Shared Memory Consistency Models

Some sample code

```cpp
Initially all pointers = null, all integers = 0.

P1, P2, P3, ..., Pn

while (there are more tasks) {
    Task = GetFromFreeList();
    Task = Data ...;
    insert Task in task queue
}
Head = head of task queue;

while (MyTask == null) {
    Begin Critical Section
    if (Head != null) {
        Myrank = Head;
        Head = Head -> Next;
    }
    End Critical Section
}
...
MyTask = Data;
```

Figure 1: What value can a read return?

- What assumptions is the programmer making?

Memory Consistency model impacts

- Cache/memory hierarchy design
- Cache design, write buffers, etc.
- CPU execution core (eg, ooo instruction scheduler)
- Compiler
- Programmer

Uniprocessor Consistency

- What are the typical guarantees for uniprocessor consistency?
- How does this effect processor design/optimization?
Sequential Consistency

- **Definition**: A multiprocessor system is **sequentially consistent** if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.

Sequential consistency implies:

- (1) maintaining program order among operations from individual processors, and
- (2) maintaining a single sequential order among operations from all processors.

---

Sequential Consistency and the Write Buffer

- **Write Buffer**

- Overlapping writes

---

Overlapping writes
Non-blocking Reads

Sequential Consistency

• Additional changes because of caches?

Sequential Consistency

• What does it mean for a write to “complete” in a multiprocessor?

• Why is sequential consistency tougher in a write-update coherence protocol than write-invalidate?

Sequential Consistency and Compilers

• What are the issues?

• How to avoid?
The cost of Sequential consistency

• What performance optimizations are not allowed?

• Performance optimizations to minimize the cost
  – Exclusive prefetch
  – Speculative loads with rollback

Relaxed Consistency Models

• Relax processor ordering constraints
• Relax write atomicity reqt

Relaxing

• Write-to-read program order.
  – What does this mean?
  – What does this buy you?
  – What do they mean by “safety nets”?

• Relaxing write-to-write program order
  – What does this mean?
  – What does this buy you?
Release Consistency

operations. For RCxx, the constraints are as follows:

- acquire — all — release, and special — special.

For RCpx, the write to read program order among special operations is eliminated:

- acquire — all, all — release, and special — special except for a special write followed by a special read.

Compiler, again

- What does it imply for the compiler if the processor supports relaxed consistency?