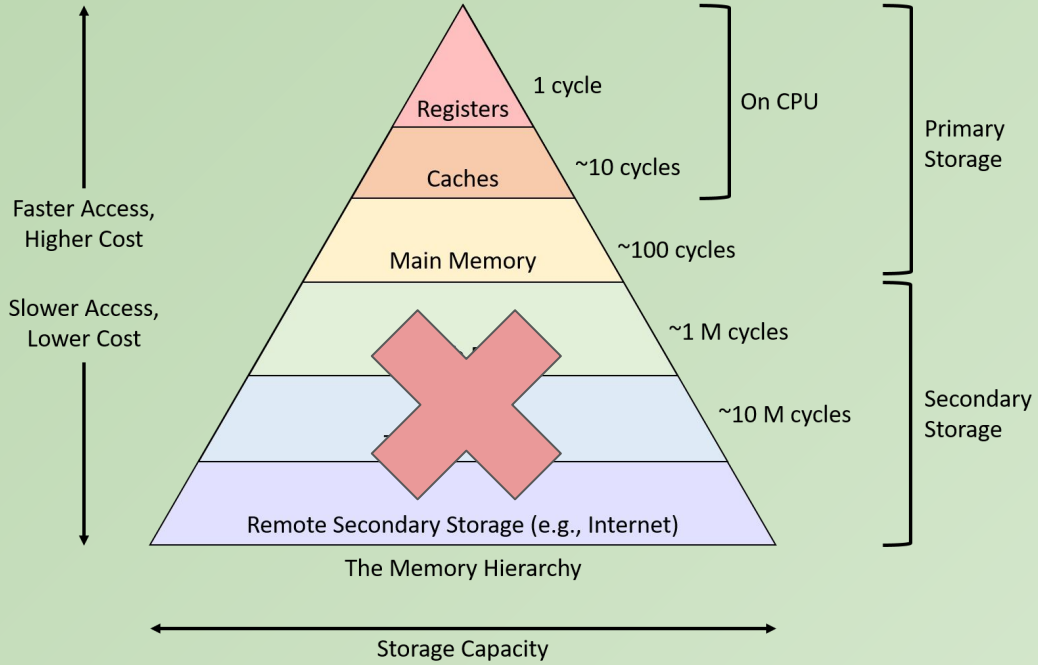
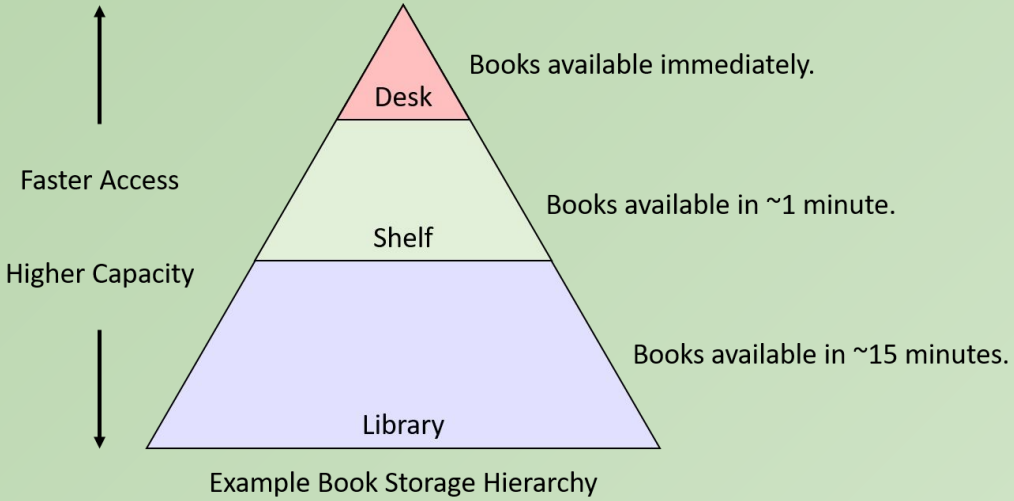

