CS Space
Клуб технологий и науки

Вычисления на видеокартах

Лекция 6

- Bitonic sort
- Radix sort
- Советы по отладке

@UnicornGlade
@PolarNick239
polarnick239@gmail.com
Николай Полярный


MIPT
ITMO
HSE
OpenCL
Khronos
CUDA
NVIDIA
GRANTA
TESLA T4

Task03: Умножение матриц на **Tesla T4** (8141 GFlops)

1. Mikhail Stulov - МФТИ, TBank - **4802 GFlops**, OpenCL
2. Хулиган Серега - ИТМО - **4571 GFlops**, OpenCL
3. Daniyar Salakhov - ВШЭ - **3660 GFlops**, CUDA+WMMA

```
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
```

Write

Read

Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

```
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
```

Write

Read

Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

Есть ли **RAW** гонка?
(**R**ead **A**fter **W**rite)

```
31      float res = 0;
32      for (int z_shift = 0; z_shift < k; z_shift += GROUP_SIZE_X) {
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
50      }
51      c[global_index] = res;
52  }
```

Write

Read

Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

Есть ли **RAW** гонка?
(**R**ead **A**fter **W**rite)

```
31      float res = 0;
32      for (int z_shift = 0; z_shift < k; z_shift += GROUP_SIZE_X) {
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
50      }
51      c[global_index] = res;
52  }
```

Write

Read

Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

Есть ли **RAW** гонка?
(**R**ead **A**fter **W**rite)

```
31      float res = 0;
32      for (int z_shift = 0; z_shift < k; z_shift += GROUP_SIZE_X) {
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
50      }
51      c[global_index] = res;
52  }
```

Write

Read

Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

Есть ли **RAW** гонка?
(**R**ead **A**fter **W**rite)

Как исправить?

```
31      float res = 0;
32      for (int z_shift = 0; z_shift < k; z_shift += GROUP_SIZE_X) {
33          global_a_index = global_x * k + (z_shift + local_y);
34          global_b_index = (z_shift + local_x) * w + global_y;
35
36          if (global_a_index < h * k) {
37              local_data_a[local_index] = a[global_a_index];
38          }
39          if (global_b_index < k * w) {
40              local_data_b[local_index] = b[global_b_index];
41          }
42
43          barrier(CLK_LOCAL_MEM_FENCE);
44
45          if (global_index < h * w) {
46              for (int dz = 0; dz < GROUP_SIZE_X; dz++) {
47                  res += local_data_a[local_x * GROUP_SIZE_X + dz] * local_data_b[dz * GROUP_SIZE_X + local_y];
48              }
49          }
50      }
51      c[global_index] = res;
52  }
```

Write

Read

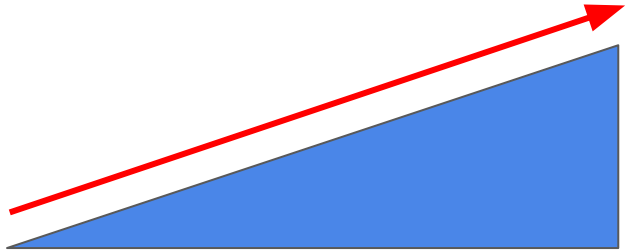
Есть ли **WAR** гонка?
(**W**rite **A**fter **R**ead)

Есть ли **RAW** гонка?
(**R**ead **A**fter **W**rite)

Как исправить?

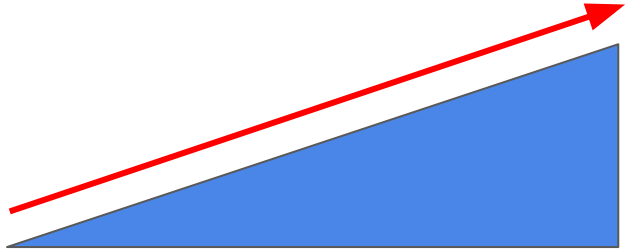
barrier!

Bitonic sort

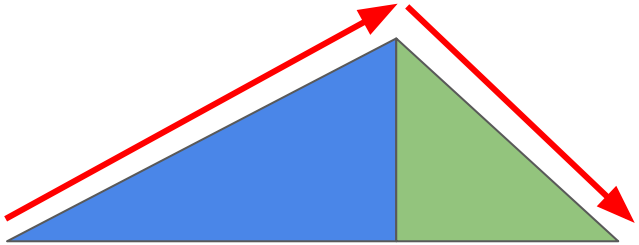


1 3 5 19 134 567 999 - монотонная последовательность

Bitonic sort

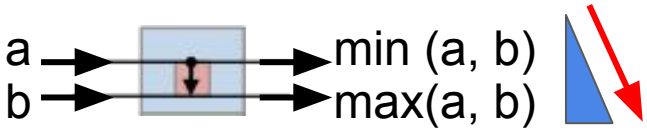


1 3 5 19 134 567 999 - монотонная последовательность

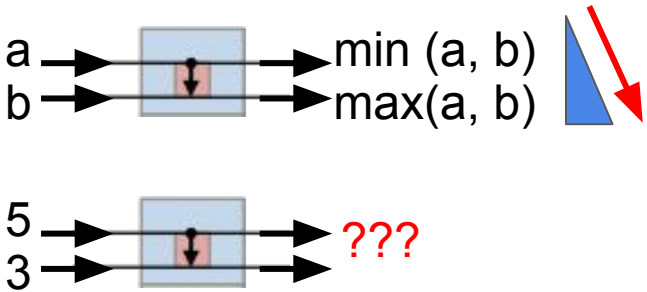


1 4 9 45 239 200 50 3 - битоническая последовательность (bitonic)

