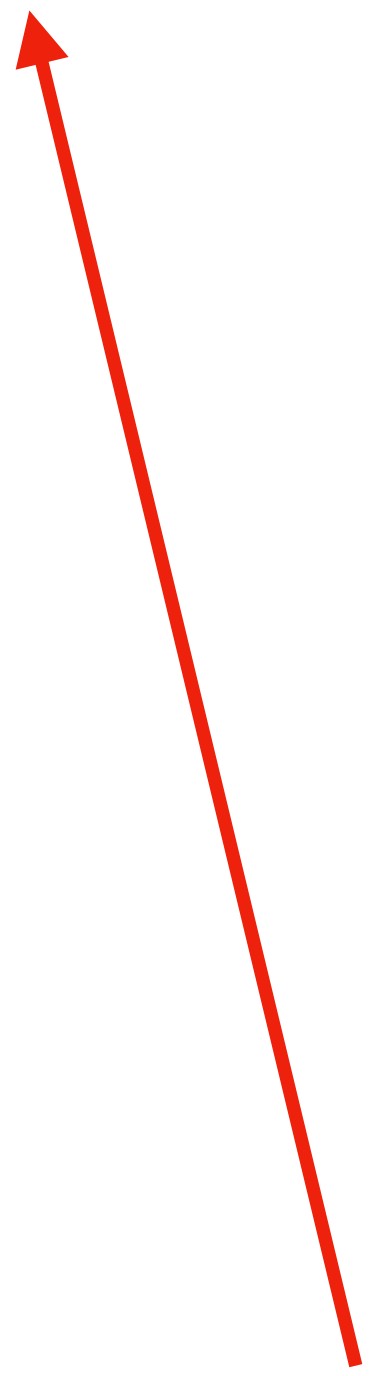
```cpp
std::atomic<float> gain (1.0f);


void processSensorData (float* sensorInOut, int n)
{
    // do some dsp
    ...

    for (int i = 0; i < n; ++i)
        sensorInOut[i] *= gain.load();
}

// called on another thread
void setSensorGain (float newGain)
{
    gain.store (newGain);
}
```

Ensures loads and stores are synchronised



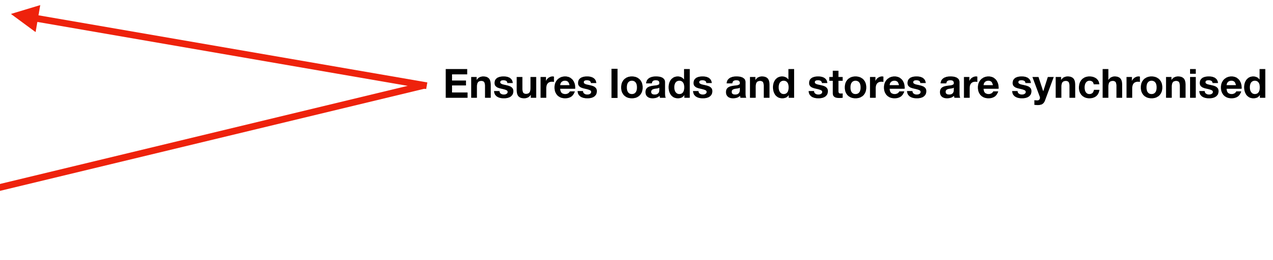
49

```cpp
std::atomic<float> gain (1.0f);
```

```cpp
void processSensorData (float* sensorInOut, int n)
{
    // do some dsp
    ...

    for (int i = 0; i < n; ++i)
        sensorInOut[i] *= gain.load();
}

// called on another thread
void setSensorGain (float newGain)
{
    gain.store (newGain);
}
```



```cpp
std::atomic<float> gain (1.0f);
static_assert (std::atomic<float>::is_always_lock_free);

void processSensorData (float* sensorInOut, int n)
{
    // do some dsp
    ...

    for (int i = 0; i < n; ++i)
        sensorInOut[i] *= gain.load();
}

// called on another thread
void setSensorGain (float newGain)
{
    gain.store (newGain);
}
```

Ensure manipulating a float is lock-free on your machine

Ensures loads and stores are synchronised