Assignment 5.1 Restricted Transactional Memory

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

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

thread P0:

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

thread P1:

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

1. Fill in the gap with either “will”, “will not” or “may or may not”:
   After P0 and P1 both terminate, data evaluates to 1.

2. Fill in the gap with either “will”, “will not” or “may or may not”:
   After P0 and P1 both terminate, data evaluates to 3.

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

   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 Transactional 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