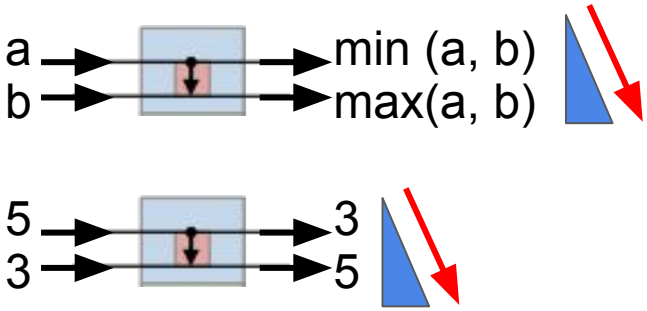
Bitonic sort



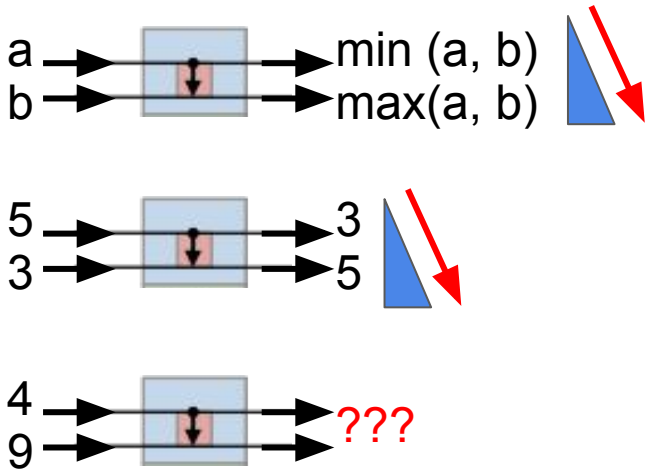
Bitonic sort



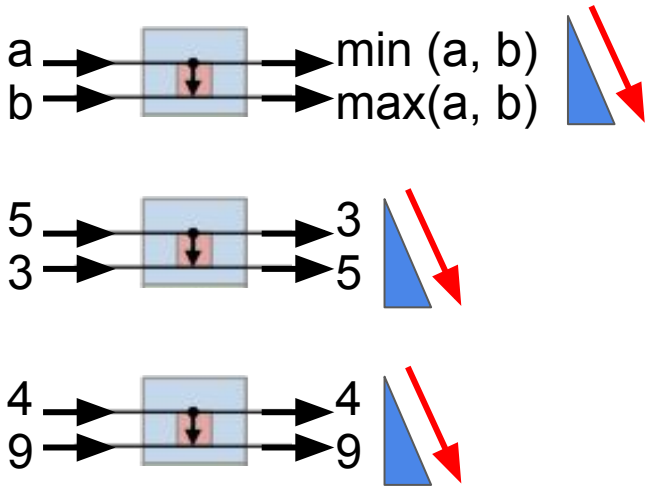
Bitonic sort



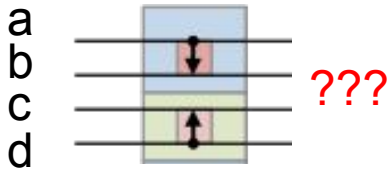
Bitonic sort



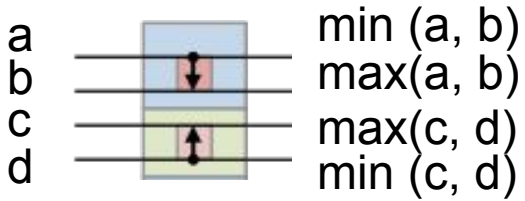
Bitonic sort



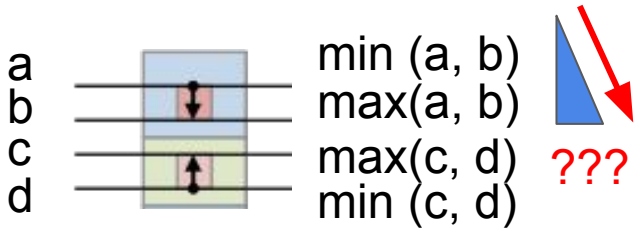
Bitonic sort



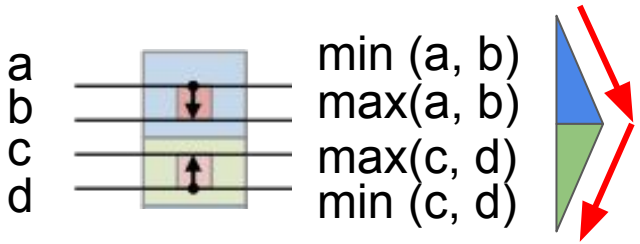
Bitonic sort



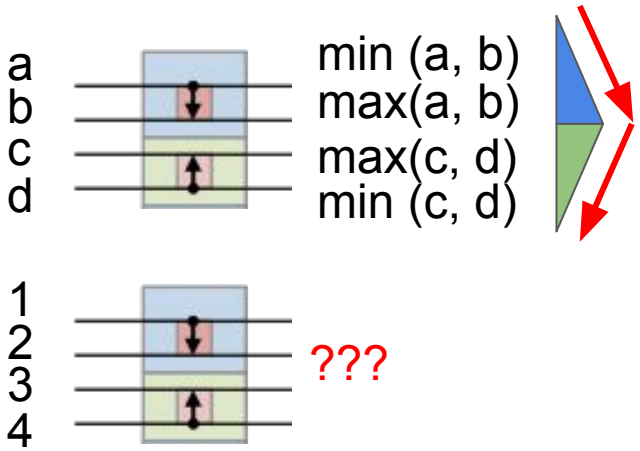
Bitonic sort



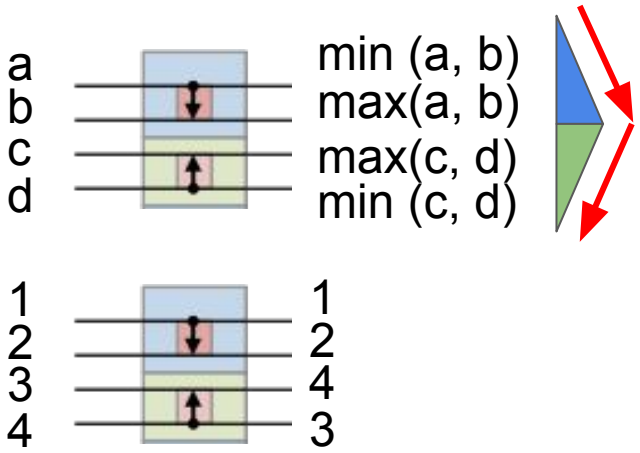
Bitonic sort



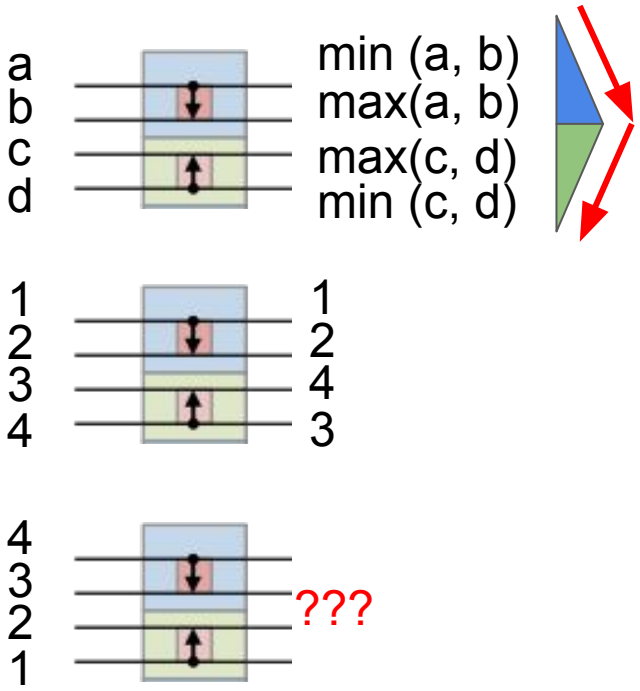
Bitonic sort



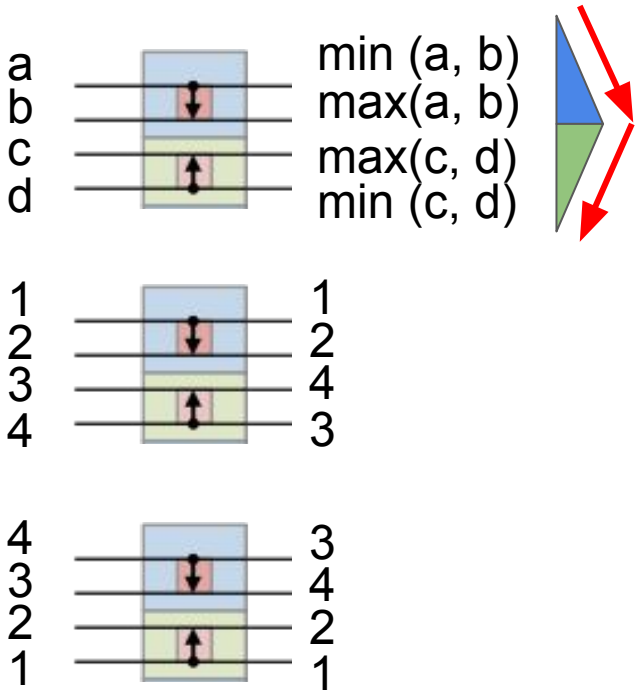
Bitonic sort



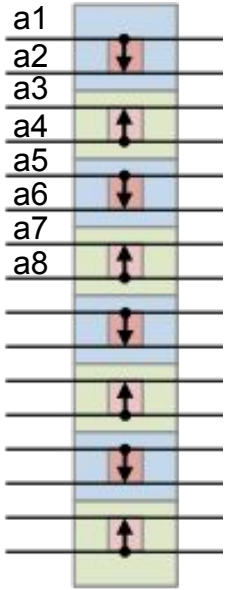
Bitonic sort



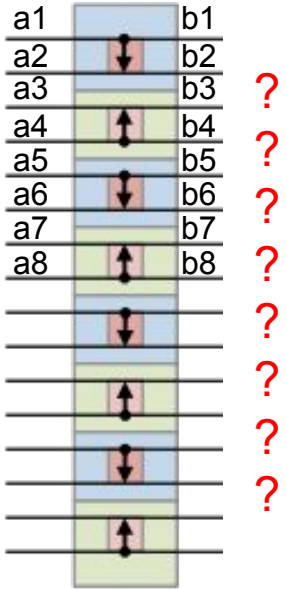
Bitonic sort



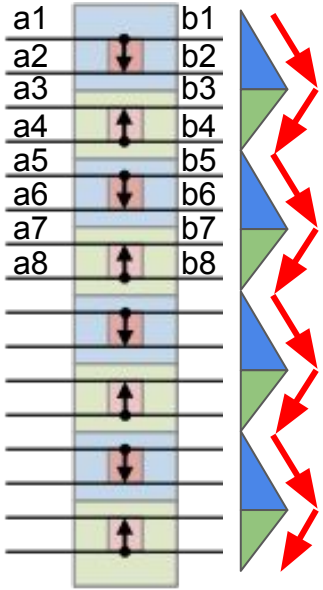
Bitonic sort



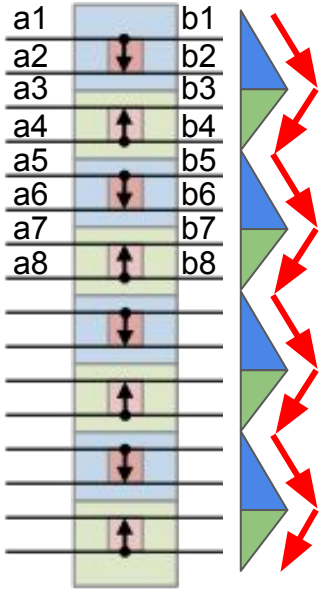
Bitonic sort

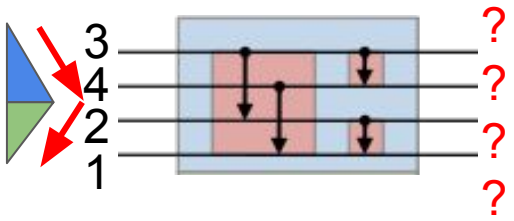


Bitonic sort

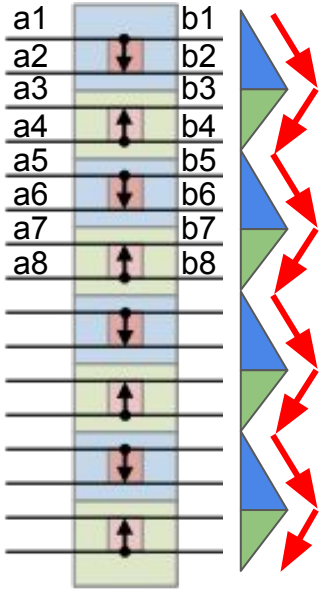


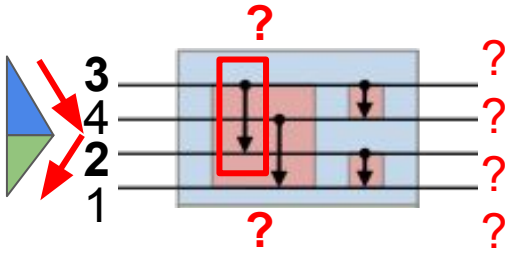
Bitonic sort



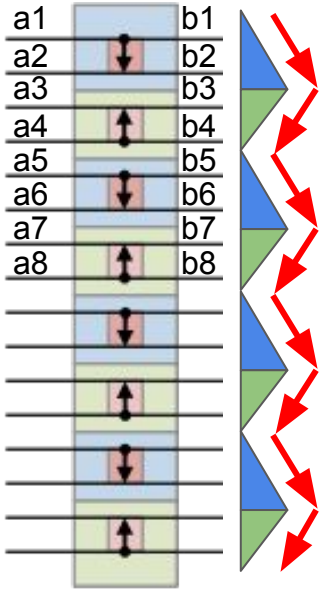


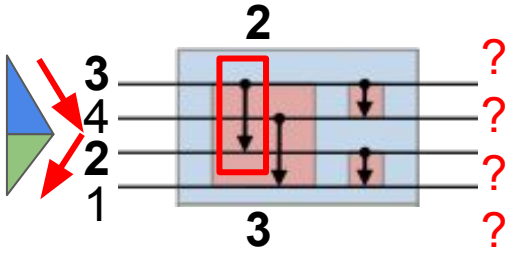
Bitonic sort



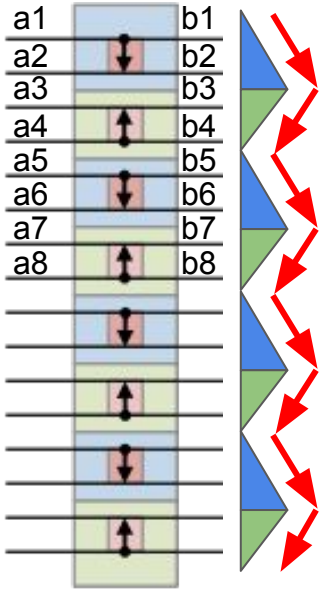


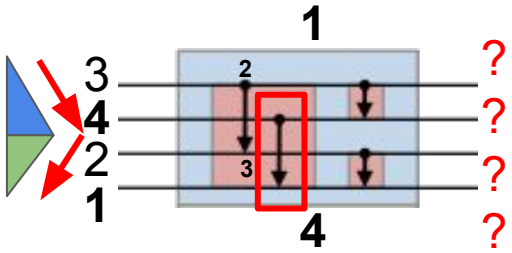
Bitonic sort



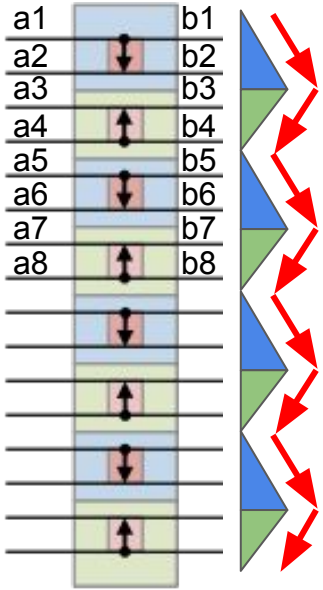


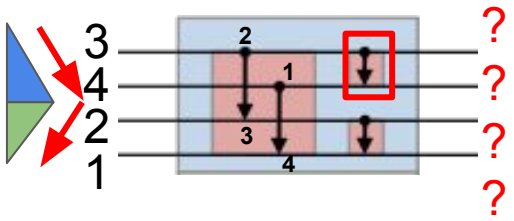
Bitonic sort



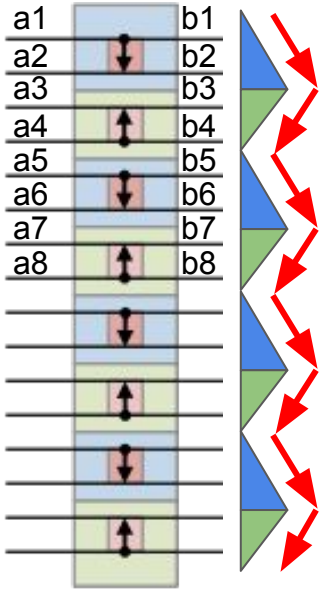


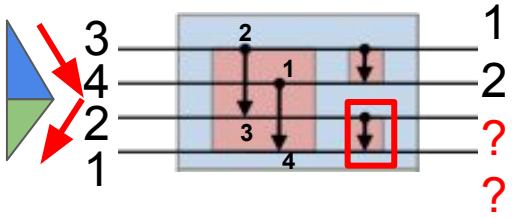
Bitonic sort



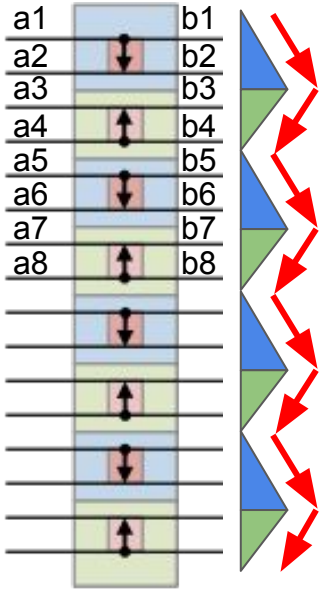


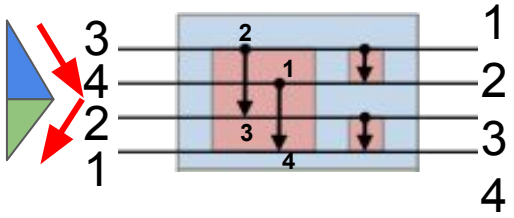
Bitonic sort



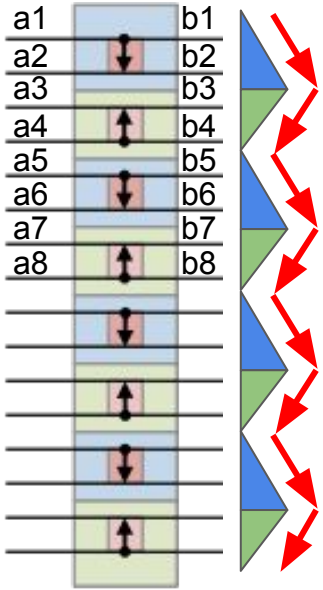


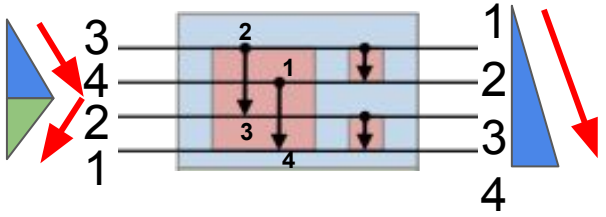
Bitonic sort



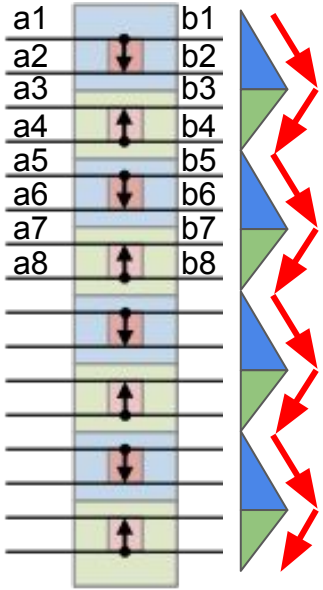


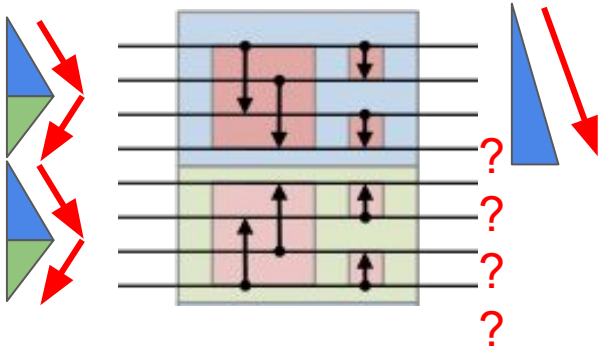
Bitonic sort



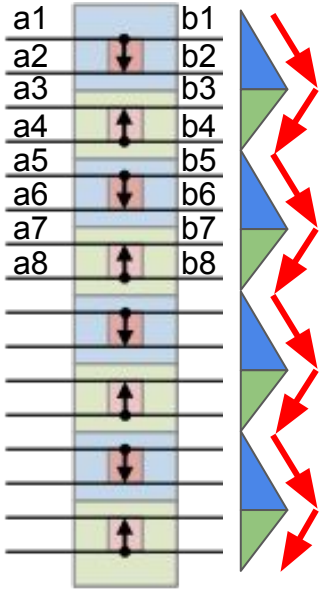


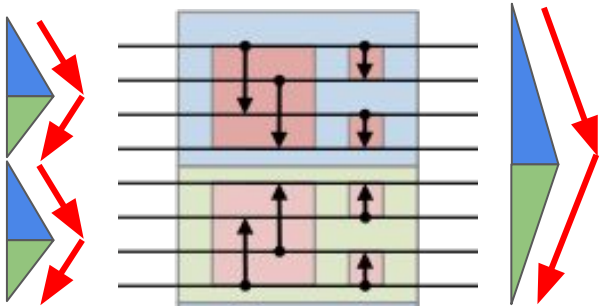
Bitonic sort



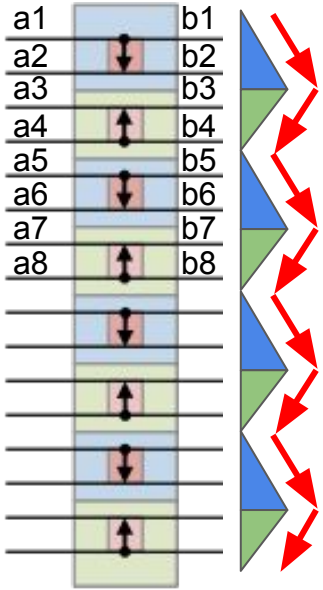


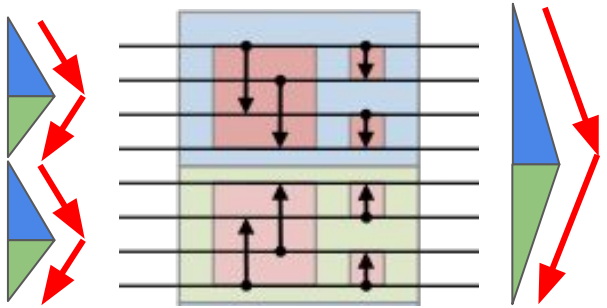
Bitonic sort





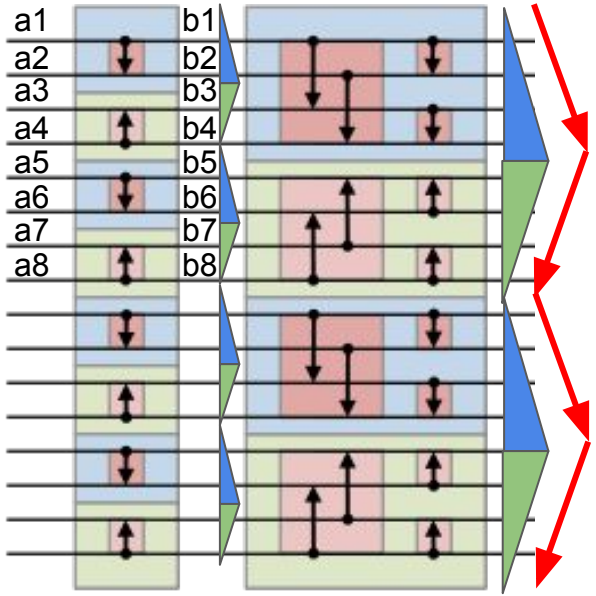
Bitonic sort



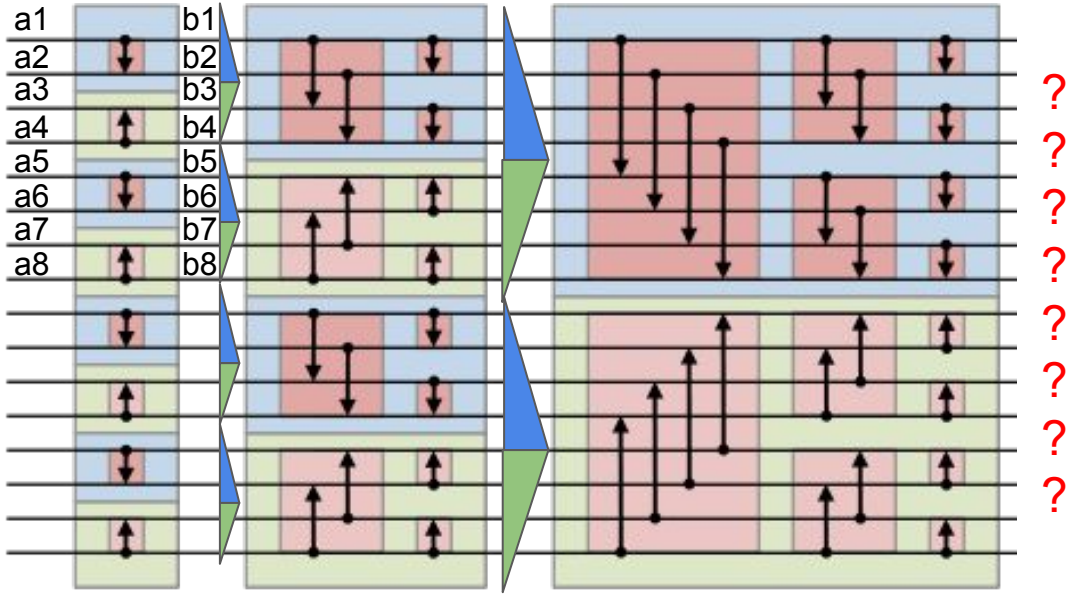


Есть ли идеи как сделать сортировочную сеть?

