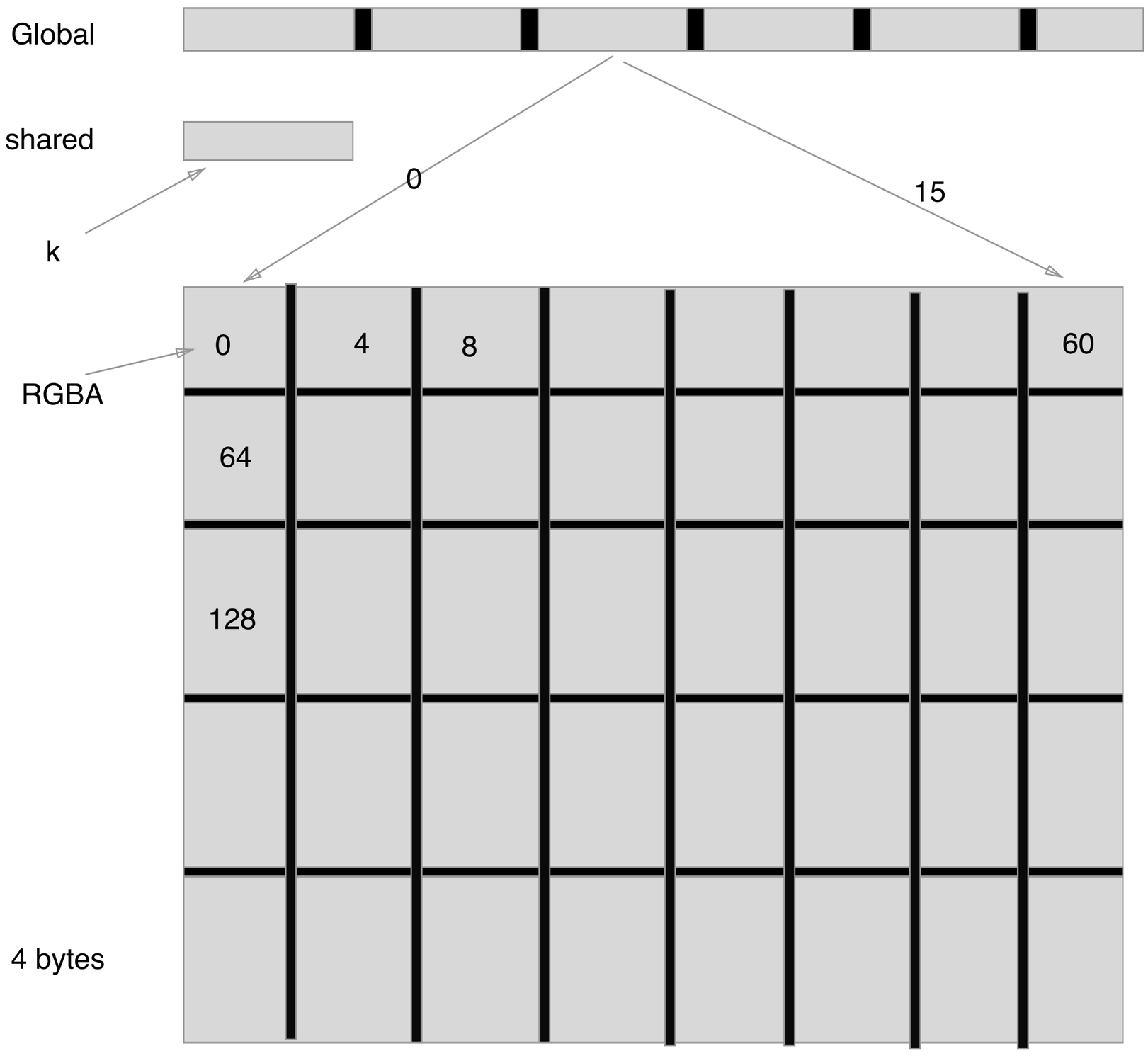


block = 128 threads
= 0 - 127

RGBA 0 -> 0-3
1 -> 4-7
.
.
.
127 -> 508-511

we can try 16 or 32 threads/block
c.c. 20

L1 memory can be split in the following ways
16 kb - default shared memory
32 kb
48 kb
you cannot do more pixels per thread because you run out of shared memory and it will run slowly
4096 = 128 x 32

if thread does more than 1 pixel, then write 0, 64, 128 , 192
1, 65 , 129, 193

**Texture memory**
- RMN story

**Guidelines for choosing Blobk and Grid Design**

Guidelines:
1. The hardware should be as much occupied and busy as possible.
   - Don't waste threads with this
2. The execution should involve as little stalling as possible
   - if else branches
   - global memory I/O
   - shared memory collisions
   - synchronisation

Practical choises:
1. A thread should do more work that can ideally be executed in parallel by a warp (32 - threads)
   - not little kernels, rather larger
2. Use more registers per thread to avoid using shared memory(if you can afford it) may result in a smaller number of threads per block
3 The number of threads per block should be multiple of the warp size (128-256)
4. The grid (totality of the blocks) should provide multiple(3-4) blocks per streaming of the processor(SM)
5. Use blocks big enough to cover multiple warps

**How many threads per block?**

$$\min\left(\begin{array}{l}\text{numOfWarpSched} * \text{warpSize}\\ \text{regsPerBlock} / \text{regsPerThread}\\ \text{sharedRamSize} / \text{sharedRamPerThread}\\ \text{MaxThreadsPerSM}\end{array}\right)$$

GPU model -> architecture -> specs

# Streams

$a_n[i]$++; // slow if done on global memory because it actually does 1 read and 1 write

atomicAdd
atomicXchg
atomicAddAndXchg