Examples

Pieter P

Acquire-release

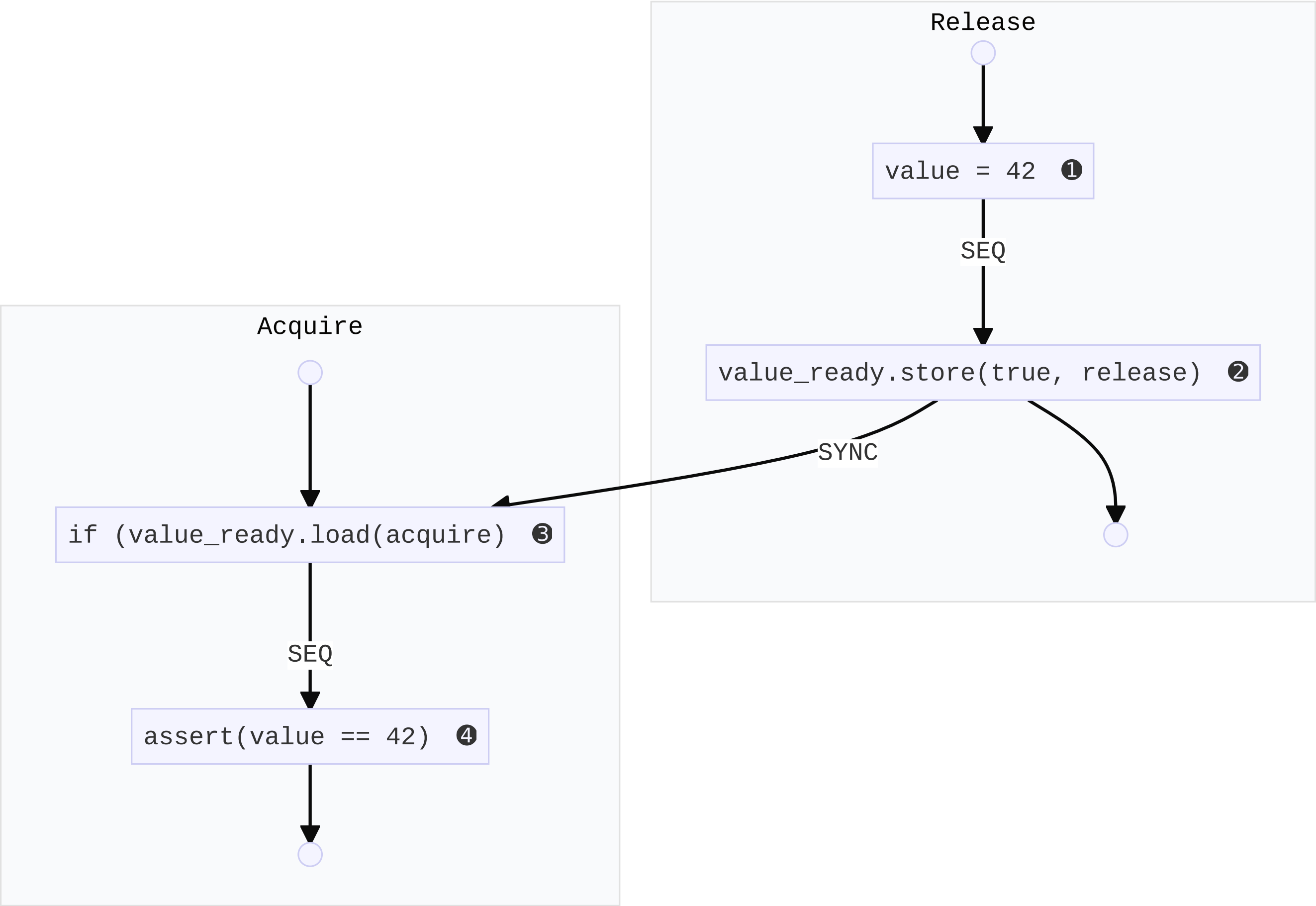
```cpp
unsigned value = 0;
std::atomic_bool value_ready{false};
```

Release

```cpp
value = 42; //                                                             ❶
value_ready.store(true, std::memory_order_release); //                     ❷ ▲▲▲▲
```

Acquire

```cpp
if (value_ready.load(std::memory_order_acquire)) //                        ❸ ▼▼▼▼
    assert(value == 42); //                                                ❹
```


Release
value = 42 ❶
SEQ
value_ready.store(true, release) ❷
SYNC
Acquire
if (value_ready.load(acquire) ❸
SEQ
assert(value == 42) ❹

If ❸ reads the value written by ❷, then the load-acquire ❸ synchronizes with the store-release ❷ on the same atomic variable. Therefore, ❷ simply happens before ❸. Combined with the fact that ❶ is sequenced before ❷ and ❸ is sequenced before ❹, we conclude that ❶ strongly happens before ❹.

If ❸ does not read the value written by ❷, then there is no synchronization, but this is not an issue because in that case, the branch is not taken and ❹ is never executed.