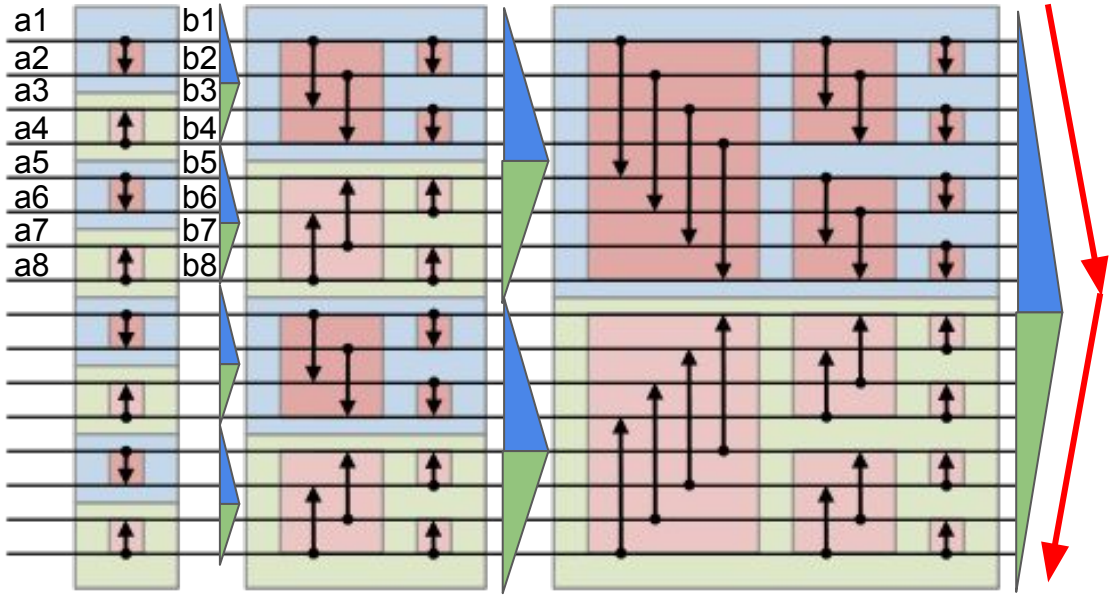
Bitonic sort



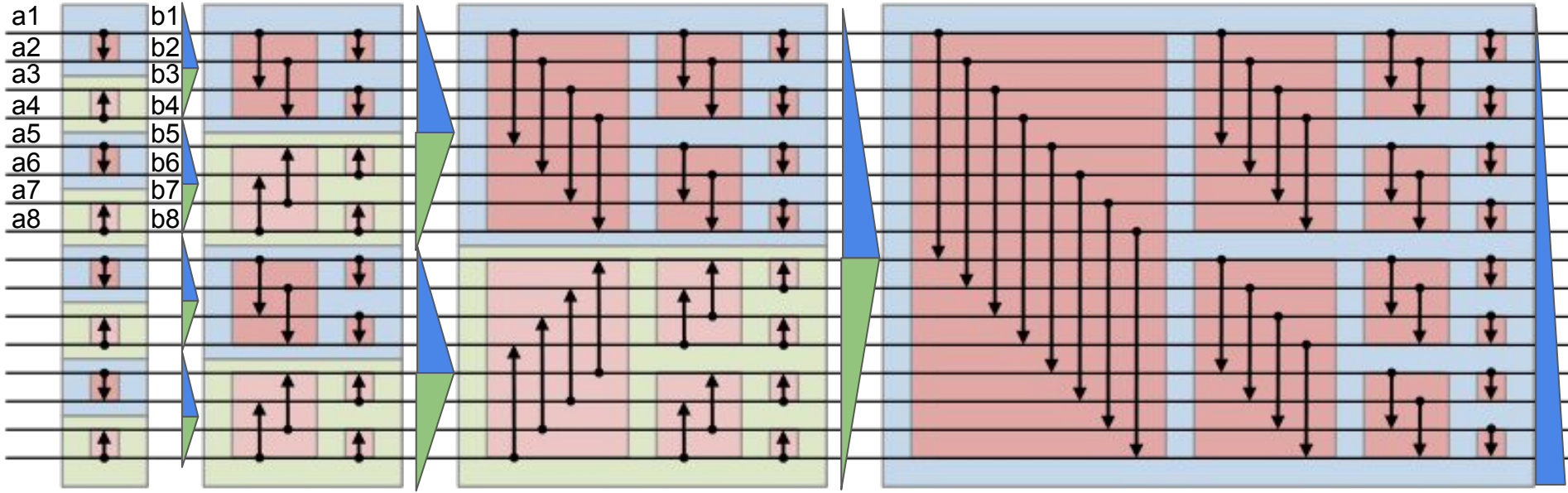
Bitonic sort



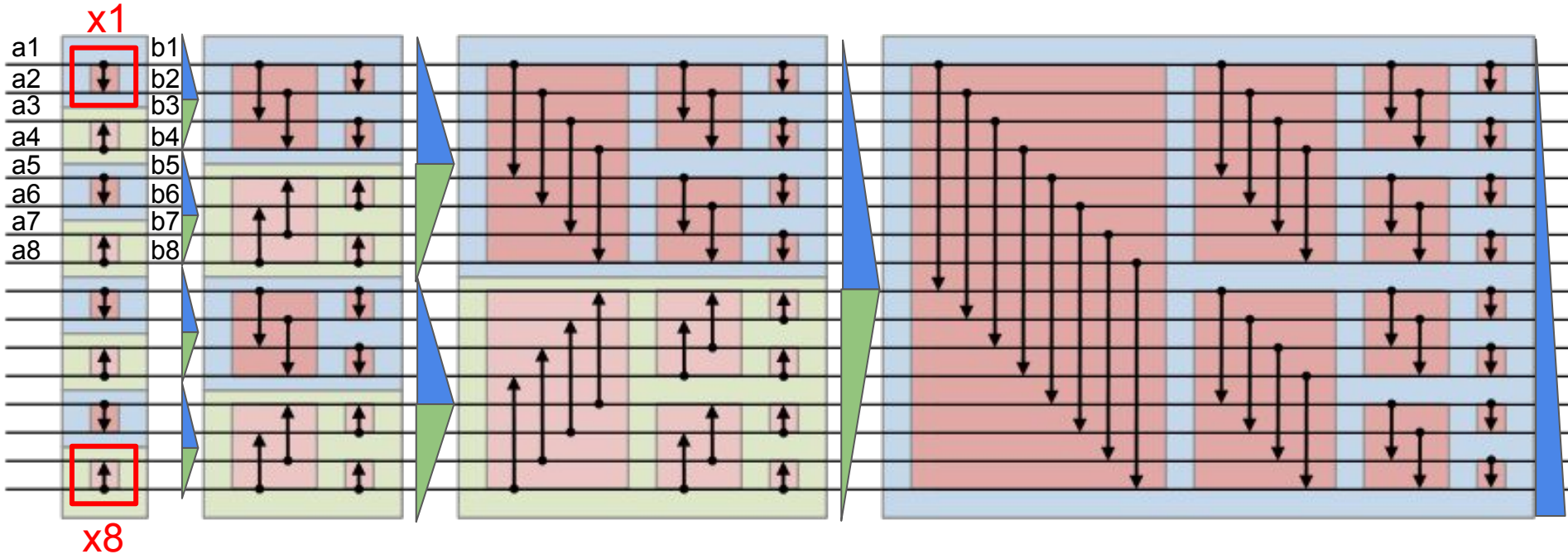
Bitonic sort



Bitonic sort

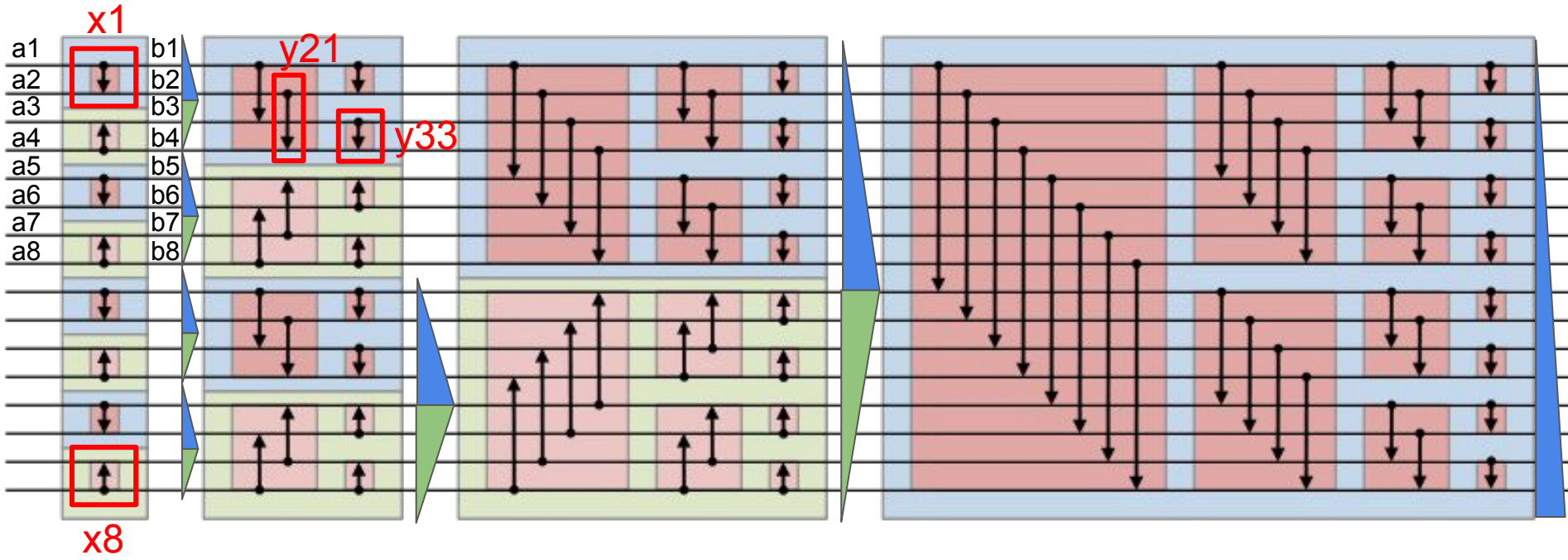


Bitonic sort



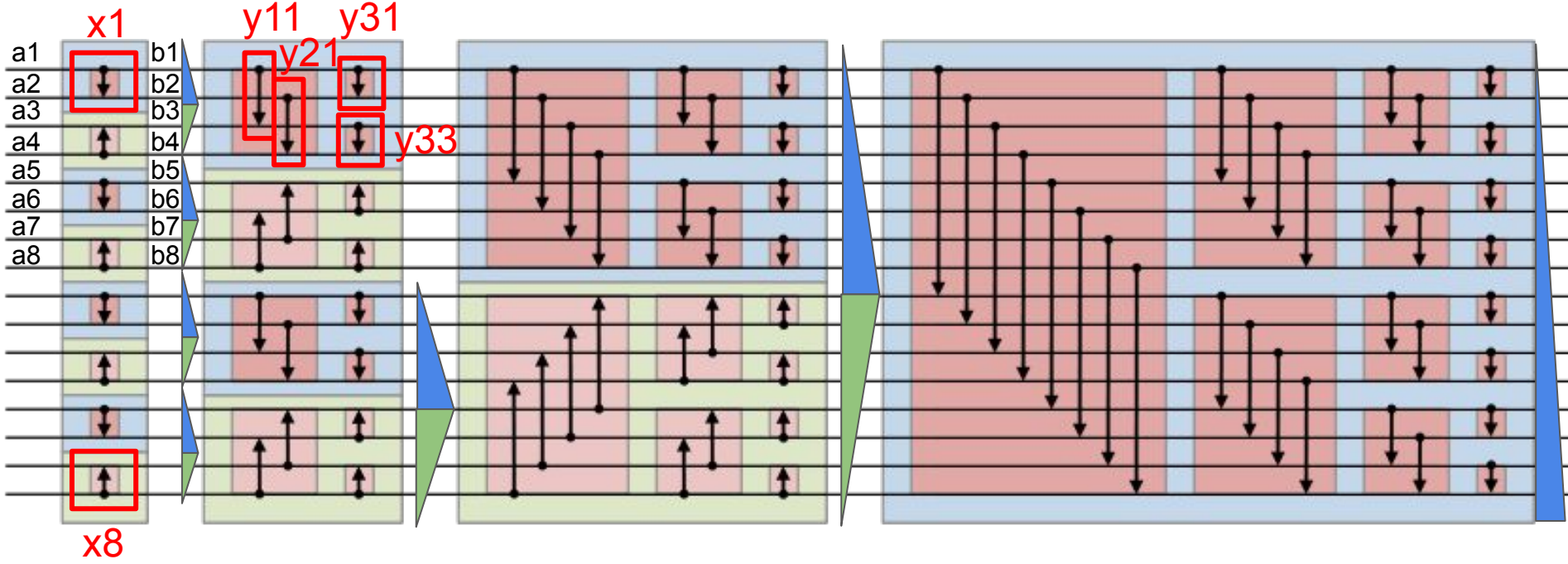
Какие операции можно делать параллельно между собой?

Bitonic sort



Какие операции можно делать параллельно между собой?

