UNIVERSITÉ DU LUXEMBOURG

Memory Consistency

Parallel Computing

Goals

- ✯ Study memory memory consistencies and litmus tests.
- ✯ Think in groups of 2 or 3 students.

1 Litmus Tests

We have seen 5 litmus tests that are summarized here below.

1.1 Message Passing (MP)

Can this program see `r1 = 1, r2 = 0`?

```
// Thread 1              // Thread 2
x = 1                    r1 = y
y = 1                    r2 = x
```

1.2 Store Buffering (SB)

Can this program see `r1 = 0, r2 = 0`?

```
// Thread 1              // Thread 2
x = 1                    y = 1
r1 = y                   r2 = x
```

1.3 Load Buffering (LB)

Can this program see `r1 = 1, r2 = 1`?

```
// Thread 1     // Thread 2
r1 = x          r2 = y
y = 1           x = 1
```

1.4 Independent Read of Independent Write (IRIW)

Can this program see `r1 = 1, r2 = 0, r3 = 1, r4 = 0`?

```
// Thread 1     // Thread 2     // Thread 3     // Thread 4
x = 1           y = 1           r1 = x          r3 = y
                                r2 = y          r4 = x
```

1.5 Coherence

Can this program see `r1 = 1, r2 = 2, r3 = 2, r4 = 1`?

```
// Thread 1     // Thread 2     // Thread 3     // Thread 4
x = 1           x = 2           r1 = x          r3 = x
                                r2 = x          r4 = x
```

Exercise 1 – Abstraction of Hardware Architectures (recall)
Describe the x86 total store order (TSO) architecture and the ArmV7/Power architecture. Help yourself with a diagram.

Exercise 2 – Litmus Tests
Fill in the following table (put an "X" if the litmus tests fail).

	MP	SB	LB	IRIW	Coherence
SC hardware					
x86-TSO					
ArmV7/relaxed mem.					
Any lang. with ordinary variables					
C++ with SC atomics					

Exercise 3 – Other litmus test (S)
Consider the following litmus test:

```
// Thread 1           // Thread 2
x = 2                 r1 = y
y = 1                 x = 1
```

Check on which memory consistency models we can observe $x = 2 \wedge r1 = 1$.

Exercise 4 – New Litmus Test

Consider the following variant of IRIW (only changing the order of read for thread 4):

> Can this program see `r1 = 1, r2 = 0, r3 = 1, r4 = 0`?
> (Can Threads 3 and 4 see x and y change in different orders?)

```
// Thread 1     // Thread 2     // Thread 3     // Thread 4
x = 1           y = 1           r1 = x          r3 = x
                                r2 = y          r4 = y
```

Can this litmus test still distinguishes between x86-TSO and relaxed consistency (ARM)? Justify your answer.

Exercise 5 – New Architecture

Modify the x86-TSO architecture to have a read buffers instead of a write buffers. Analyze this read buffers architecture on the litmus tests.

Exercise 6 – SB

Do we need to add two `MFENCE` instructions on x86-TSO hardware to fix the store buffering litmus test? Would the following code fix it?

```
// Thread 1             // Thread 2
x = 1                   y = 1
MFENCE
r1 = y                  r2 = x
```

Justify your answer.

Exercise 7 – Compare and swap

The compare-and-swap operation for minimum can be implemented as follows:

```
void atomic_min(std::atomic<int>& a, int b) {
  int old = a.load();
  while (old > b && !a.compare_exchange_weak(old, b)) {
    // Note that old will be reload if the compare_exchange could not work.
  }
}
```

Implement a general `template <class F> void atomic_op(std::atomic<int>& a, int b, F f).`