Assignment 5, Operating Systems

Jay R Bolton

November 8, 2011

Ch 5 problems: 5.1, 5.3, 5.4, 5.5, 5.12, 5.19, 5.21, 5.25

5.1 Multiprogramming is simpler because you do not need to save processor state.

5.3 a.
```
P2 is two lines behind P1
P1: shared int x;
P1: x = 10;
P2: shared int x;
P1: while(1)
P2: x = 10;
P1: x = 9;
P2: while(1)
P1: x = 10;
P2: x = 9;
P1: if ...
P2: x = 10;
P1: print...

...
P2: shared int x
P1: load x r1
P2: x = 10
P1: decrement r1
P2: while(1)

P1: load x r1
P2: load x r2
P1: increment r1
P2: decrement r2
P1: store r1 (x = 9)
P2: store r2 (x = 8)
...
```

5.4 2 ... 100

Explanation of lower bound:

- P1 does one increment but does not store.
- P2 increments/stores 49 times
- P1 resumes and stores tally as 1
- P2 resumes loads 1
- P1 resumes and loads 1, then increments/stores 49 times
- P2 resumes with 1 already loaded and increments/stores once

5.5 It seems like it, since busy waiting has to be scheduled by the OS as if it was a real process, wasting processor time. It does eliminate process context saving though.

5.12 Yes, the if statement in semSignal serves the purpose of a negative semaphore count.

5.19 Have the producer run until it signals s. Then have the consumer run until it hits the moved conditional. The consumer will wait on delay while the producer will wait on s.

5.21 They would all affect the program.

5.25 If we do one read and one write, we'll get deadlocked on wrt.