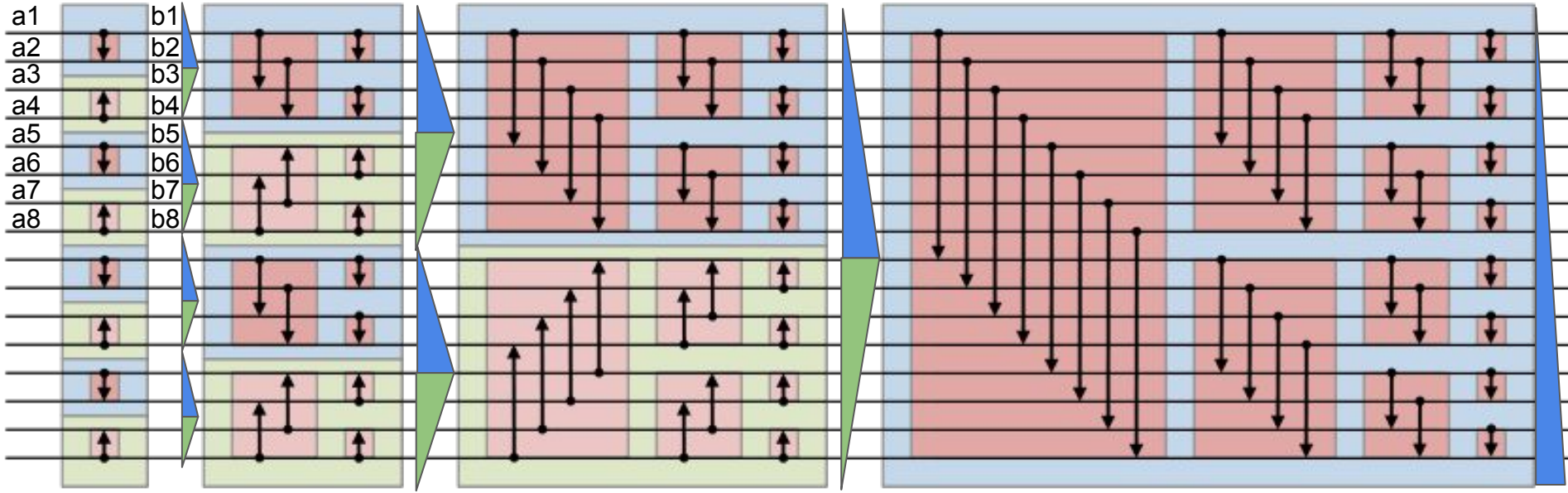
Bitonic sort



Какие операции можно делать параллельно между собой?

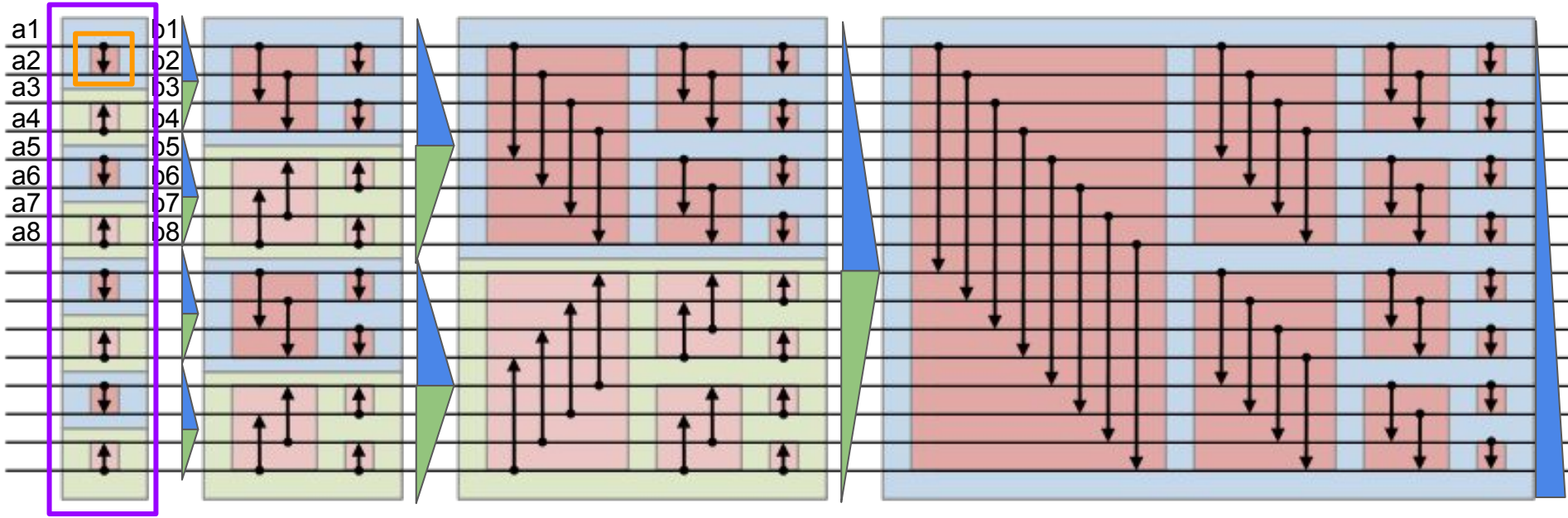
Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



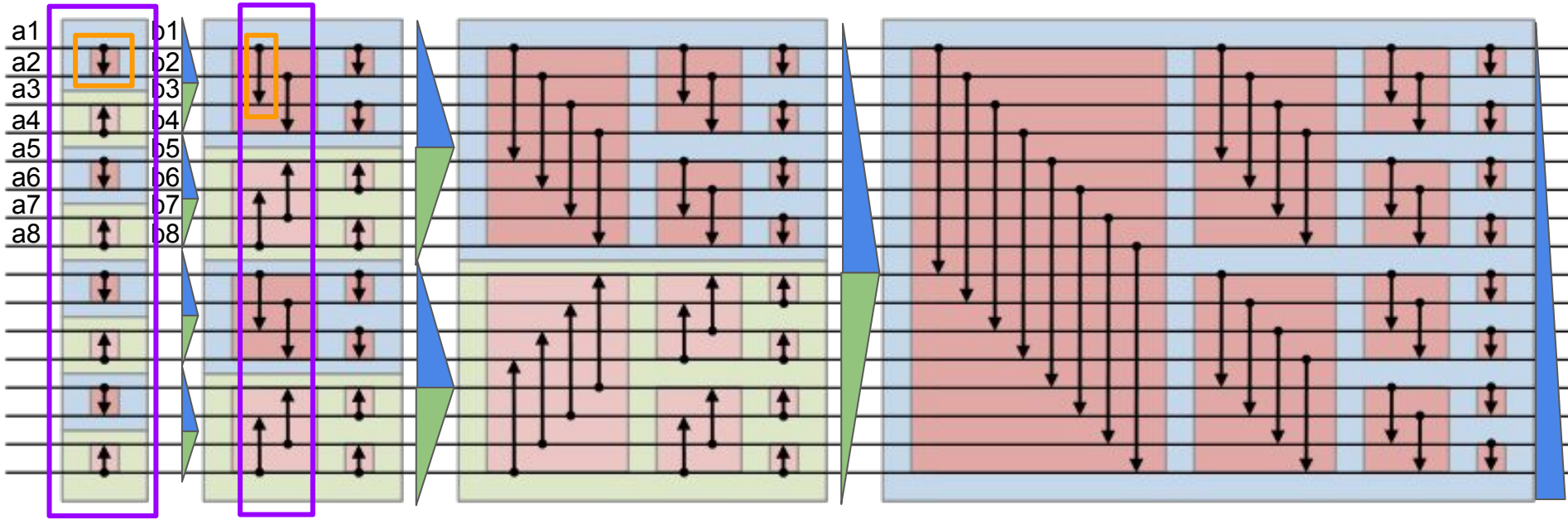
Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



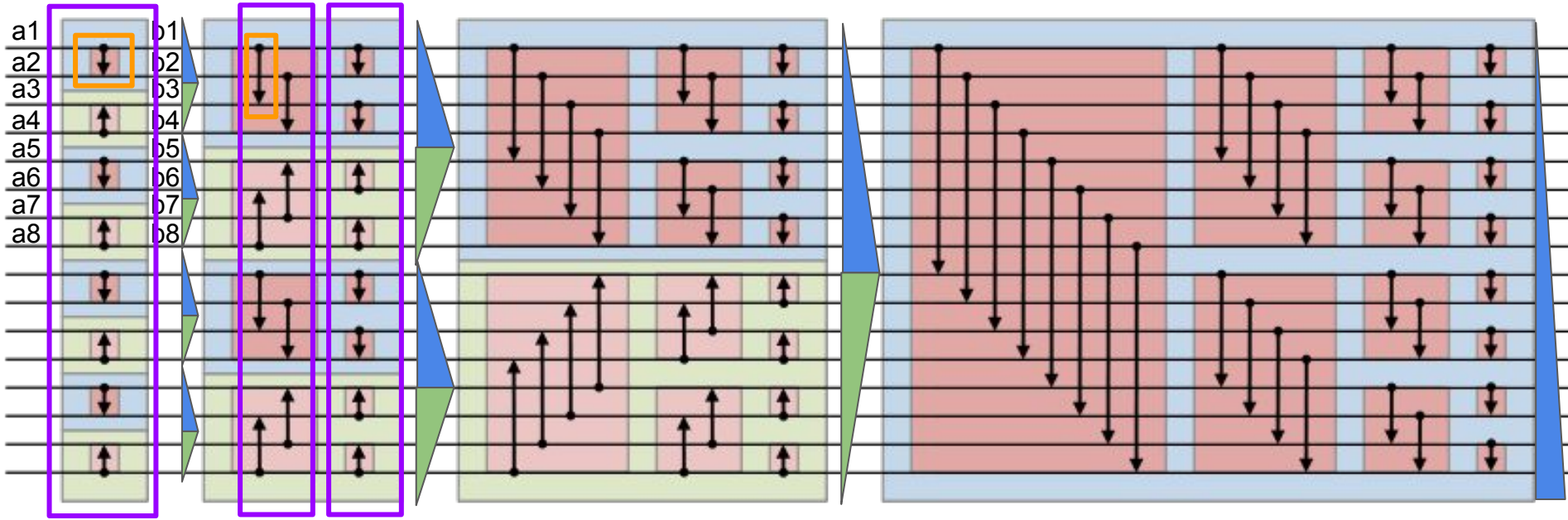
Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



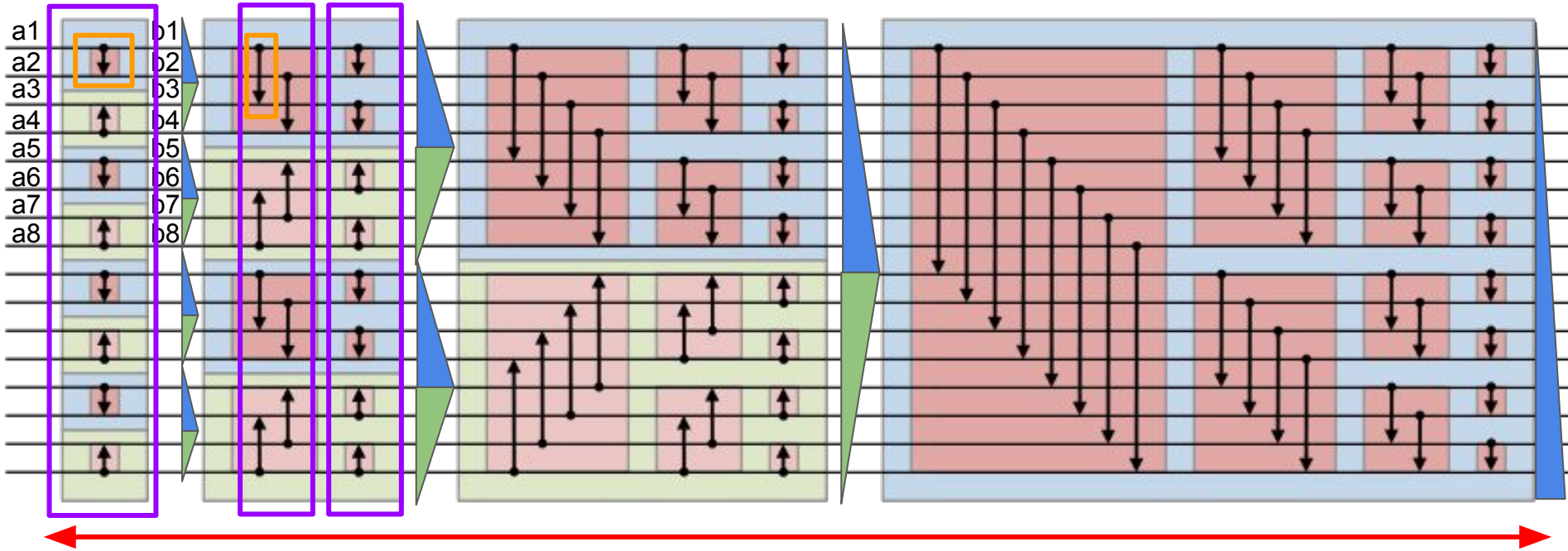
Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Bitonic sort

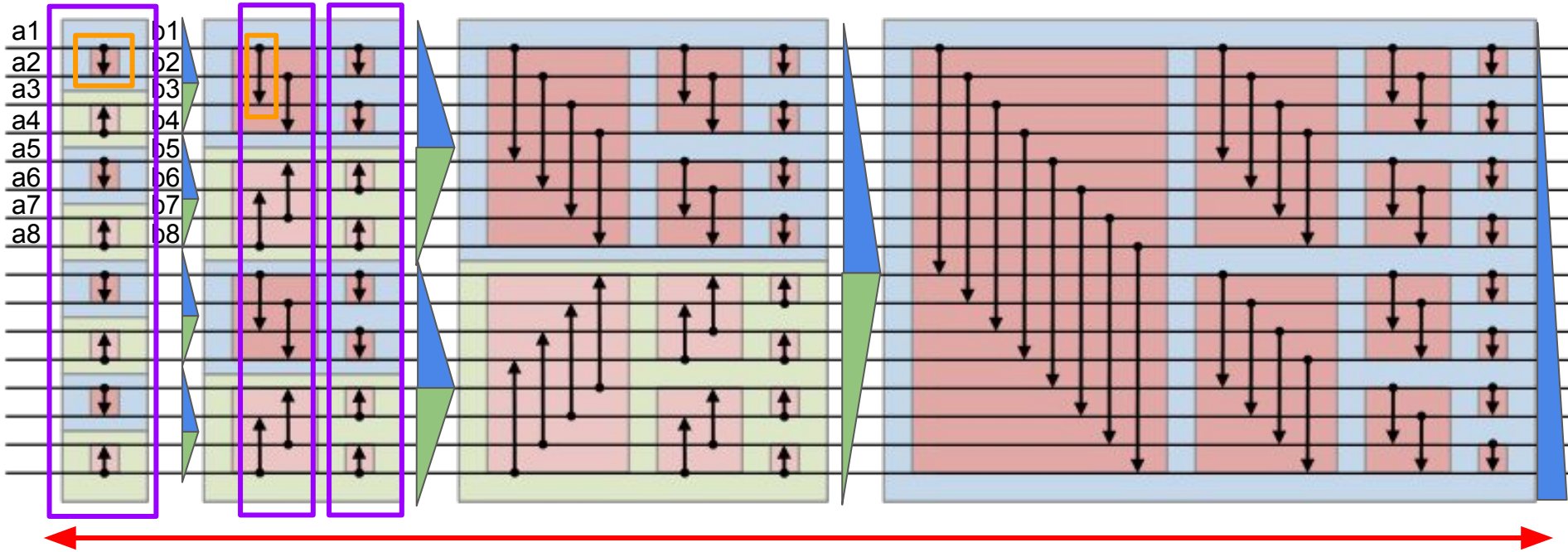
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Сколько раз читаем весь массив?