OX
CB
BARE
METAL



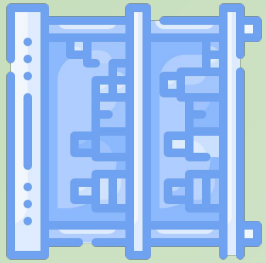
The Memory Hierarchy

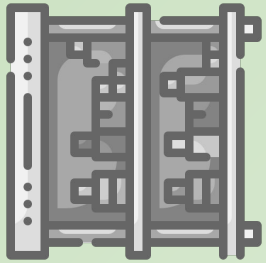


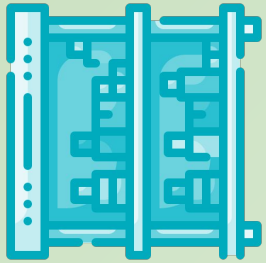
Example Book Storage Hierarchy

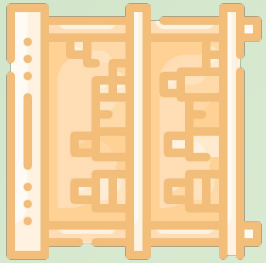
← “Temporal” locality

Cache rules everything around me

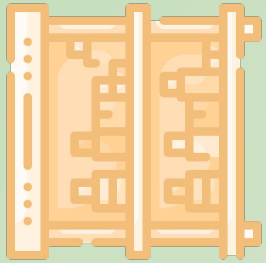


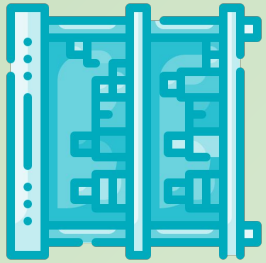


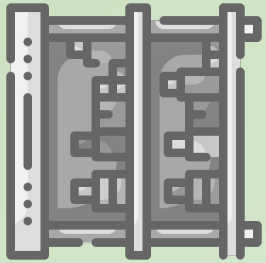


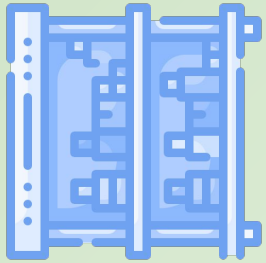


Cache rules everything around me

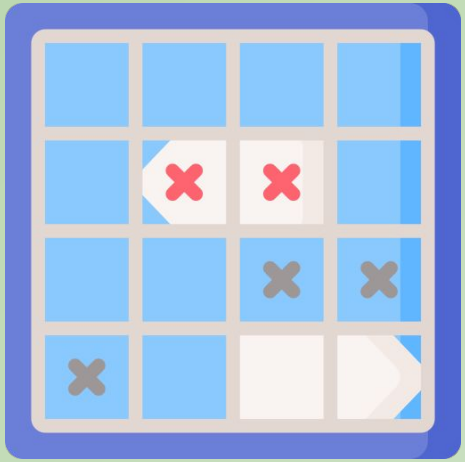


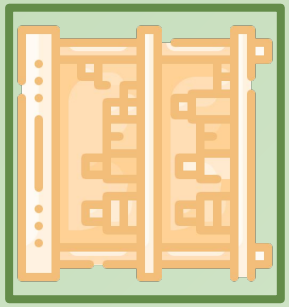


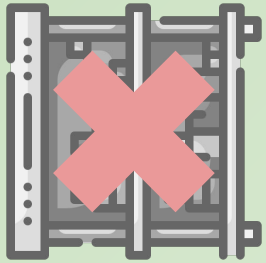


