Design of Parallel and High Performance Computing
HS 2018
Markus Püschel, Torsten Hoefler
Department of Computer Science
ETH Zurich

Homework 3

Sequential Consistency

Exercise 1

For the following executions, please indicate if they are sequentially consistent. All variables are initially set to 0.

1. P1: W(x,1);
   P2: R(x,0); R(x,1)
2. P1: W(x,1);
   P2: R(x,1); R(x,0);
3. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,1); R(x,2);
4. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,2); R(x,1);
5. P1: W(x,1);
   P2: W(x,2);
   P3: R(x,2); R(x,1);
   P4: R(x,1); R(x,2);
6. P1: W(x,1); R(x,1); R(y,0);
   P2: W(y,1); R(y,1); R(x,1);
   P3: R(x,1); R(y,0);
   P4: R(y,0); R(x,0);
7. P1: W(x,1); R(x,1); R(y,0);
   P2: W(y,1); R(y,1); R(x,1);
   P3: R(y,1); R(x,0);

The x86 Memory Model: TLO+CC

1. Describe the memory model of the x86 architecture. Where does it differ from sequential consistency.
2. Two threads execute the following code (given in AT&T assembly syntax) on a machine using TLO+CC. Is it possible that the registers eax $= 0 \wedge$ ebx $= 0$ after both threads have finished?. Would it be possible in sequential consistency?

Initial: All registers and memory locations are 0	
Thread A	Thread B
movl $1, 0x42 movl 0x50, %eax	movl $1, 0x50 movl 0x42, %ebx

A new lock?

Does the following code ensure that at any time, at most one thread is in the critical region if we assume sequential consistency? Use sequential consistency to prove your answer.

The variables me and $other$ are thread-local, the variables $want[0]$, $want[1]$ and $turn$ are shared. The variables $want$ and $turn$ are initially zero, me contains the thread id($\in 0, 1$) for each thread, while $other$ contains the value $1 - me$.

```
want[me] = 1;
while (turn != me) {
    while (want[other]) {}
    turn = me;
}
// CR
want[me] = 0;
```