Bitonic sort

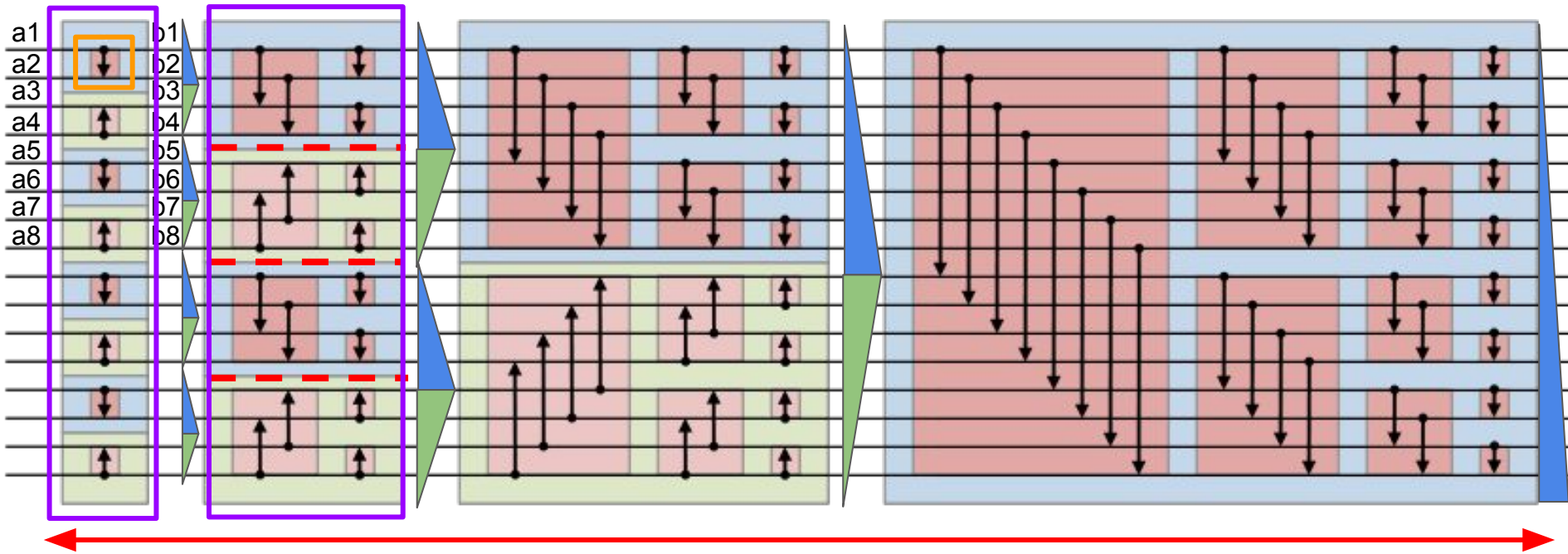
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Сколько раз читаем весь массив? Можно лучше?

Bitonic sort

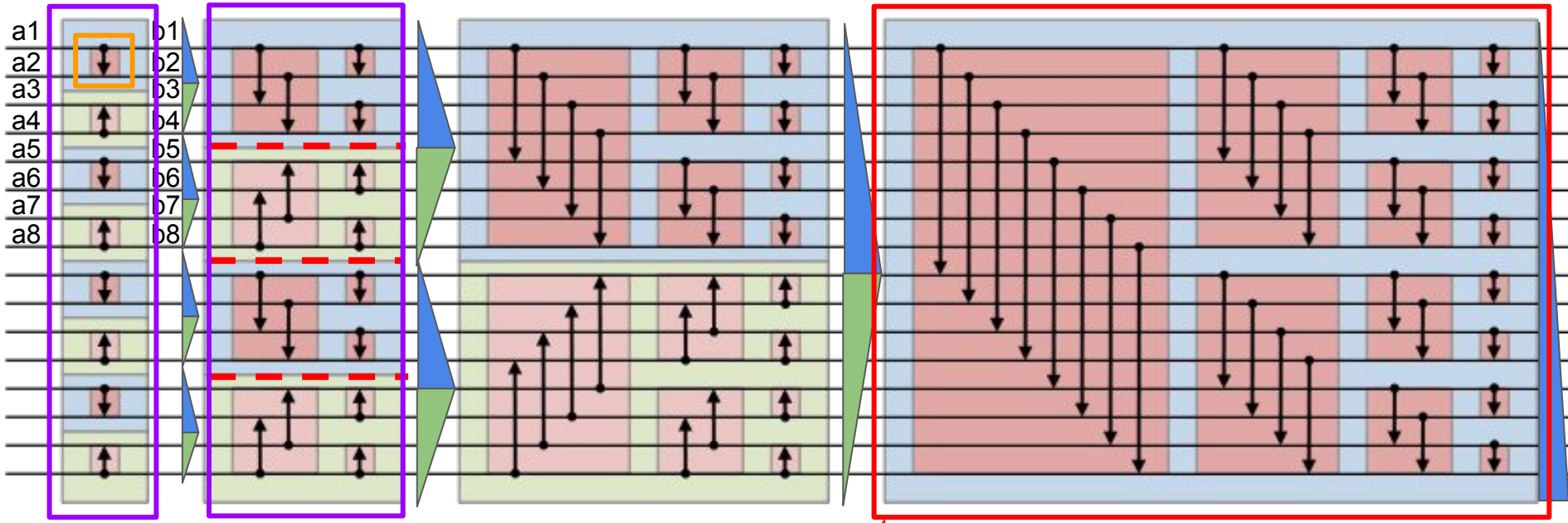
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Сколько раз читаем весь массив? Можно лучше?
Частичная сортировка в локальной памяти!

Bitonic sort

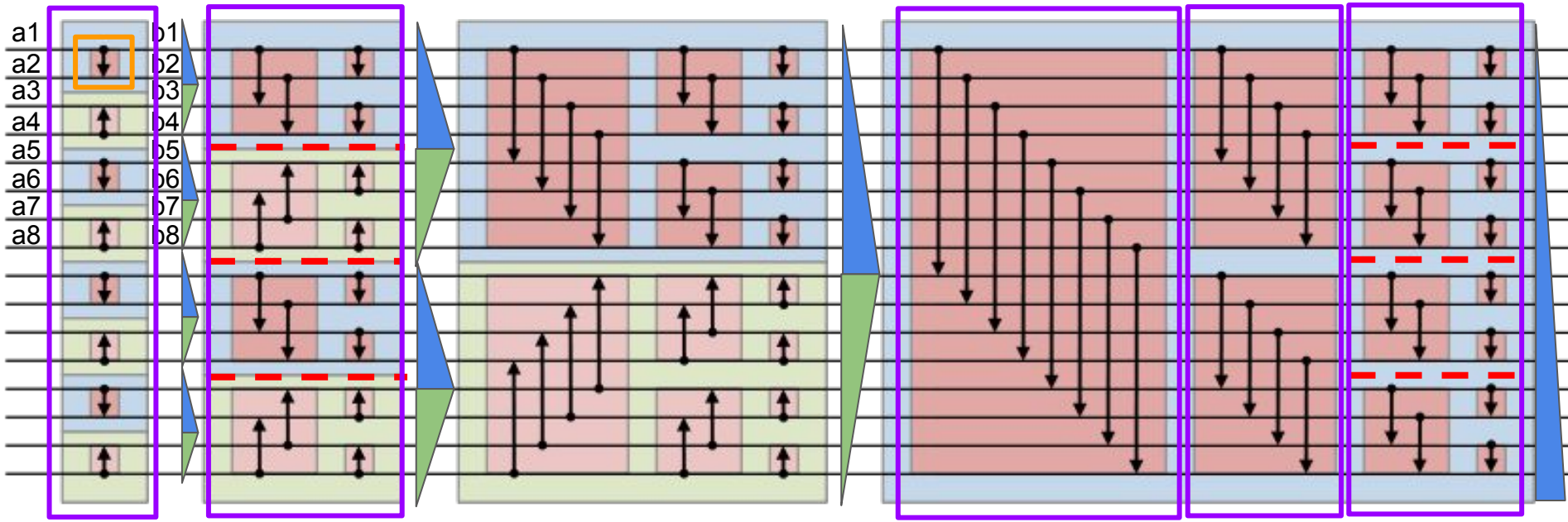
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



В последнем синем блоке влезает ли в локальную память?

Bitonic sort

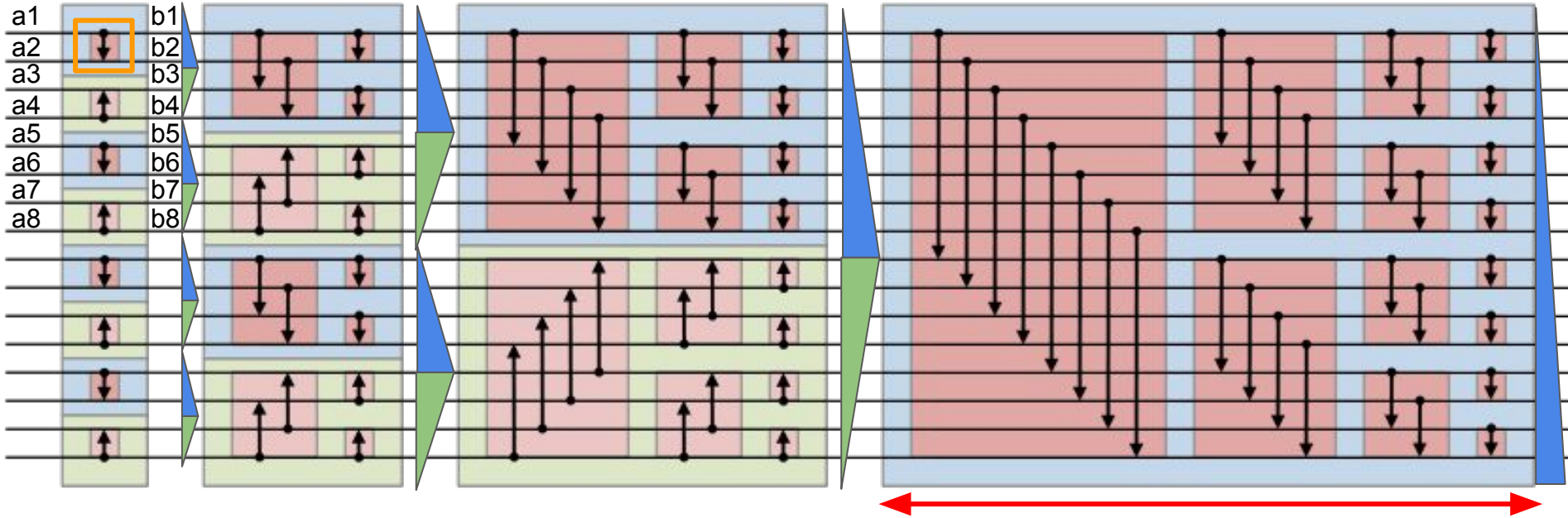
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



В последнем синем блоке влезает ли в локальную память?

Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?

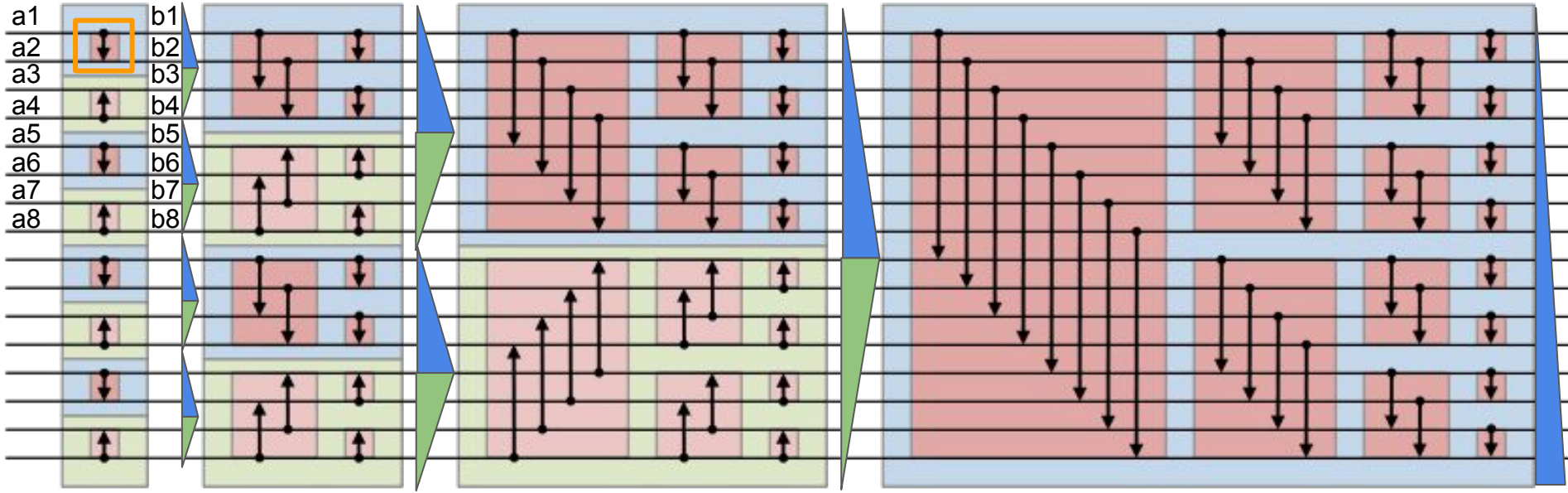


Сколько всего сравнений?

Какое количество красных подблоков?

Bitonic sort

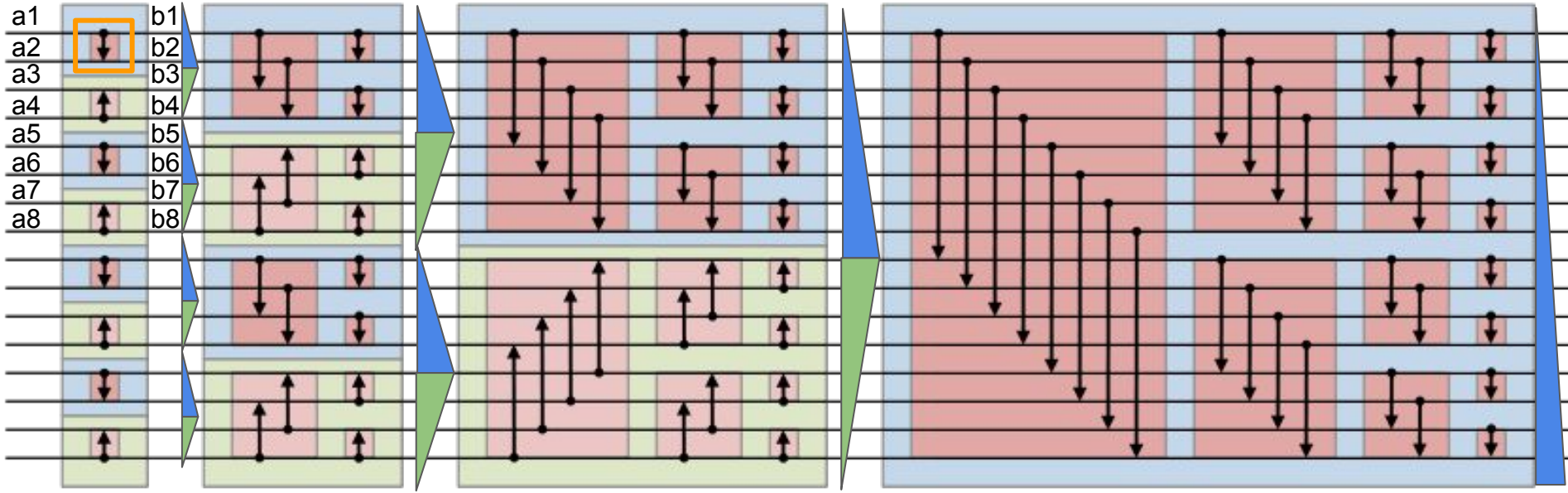
Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Сколько всего сравнений? **O(N * log N * log N)**

Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?