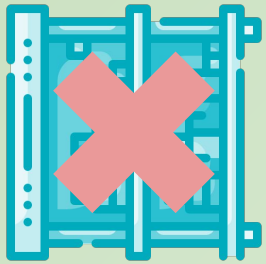


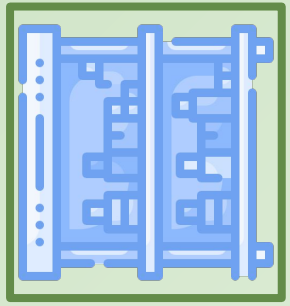
Cache rules everything around me





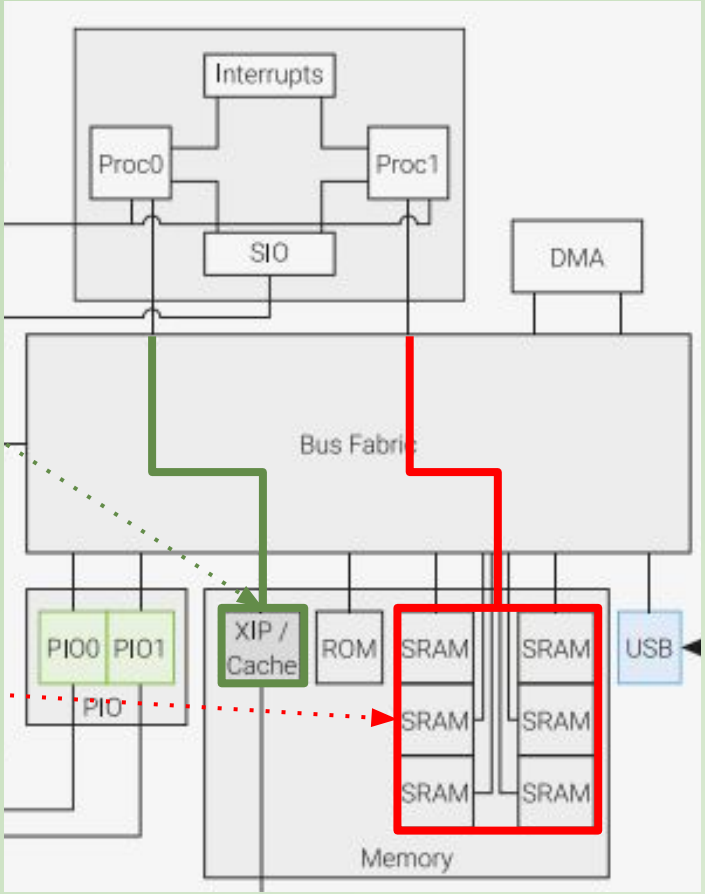






- Single source
- Direct line to processing unit

- Many sources
- First have to figure out *where* data is



```
for (int i = 0; i < size; i++) {
    sum += array[i];
}
```

Spatial Locality

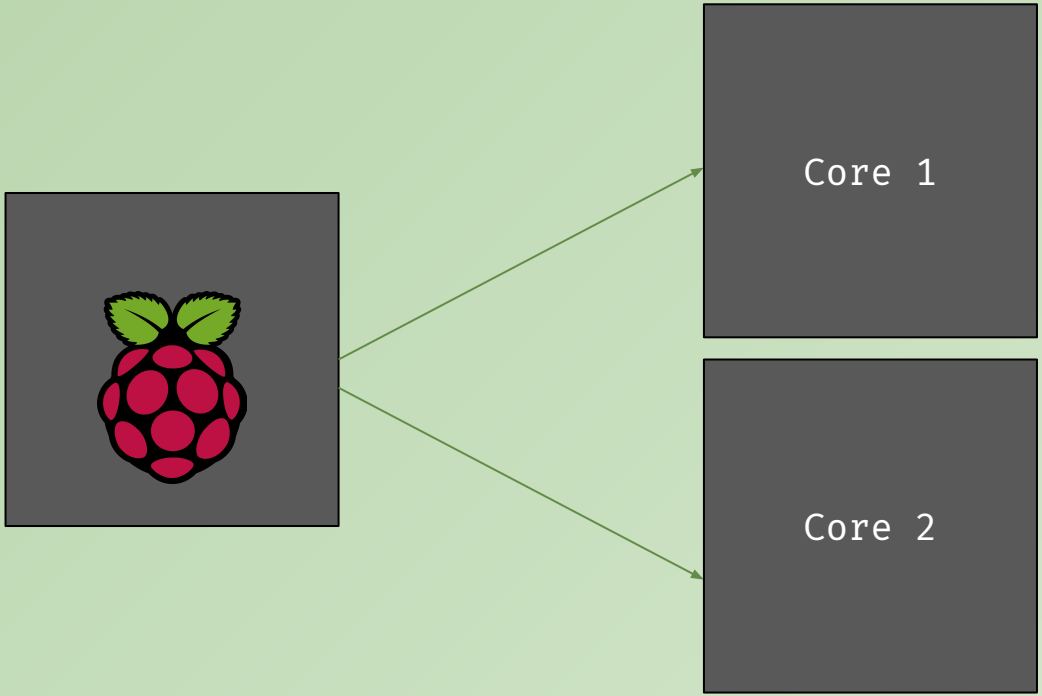
```
0000000020041fc0 93 11 00 00 00 00 00 00 01 00 00 00 02 00 00 00
0000000020041fd0 03 00 00 00 04 00 00 00 05 00 00 00 06 00 00 00
0000000020041fe0 07 00 00 00 08 00 00 00 09 00 00 00 0a 00 00 00
```

