Cache Coherence

To implement CPUs, GPUs and more generaly efficient harware application, we need a way to cache the data we access the most. But these caching technics may introduce concurency limitations, because two parallel components may have different view on the same data due to mis-synchronisation of their caches. It is important to have a way to synchronise cache resources in software or hardware. This document present a homemade minimal but very powerfull cache coherence protocol.

States

- Shared: The cache line is shared between multiple caches
- Owned: The cache line is shared between multiple caches, this line is dirty (the value in the main memory is not up-to-date), and this cache is the only owned of the cache line, so it is in charge of updating the memory.
- Exclusive: The cache line is only in this cache and is clean.
- Modified: The cache line is only in this cache but is dirty.
- Invalid: The cache line is not present in this cache.

I also provide a special pending state for cache lines in transition. If a CPU want to do an operation in the cache with a line in this state, it must wait the cache line to be in a valid state.

Channels

- Cache acquire requests (M -> S)
- Controller invalidations (S -> M)
- Cache release message (M -> S)
- Controller response (S -> M)
- Cache final message (M -> S)

Messages

- Acquire

```
enum Perms {
  Owned,
  Borrow
}

struct Acquire {
  address,
  perms: Perms
}
```