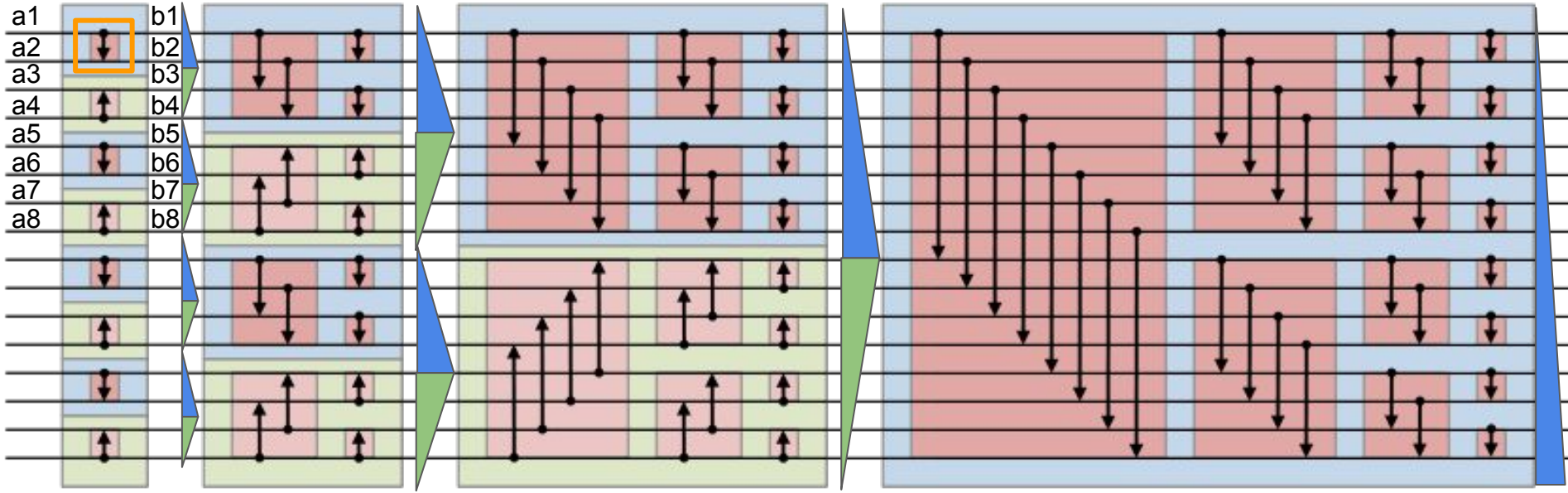


Сколько всего сравнений? **O(N * log N * log N)**

Какая итого скорость? **O(N * log N * log N)???**

Bitonic sort

Как это реализовать в модели массового параллелизма?
Что делает **один WorkItem**? Что делает **один кернел**?



Сколько всего сравнений? **O(N * log N * log N)**

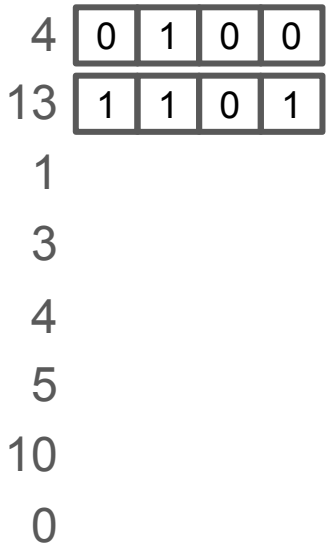
Какая итого скорость? **O(N * log N * log N / K)**

Radix sort (поразрядная сортировка)

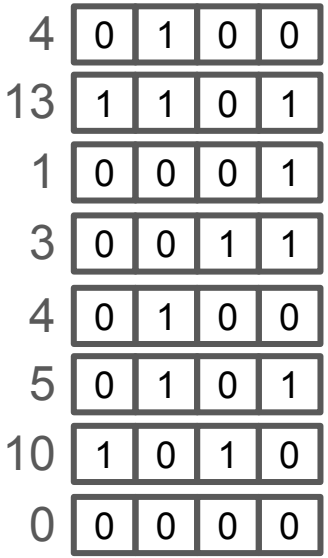
4
13
1
3
4
5
10
0

Radix sort (поразрядная сортировка)

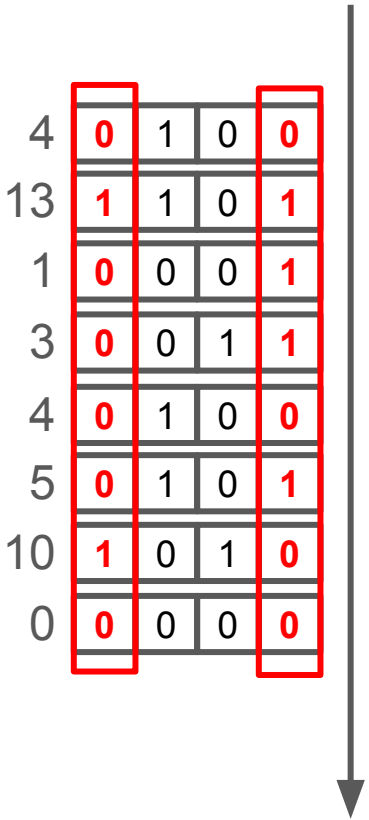
Radix sort (поразрядная сортировка)



Radix sort (поразрядная сортировка)

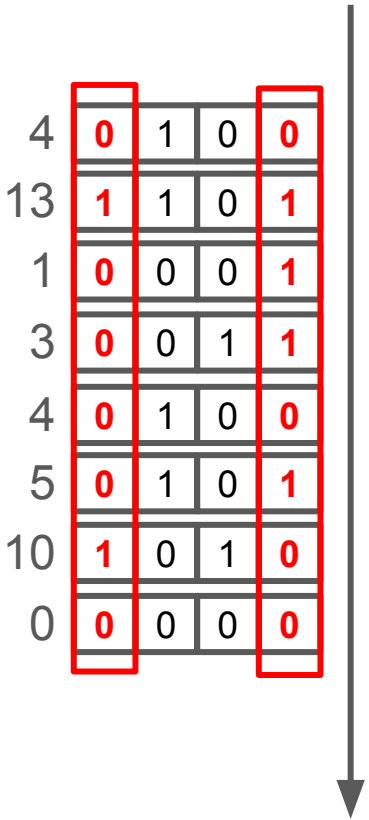


Radix sort (поразрядная сортировка)

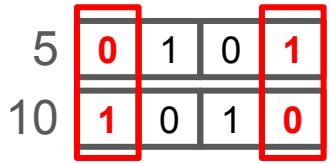


По какому разряду сортировать сначала?

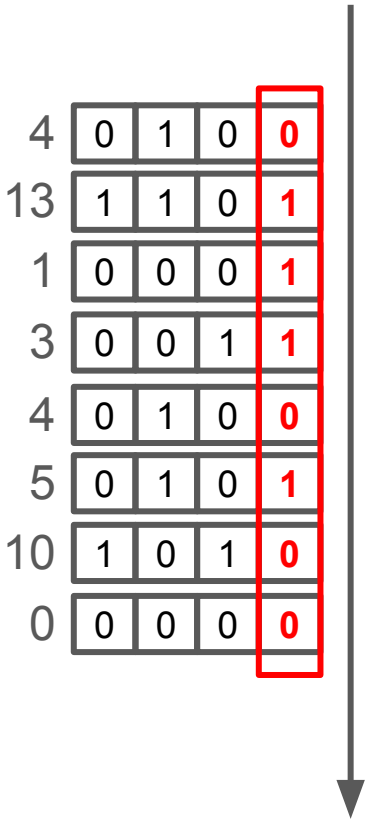
Radix sort (поразрядная сортировка)



По какому разряду сортировать сначала?
По какому разряду сортировать в конце?

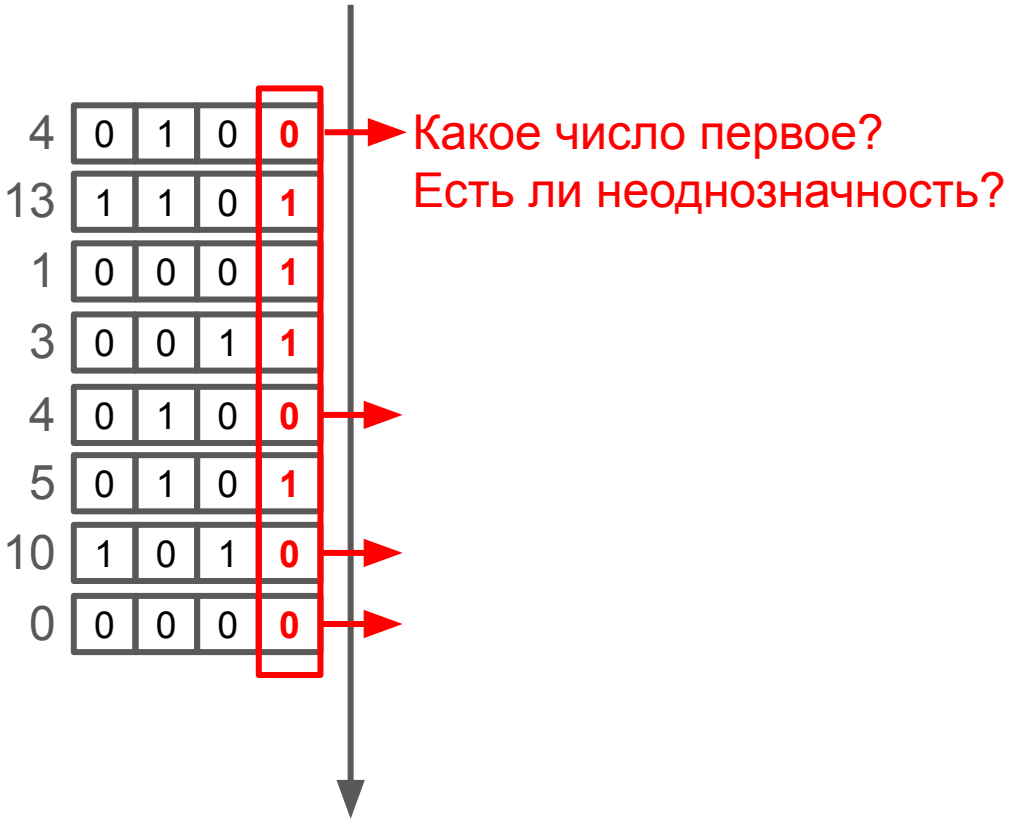


Radix sort (поразрядная сортировка)

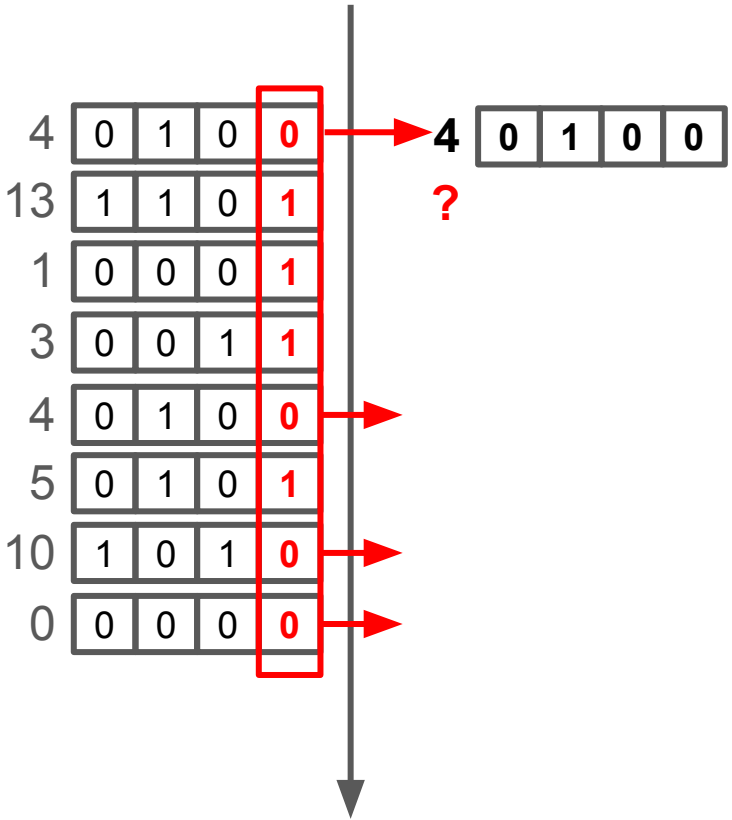


Какое число первое?
Есть ли неоднозначность?

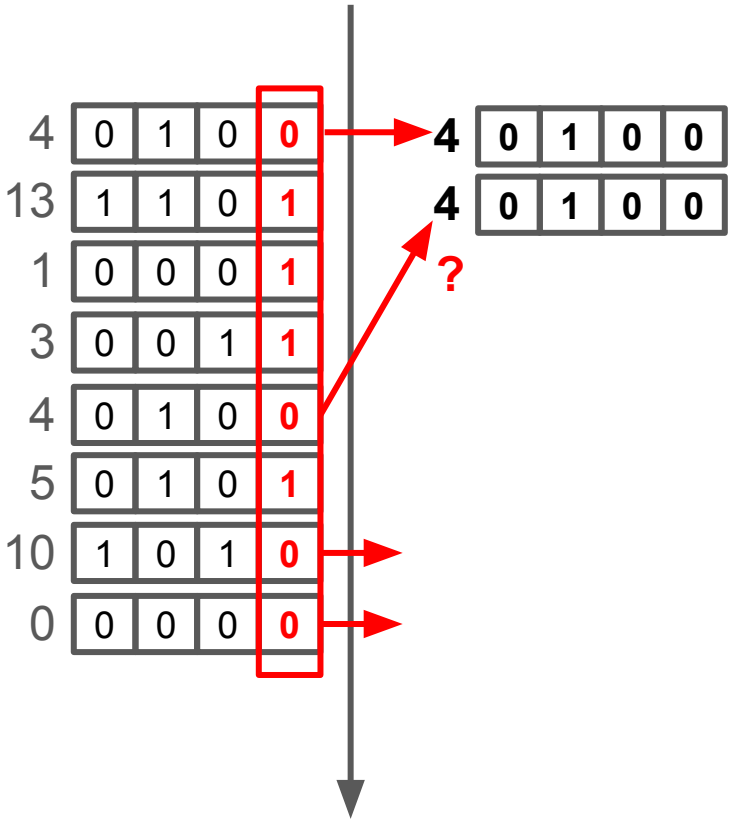
Radix sort (**стабильная** поразрядная сортировка)



