Теория и практика многопоточного программирования

Семинар 3

Неганов Алексей

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

Москва 2020

Неатомарные операции

- Инструкция процессора, работающая с невыровненным адресом

```
struct  __attribute__((__packed__)) my_struct {
   char padding[2];
   int32_t cnt;
};
```

- Инструкция процессора, работающая с выровненным адресом

```
strd r0, r1, [r2] ; ARMv7
```

- Любая операция языка C

Атомарные регистры

Теорема

Если поведение соисполняемого регистра удовлетворяет условиям:

1. $\nexists i\, R^i \to W^i$ (ни одно чтение не может вернуть значение из будущего),
2. $\nexists j\, W^i \to W^j \to R^i$ (ни одно чтение не может вернуть значение из отдалённого прошлого),
3. если $R^i \to R^j$, то $i \leqslant j$,

то такая реализация является атомарной.

Алгоритм «булочной» Лампорта

```
bool entering[n];
int num[n] = {0};

void lock(int i) {
   // choose number
   entering[i] = true;
   num[i] = max(num[0], ..., num[n-1]) + 1;
   entering[i] = false;

   for (j = 0; j < n; j++) {
      // wait until thread j receives its number
      while(entering[j]);

      // wait for threads with smaller numbers or same number and smaller index
      while(num[j] > 0 && (num[j] < num[i] || (num[j] == num[i] && j < i)));
   }
}

void unlock(int i) {
   num[i] = 0;
}
```

RMW-операции

CAS:

```
bool bool_compare_and_swap(type *ptr, type oldval, type newval) {
   if (*ptr == oldval) {
      *ptr = newval;
      return true;
   }
   else
      return false;
}

type val_compare_and_swap(type *ptr, type oldval, type newval) {
   if (*ptr == oldval) {
      *ptr = newval;
      return newval;
   }
   else
      return *ptr;
}
```

RMW-операции

Реализация прочих примитивов:

```
type fetch_and_add(type *ptr, type value) {
   int cur;
   do {
      cur = *ptr;
   } while (!bool_compare_and_swap(ptr, cur, cur + value));
   return cur;
}
```

Типичный паттерн использования:

```
while (1) {
   oldval = *ptr;
   newval = /* ... */
   if (bool_compare_and_swap( ptr, oldval, newval))
      break;
}
```

RMW-операции: ABA

```
1  struct node {
2        struct node *next;
3  }
4
5  static struct node *top = NULL;
6
7  void push(struct node *n) {
8        do {
9              struct node *t = top;
10             n->next = t;
11       while (!CAS(&top, t, n));
12 }
13
14 void struct node *pop(void) {
15       struct node *next;
16       do {
17             struct node *t = top;
18             if (t == NULL)
19                   return NULL;
20             next = t->next;
21       } while (!CAS(&top,t,next));
22       return t;
23 }
```

Пусть имеются два потока, выполняющий каждый следующие действия:

```
    #1:                          #2:

struct node *a = pop();          struct node *b = pop();
                                 pop();
                                 push(b);
```

Может ли этот код привести к некорректному состоянию стека? Если да, на каких строках должно произойти переключение контекста?

Задачи

1. Напишите реализацию односвязного списка, поддерживающего чтение и вставку **(пока без удаления)**, безопасного в многопоточной среде, используя примитив CAS. Сравните реализации на C++11 и GNU C.