Interrupt handle once

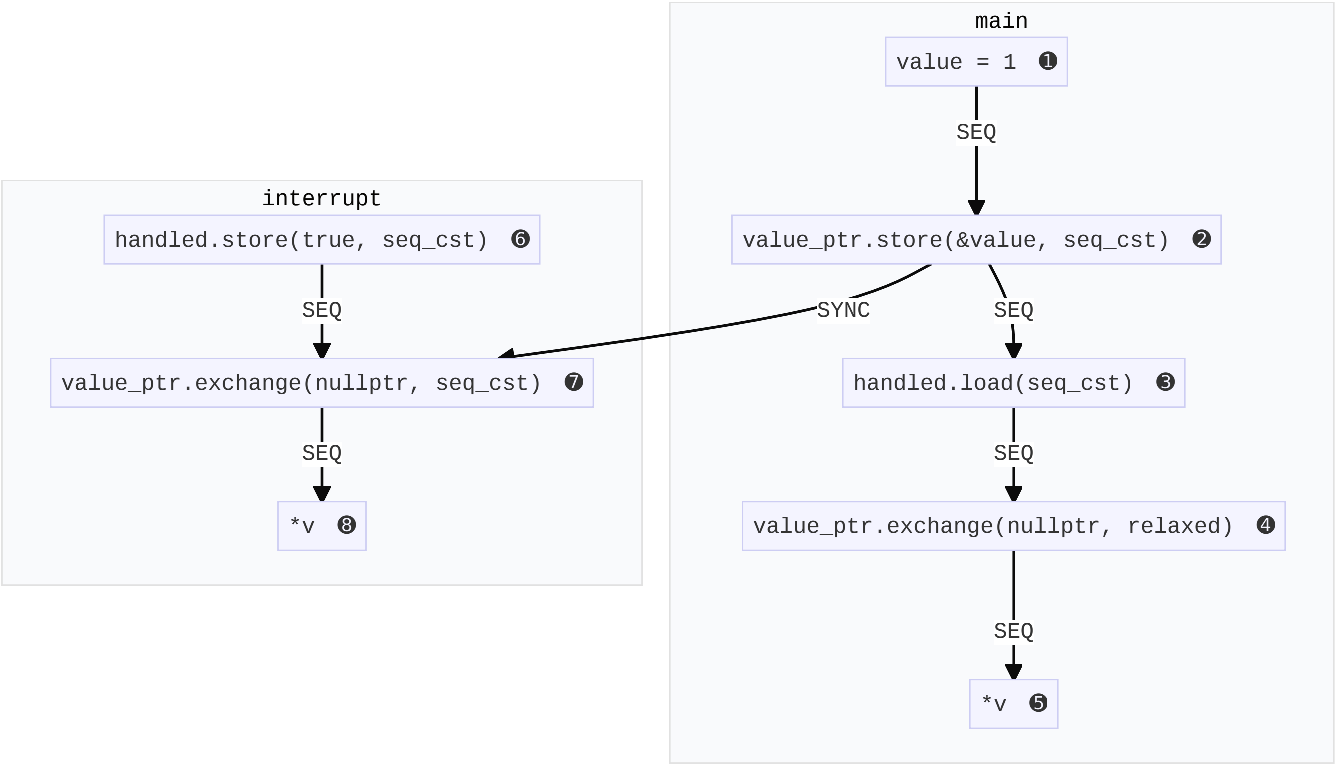
```
std::atomic_bool handled{false};
unsigned value = 0;
std::atomic<const unsigned *> value_ptr{nullptr};
std::atomic_uint total{0};
```

Main

```
// Store the value to be handled
value = 1; //                                                             ❶
value_ptr.store(&value, std::memory_order_seq_cst); //                    ❷ ▲▲▲▲
// Check if the interrupt was handled before we ran
auto h = handled.load(std::memory_order_seq_cst); //                      ❸ ▼▼▼▼
// If the interrupt ran earlier, it might not have seen our value
if (h) {
    // See if our value is still there
    auto v = value_ptr.exchange(nullptr, std::memory_order_relaxed); //   ❹
    // And handle it ourselves
    if (v)
        total.fetch_add(*v, std::memory_order_relaxed); //                ❺
}
```

Interrupt

```
// Notify the main thread that the interrupt ran
handled.store(true, std::memory_order_seq_cst); //                        ❻ ▲▲▲▲
// Read the value from the main thread
auto v = value_ptr.exchange(nullptr, std::memory_order_seq_cst); //       ❼ ▼▼▼▼
// Handle it
if (v)
    total.fetch_add(*v, std::memory_order_relaxed); //                    ❽
```



main
value = 1 ❶
SEQ
value_ptr.store(&value, seq_cst) ❷
SYNC
SEQ
handled.load(seq_cst) ❸
SEQ
value_ptr.exchange(nullptr, relaxed) ❹
SEQ
*v ❺
interrupt
handled.store(true, seq_cst) ❻
SEQ
value_ptr.exchange(nullptr, seq_cst) ❼
SEQ
*v ❽