We can naively prove the effectiveness of the cache by looking at program execution times with data *in* and *out* of the cache.

$$net\ time = time\ at\ end - time\ at\ start$$

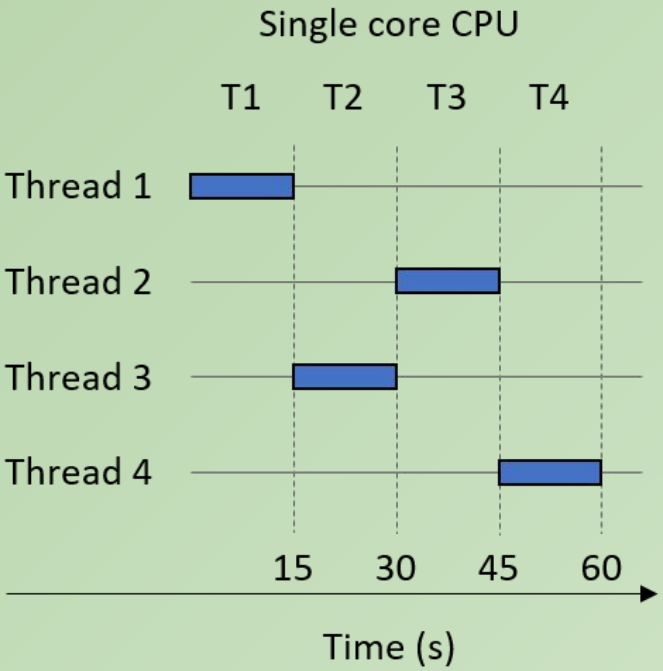
$$misses = \frac{num_{elements}}{\frac{block\ size}{size_{elements}}}$$

Cortex M0+



- Both have equal compute power
- Both run simultaneously
- So far, we've only used 1

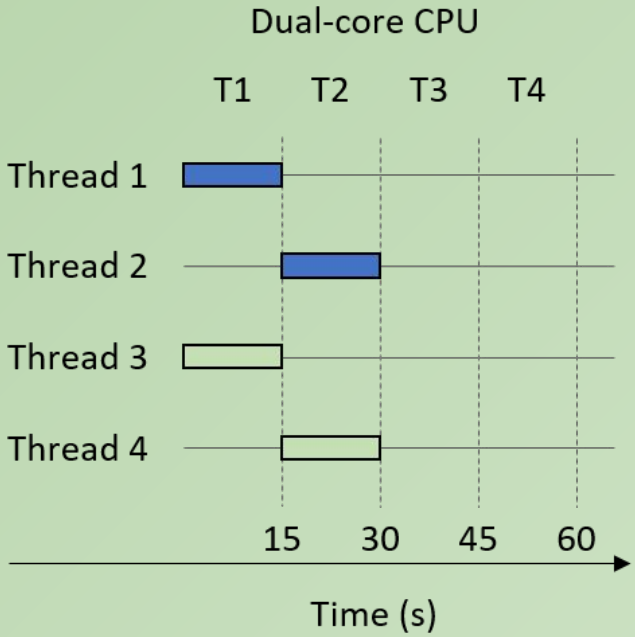
0x BARE C6 METAL



“Synchronous”

