Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 2:00-3:50pm @ FAB 40-07; Office Hr: MW 1-2pm & by appt.

Labs 3 and 4 Solutions

Lab 3. Memory Consistency Models:

x	y	z	(a)	(b)
0	0	0	yes	no
0	0	1	yes	yes
0	1	0	yes	no
0	1	1	yes	yes

x	y	z	(a)	(b)
1	0	0	yes	no
1	0	1	yes	yes
1	1	0	no	no
1	1	1	yes	yes

Lab 3. OpenMP Exercises:

The first `printf` may print either 2 or 5; the second `printf` always prints 5.

Lab 4. Problem 1 (Shared-memory programming):

(a) The threads read and write to the same shared variable `sum`. These operations may interleave and cause incorrect result. For instance, the following scenario would give a wrong result:

```
(assume sum has value x)
T1 reads sum and gets x
T2 reads sum and gets x
T1 writes a new value y to sum
T2 writes a new value z to sum (which overrides y)
```

(b) Any single value of `a[i]` as well as any partial sum of `a[i], 0<=i<n` can be a valid return value. For this particular case, the possible return values are 0, 1, 2, 3, 4, 5, 6.

Lab 4. Problem 2 (Pthreads):

(a)

```
son 1
son 2
grandson
grandson
Work done
```

The first four lines could be in any order.

(b) The purpose of the `pthread_join` calls is to synchronize the spawned threads back to their parent. If they are omitted, parent may terminate before children. For the above example, we may not see all four output lines from the child threads.

Lab 4. Problem 3 (OpenMP):

(a) A for loop is expected after the OMP `for` directive.

(b) The OMP `parallel` directive is missing.

Lab 4. Problem 4 (Synchronization):

(a) This routine can only work correctly once. It will break if the group of threads call `Barrier()` a second time — Since the variable `count` never gets reset, there will be no waiting at all.

(b) The last-arriving thread to the barrier synchronization may decide to start a second round of barrier synchronization immediately. If this thread beats some waiting threads to get the lock and reset the `count`, those waiting threads will miss the `count==0` window and never be able to continue.

Lab 4. Problem 5 (Fortran 90/95):

(a) `A = (0,2,2,2,4)`

(b) `A(2:4) = B(3:5) + A(1:3)`

(c) No. According to the semantics of `forall`, the parallel loop will use the *old* value of `A(i-1)`. However, due to the iterations' execution order, the sequential loop will use the *new* value of `A(i-1)`.

(d) Yes. In this case, both the sequential loop and the parallel loop use the *old* value of `A(i-1)`.