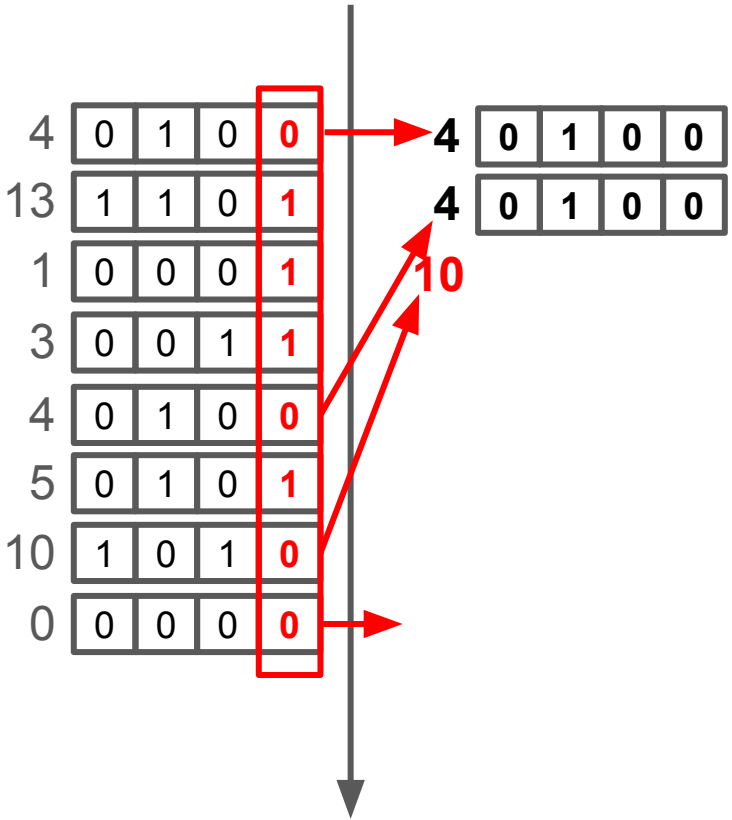
Radix sort (**стабильная** поразрядная сортировка)



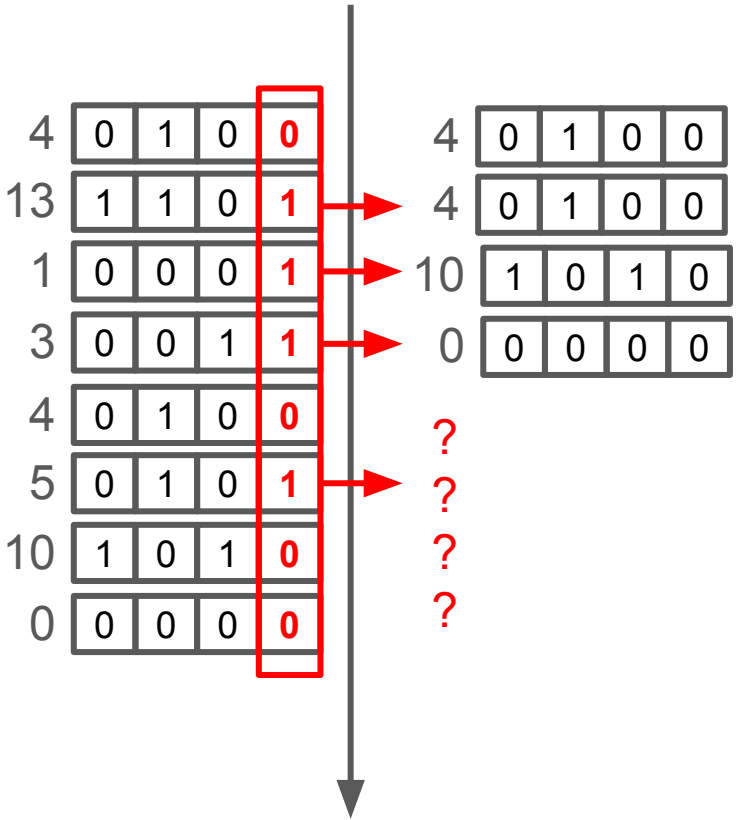
Radix sort (**стабильная** поразрядная сортировка)



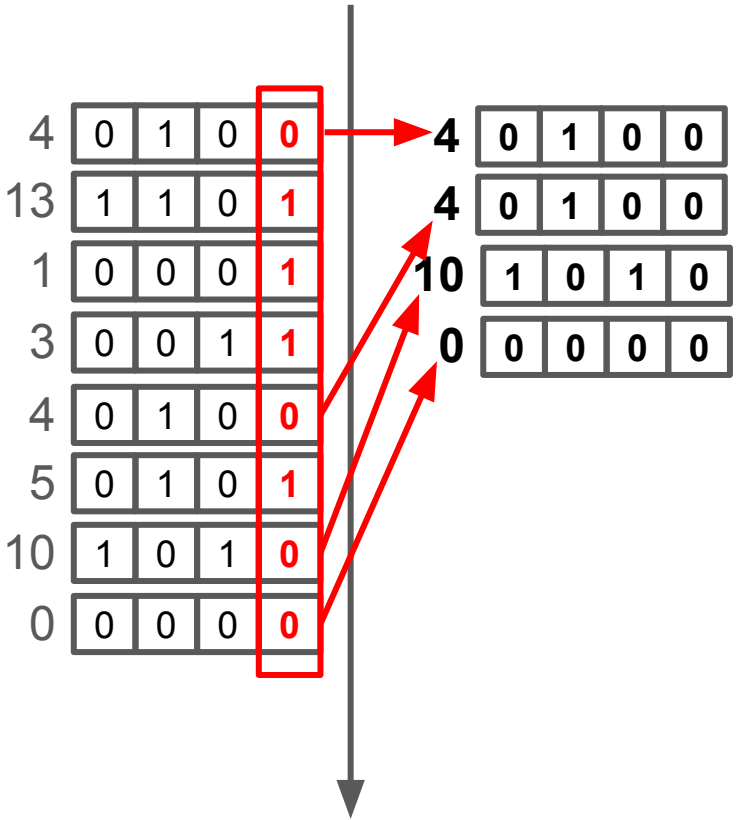
Radix sort (**стабильная** поразрядная сортировка)



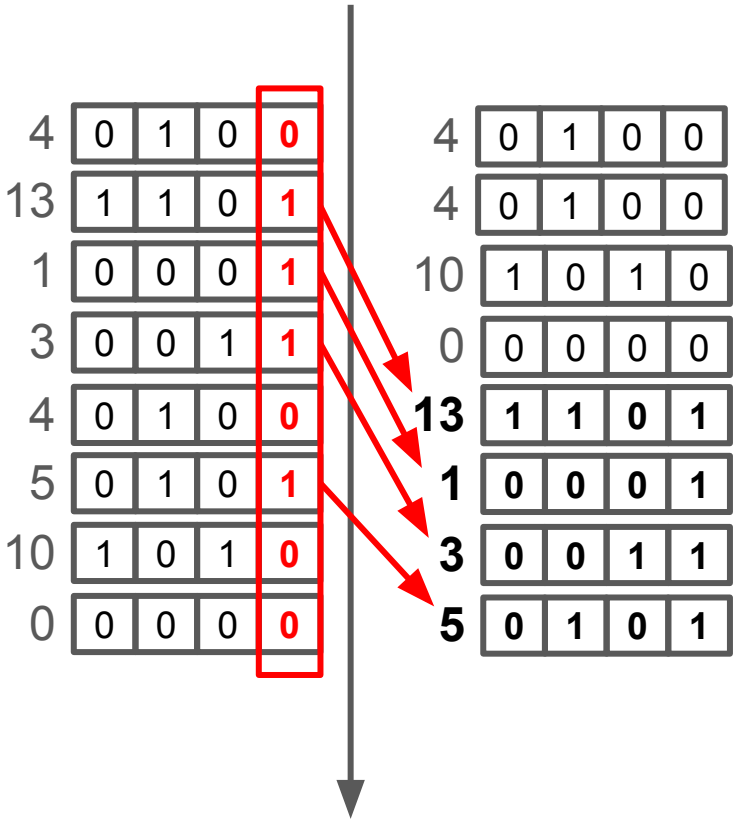
Radix sort (**стабильная** поразрядная сортировка)



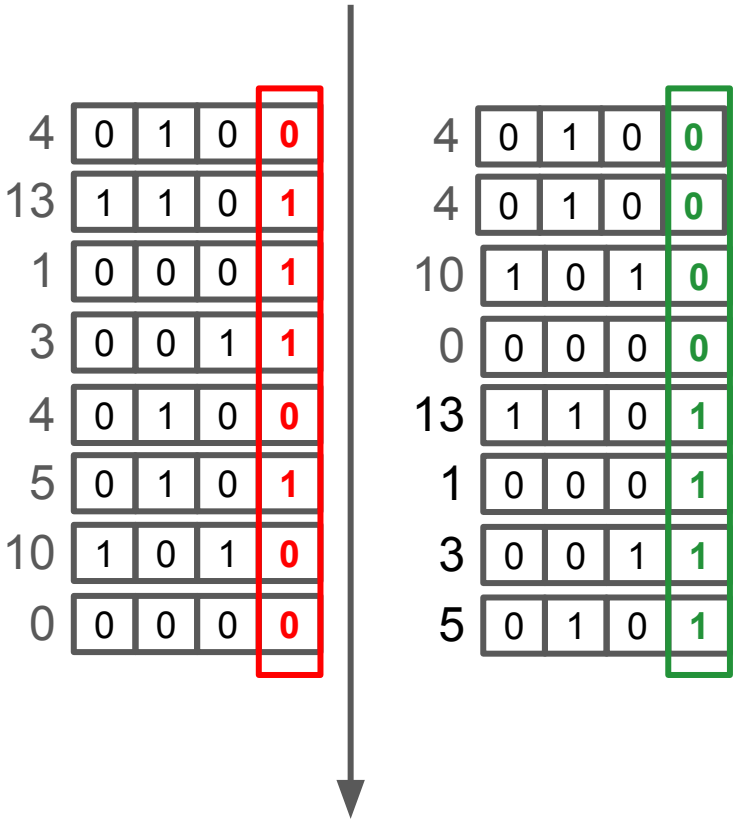
Radix sort (**стабильная** поразрядная сортировка)



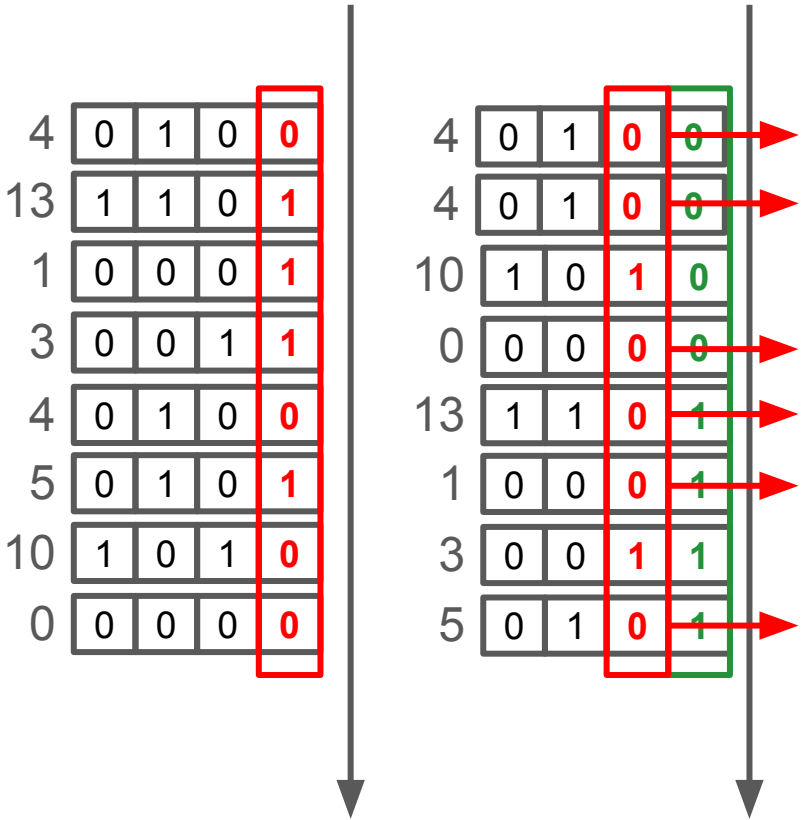
Radix sort (**стабильная** поразрядная сортировка)



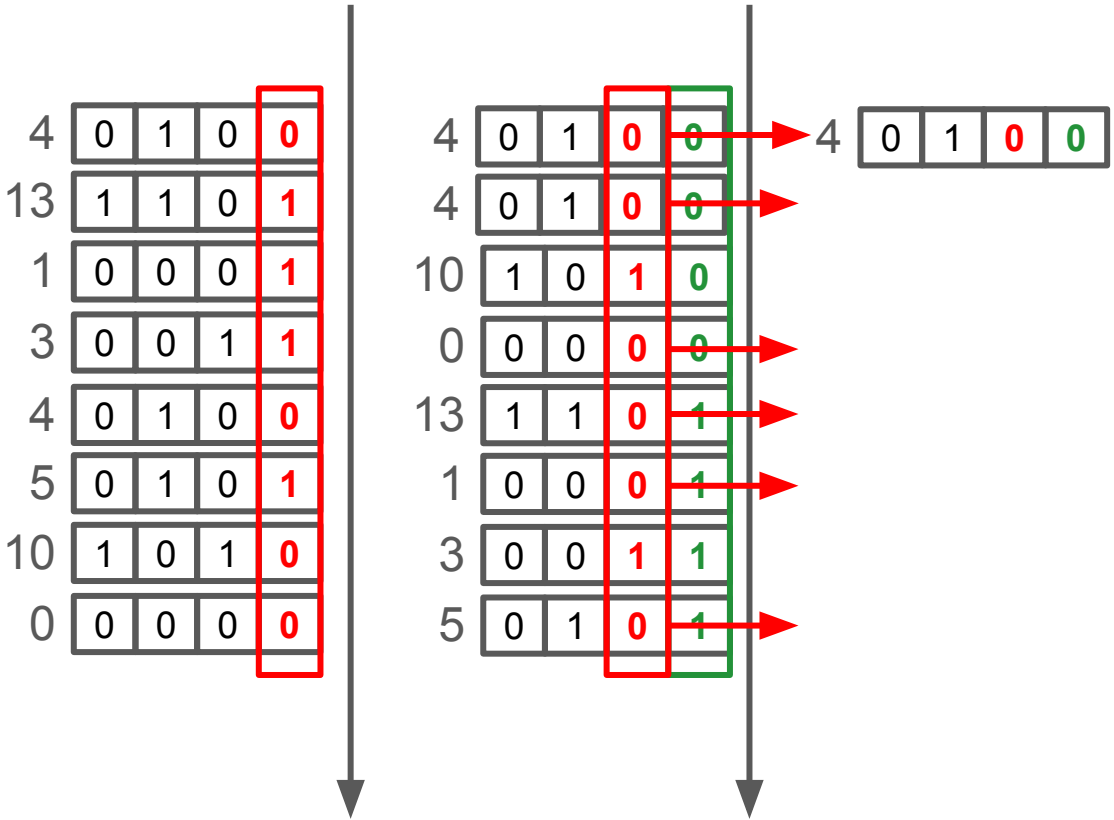
Radix sort (**стабильная** поразрядная сортировка)



