Programming Languages

Dr. Michael Petter, Raphaela Palenta WS 2017/18

Exercise Sheet 5

Assignment 5.1 Restricted Transactional Memory

Consider the following code fragment on a machine with RTM and Caches:

```
int data = 0;
int s=0;
```

thread P_0:

```
while (s!=-1)
  if((s=_xbegin())==-1){
    data++;
    _xend();
  }
```

thread P_1:

```
if (_xbegin()==-1){
  data++;
  _xend();
}else {
  data++;
}
```

1. Fill in the gap with either "**will**", "**will not**" or "**may or may not**":

   After P_0 and P_1 both terminate, `data` ____________________ evaluate to 1.

2. Fill in the gap with either "**will**", "**will not**" or "**may or may not**":

   After P_0 and P_1 both terminate, `data` ____________________ evaluate to 3.

3. Consider the following interleaving of paths through the program:

```
P0 s!=-1 (s=_xbegin())==-1                data++;          s!=-1  (s=_xbegin())==-1   data++  _xend()
P1                               _xbegin()==-1        data++;
---------------------------------------------------------------------------------------------------> time
```

   Draw a happened-before diagram of this interleaving. The initial cache states for `s` and `data` are $S0, S0$. (No store buffer and invalidate queue.)

4. Fix the program, such that 2 is the only value, that `data` may evaluate to after termination of both threads. (Of course without hardcoding!)

Assignment 5.2 STM vs. RTM

This time, we want to compare Tranactional Memory implementations with each other as well as the old implementations from tutorial sheet 3. Thus, we will equip the bumper allocator with TM implementations.
Equip the bumper allocation implementation with

- explicit RTM (this will only run on a CPU with transactional memory)
- GCC transaction extensions