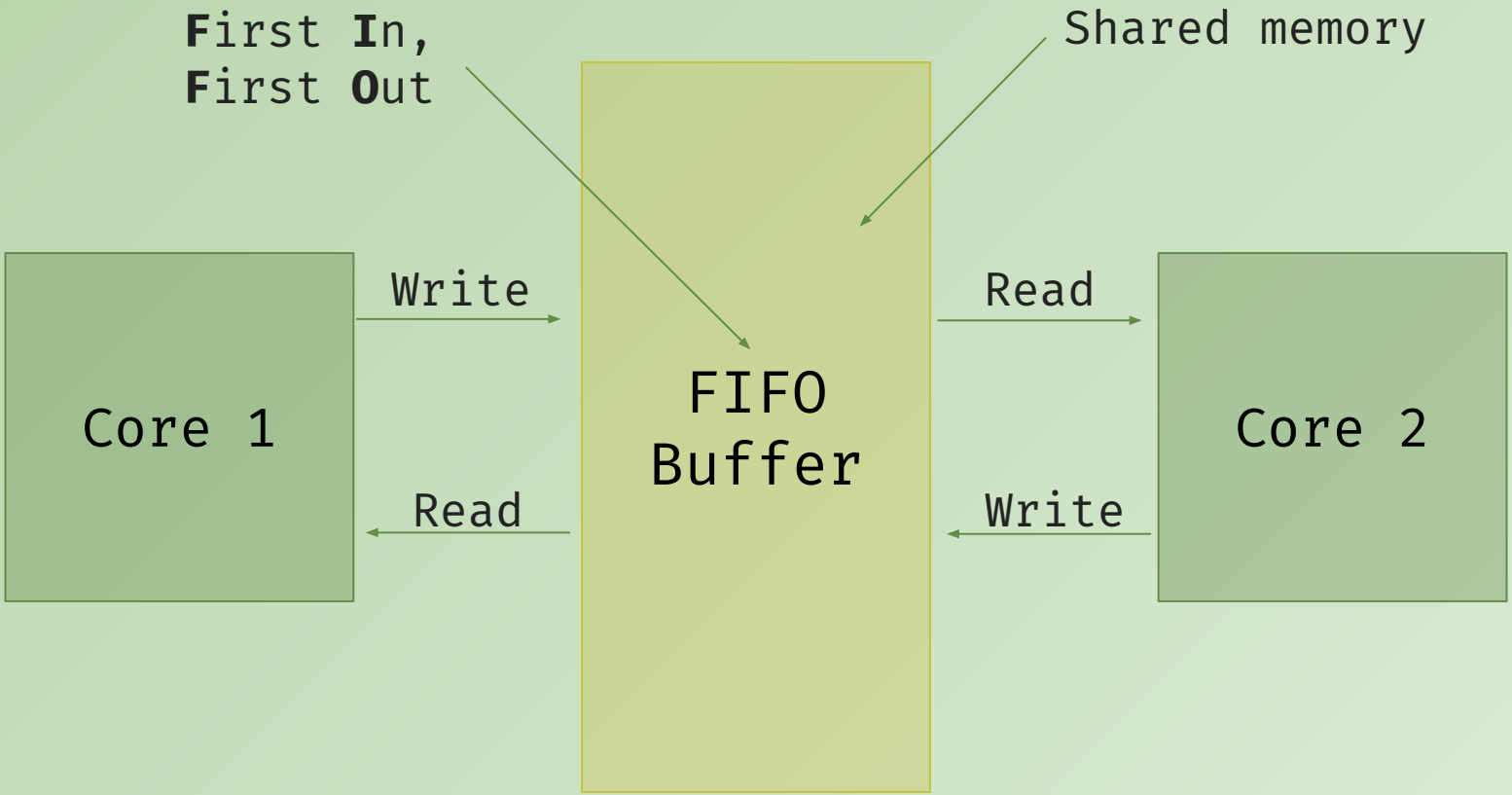
Each process has to finish before another can begin

Compute time distributed on a single-core processor



Each core is **synchronous** in itself, but performance is effectively *asynchronous.*

Compute time distributed on a two-core processor



FIFO Buffer

"Buffer"

