```
MemManager::register_thread(int num) { self = new ThreadContext(num, this); }
MemManager::unregister_thread() { /* no-op */ }
MemManager::op_begin() { /* no-op */ }
void MemManager::sched_for_reclaim(void* ptr) { self->pending_reclaims.push_back(ptr); }
bool MemManager::try_reserve(void* ptr) {
  for (int i = 0; i < num; ++i) {
    if (self->reservations[i] == nullptr) {
      self->reservations[i].store(ptr);
      return true;
    }
  }
  throw error;
}
void MemManager::unreserve(void* ptr) {
  for (int i = 0; i < num; ++i) {
    if (self->reservations[i] == ptr) {
      self->reservations[i].store(nullptr);
      return;
    }
  }
}
void MemManager::op_end() {
  for (int i = 0; i < self->num; ++i)
    self->reservations[i].store(nullptr);
  for (auto i : pending_reclaims) {
    wait_until_unreserved(p);
    free(p);
  }
  pending_reclaims.clear();
}
MemManager::wait_until_unreserved(void* ptr) {
  ThreadContext* curr = head;
  while (curr) {
    for (int i = 0; i < curr->num; ++i) {
      while (curr->reservations[i] == ptr)
        wait();
    }
    curr = curr->next;
  }
}
```

FIGURE 19.7 MemManager methods to support hazard pointers with blocking reclamation.