
            expected = 0;
        }
        while ( !m_spin.compare_exchange_weak(expected, 1));
    }

    void unlock() {
        m_spin.store(0);
    }
};
```

Spinlock

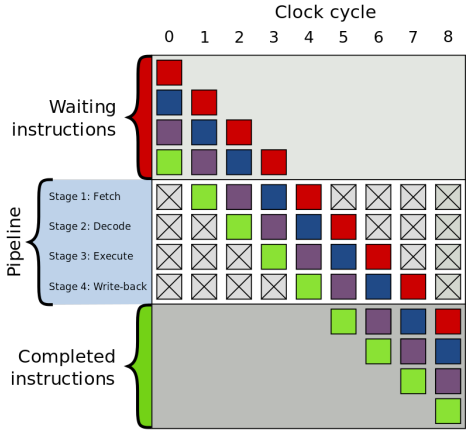


Жизненный цикл потока: yield vs sleep



Как нам обустроить spin loop

```
while(flag.load() == 0) {
  __asm volatile ("pause" ::: "memory");
}
```

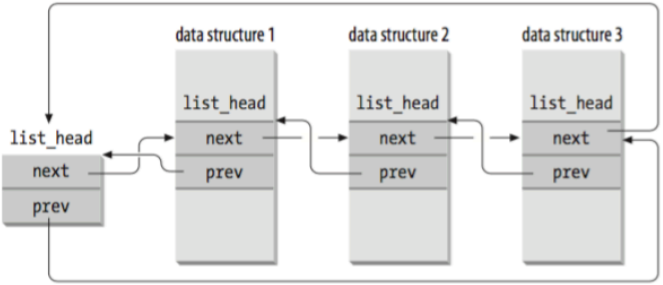


Ticket lock

```
class ticket_lock
{
    std::atomic_size_t now_serving = {0};
    std::atomic_size_t next_ticket = {0};

public:
    void lock() {
        const auto ticket = next_ticket.fetch_add(1);
        while (now_serving.load() != ticket);
    }

    void unlock() {
        const auto successor = now_serving.load() + 1;
        now_serving.store(successor);
    }
};
```

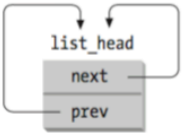
Вопрос

Как в атомарных операциях расставить `memory_order`?



(a) a doubly linked listed with three elements

(b) an empty doubly linked list



Offtop: контейнеры на Си

```
#include <stddef.h>
// #define offsetof(st, m) \
//     ((size_t)((char *)&((st *)0)->m - (char *)0))

#if defined(__GNUC__) || defined(__clang__)

#define container_of(ptr, type, member)                                 \
  ({                                                                    \
    const typeof(((type *)0)->member) *__mptr = (ptr);                  \
    (type *)((char *)__mptr - offsetof(type, member));                  \
  })

#elif defined(_MSC_VER)

#define container_of(ptr, type, member) (type*)((char*)ptr - offsetof(type, member))

#endif
```

Задачи

1. Подумайте, как нужно поставить `memory_order` в обращениях к атомарным переменным в примерах с семинара.
2. **(Обязательная)** Напишите свои mutex'ы, использующие yield / exponential backoff. Сравните производительность TAS lock / TTAS lock / ticket lock. Предлагается использовать C++11 (и выше), при желании можно GNU C11 и pthreads.