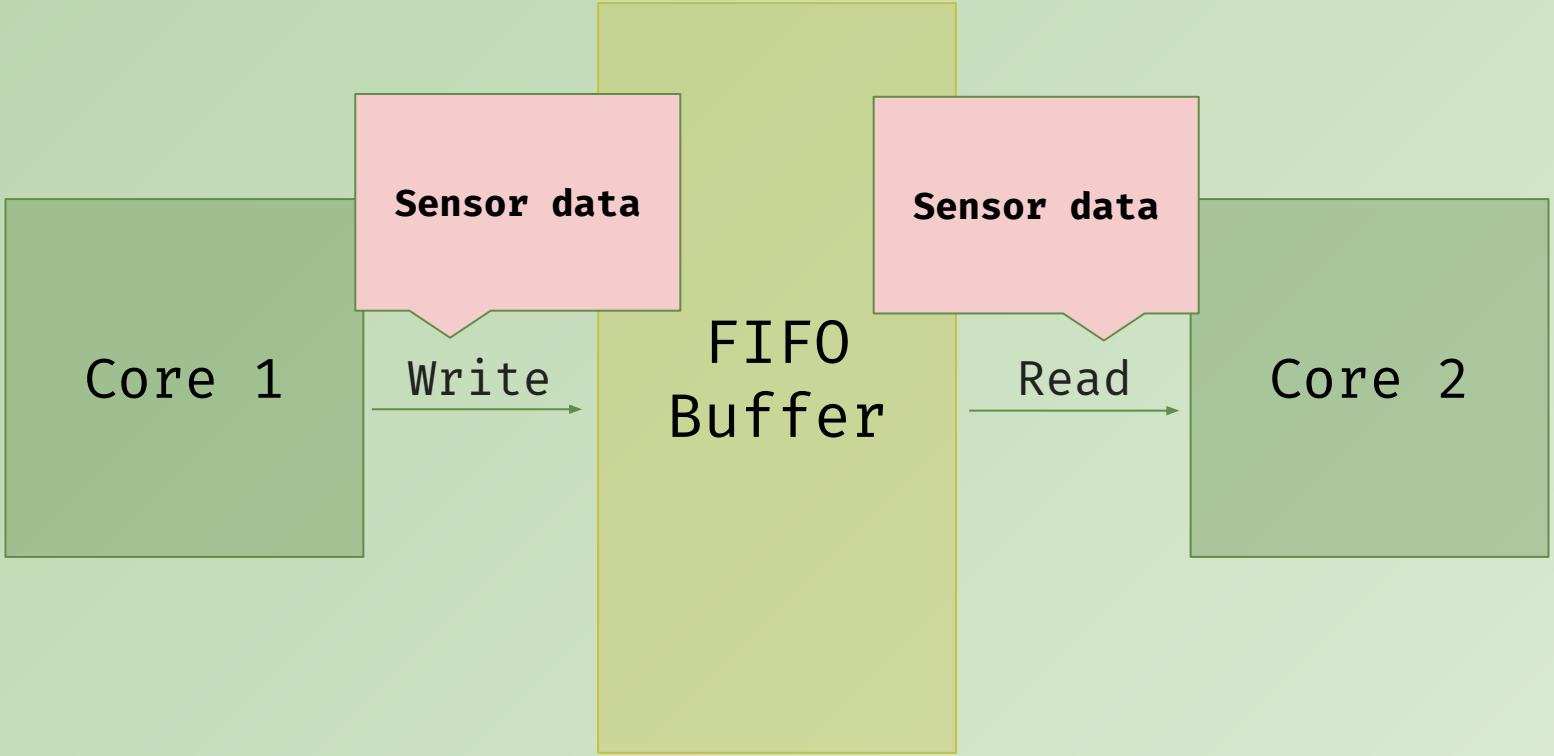
Location that temporarily stores data in transit from one place to another

- This buffer can only hold **integers**

Becomes important later

All in the timing

Cores	Approach	Speed Up	Efficiency
1	Synchronous		

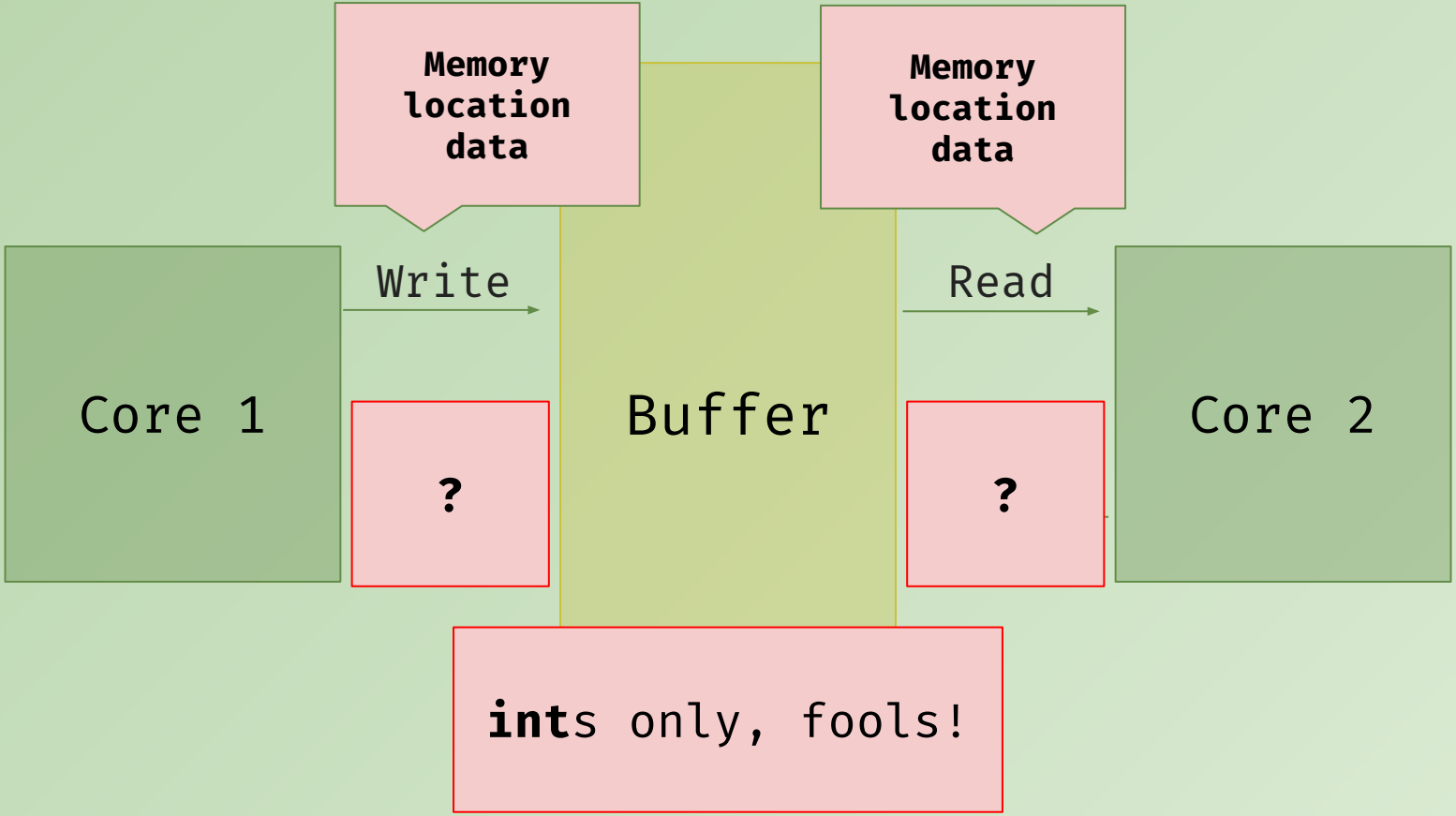


Time on 1 core

$$Speedup_c = \frac{T_1}{T_c}$$

Time on n cores

$$Efficiency_c = \frac{T_1}{T_c \times c} = \frac{Speedup_c}{c}$$


Memory location data
Memory location data
Write
Read
Core 1
Buffer
Core 2
?
?
ints only, fools!

Casting in C

```
float decimal = -1.302;
/*
    Some GCCs have uint, all will have
    unsigned int; ours has uint
*/
uint int_value = (uint) decimal;
printf("%u", int_value);

>> 4294967295
```

free()'ing memory

```
void free_data(struct data *head) {
    struct data *tmp;
    while(head != NULL) {
        tmp = head→next;
        free(head);
        head = tmp;
    }
}
```

Create temporary storage for head node

free() individual address for head node; make available to system

Set head node to next memory address