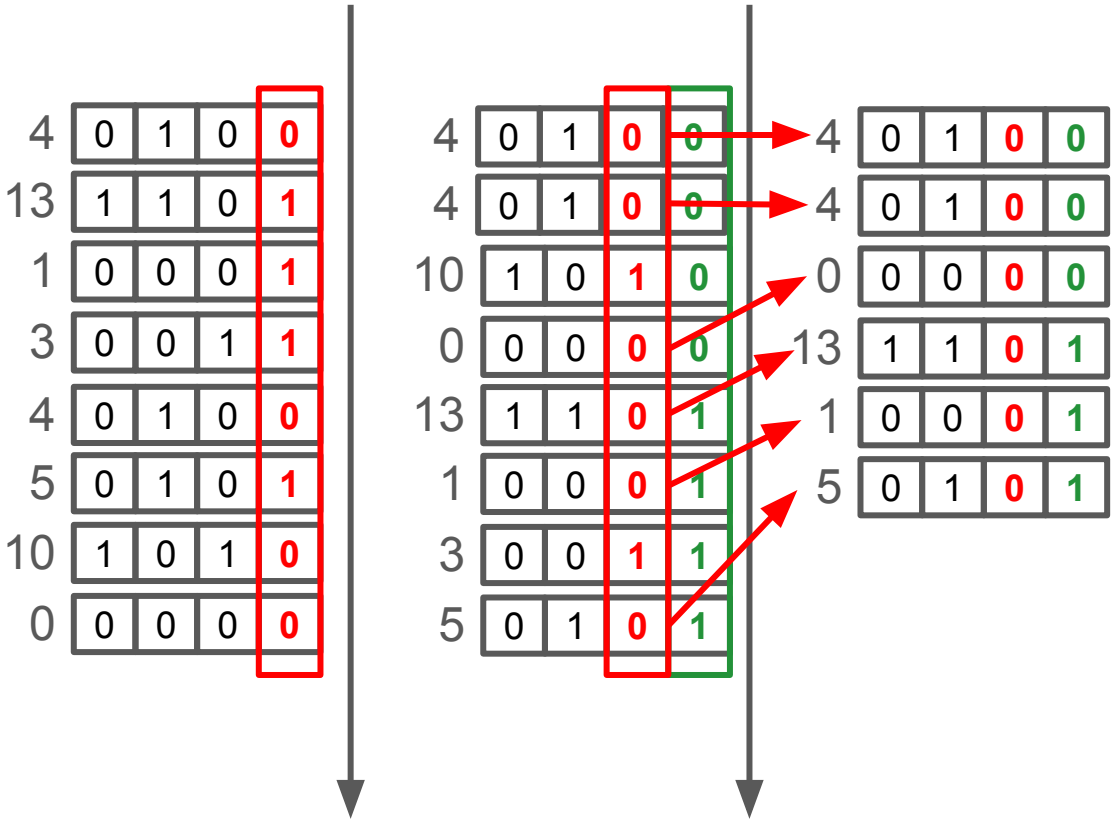
Radix sort (**стабильная** поразрядная сортировка)



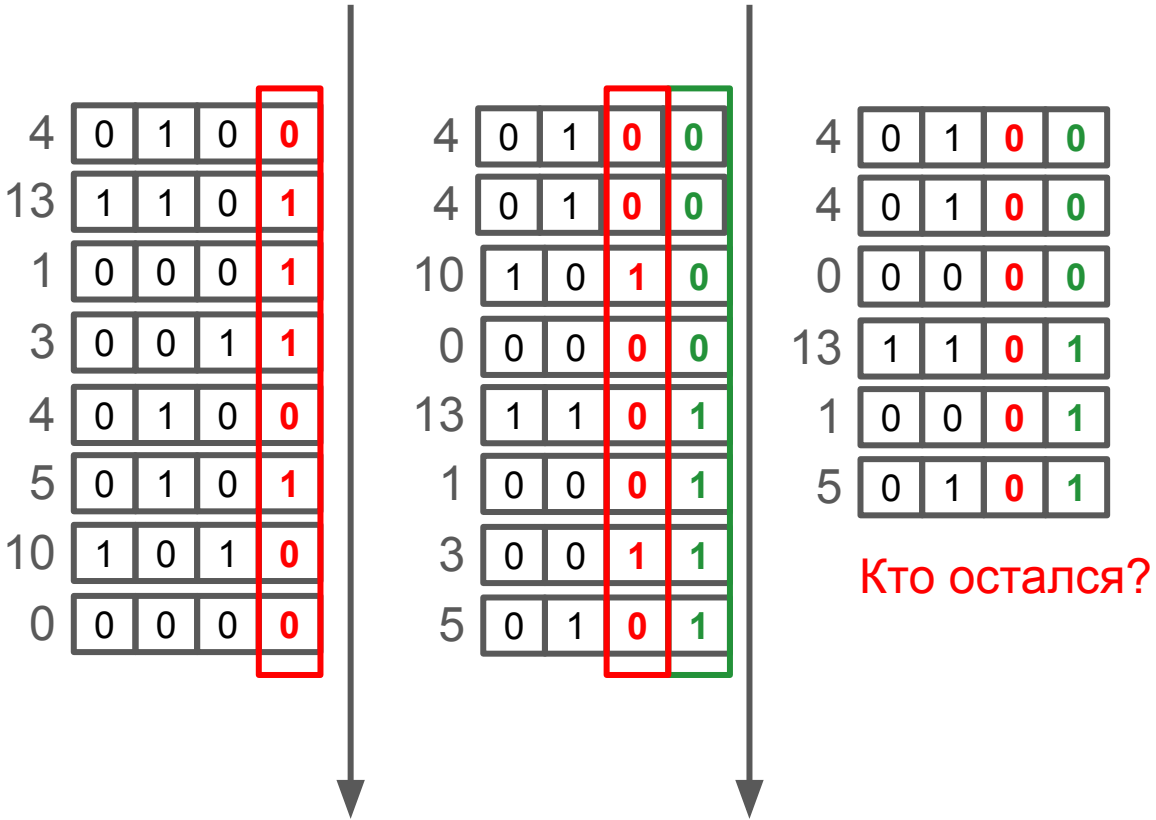
Radix sort (**стабильная** поразрядная сортировка)



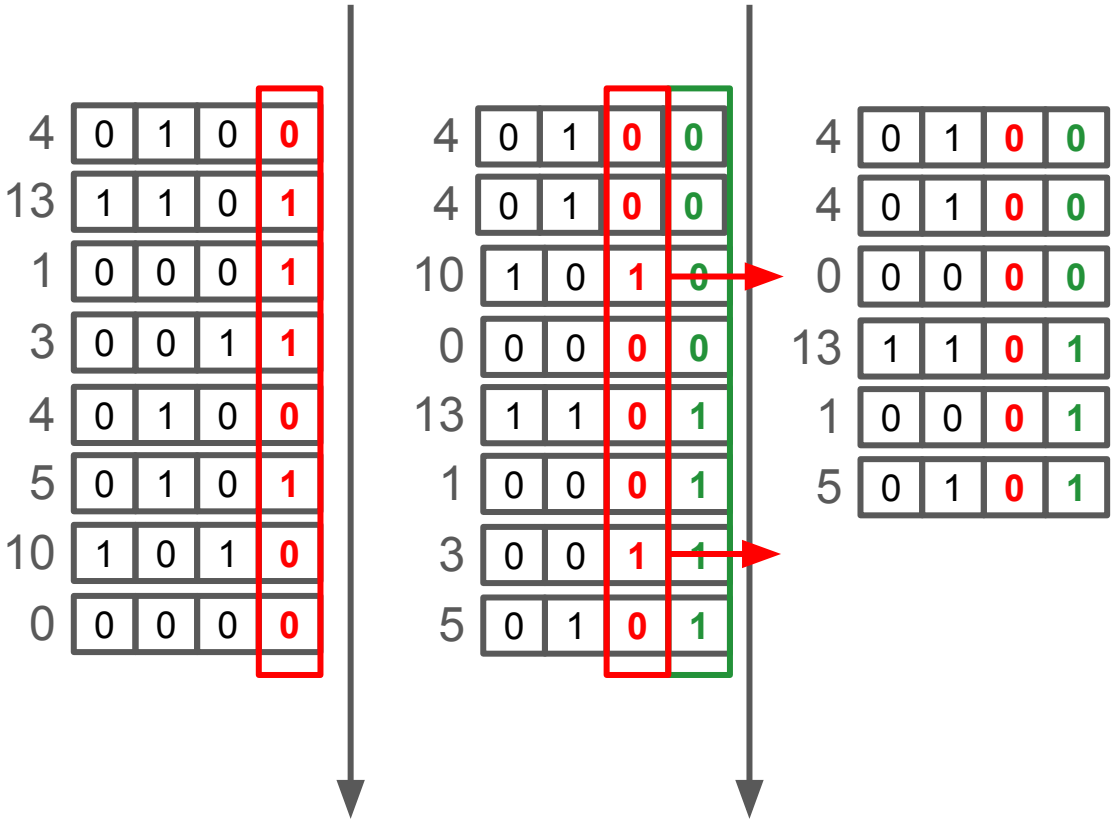
Radix sort (**стабильная** поразрядная сортировка)



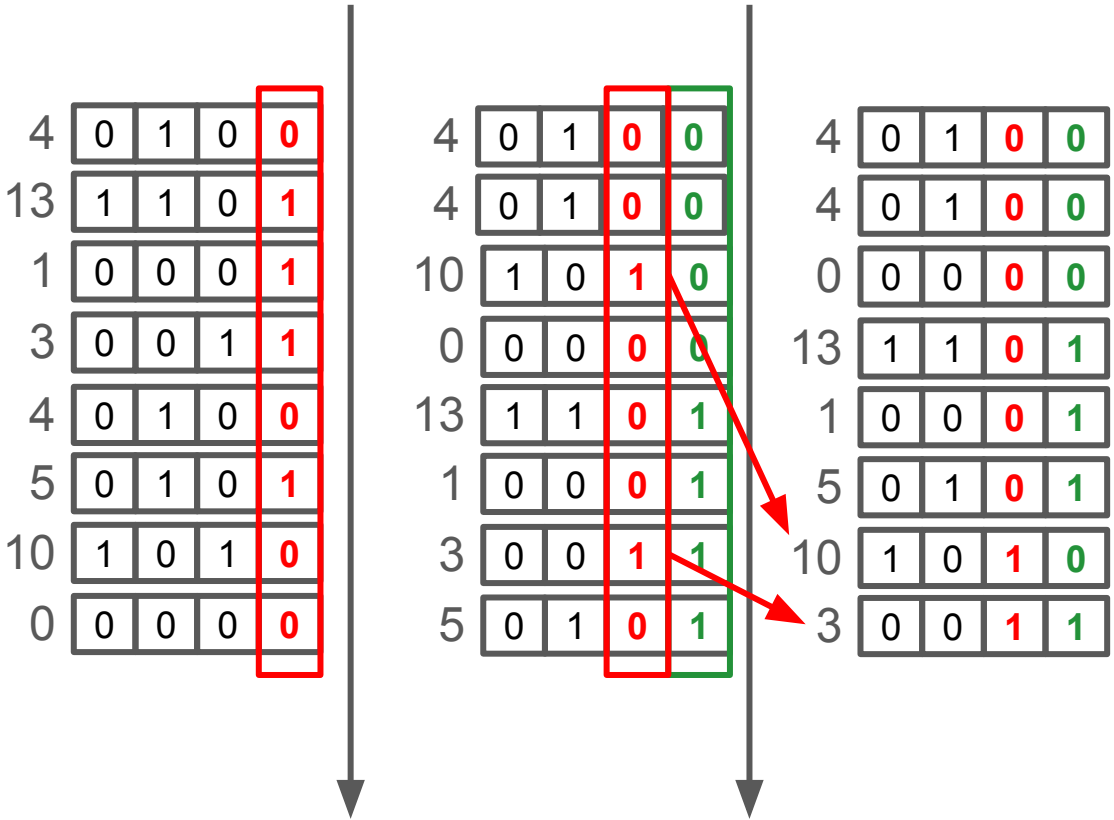
Radix sort (**стабильная** поразрядная сортировка)



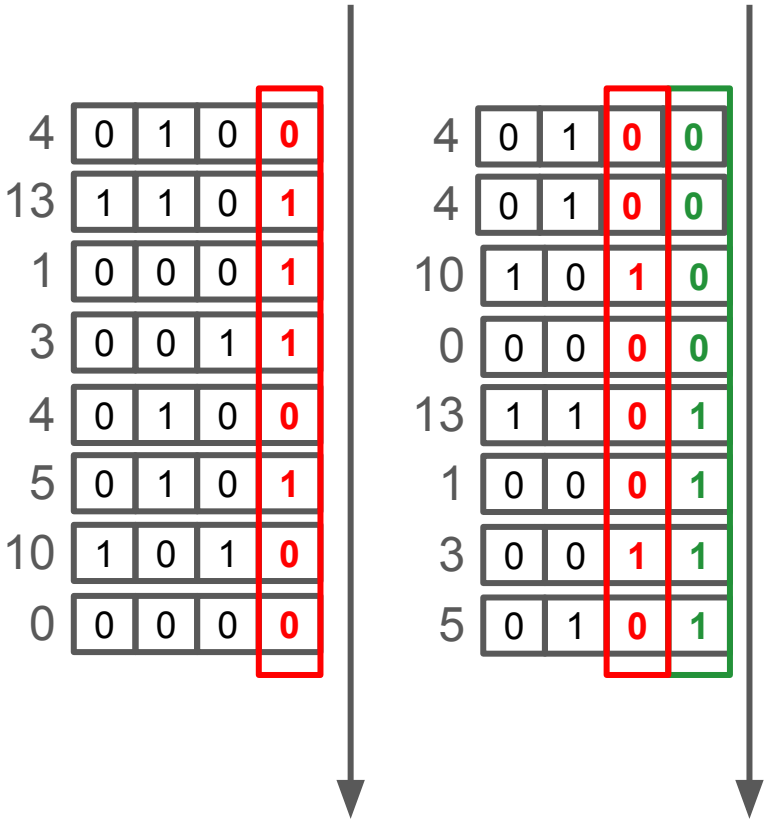
Radix sort (**стабильная** поразрядная сортировка)



Radix sort (**стабильная** поразрядная сортировка)

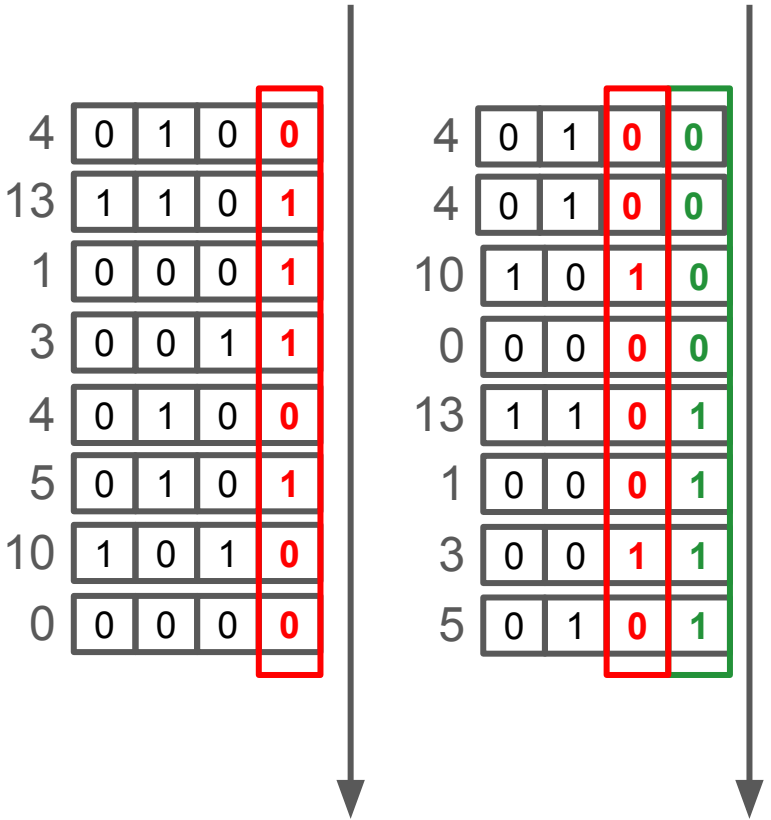


Radix sort (**стабильная** поразрядная сортировка)



Упорядочены по второму биту

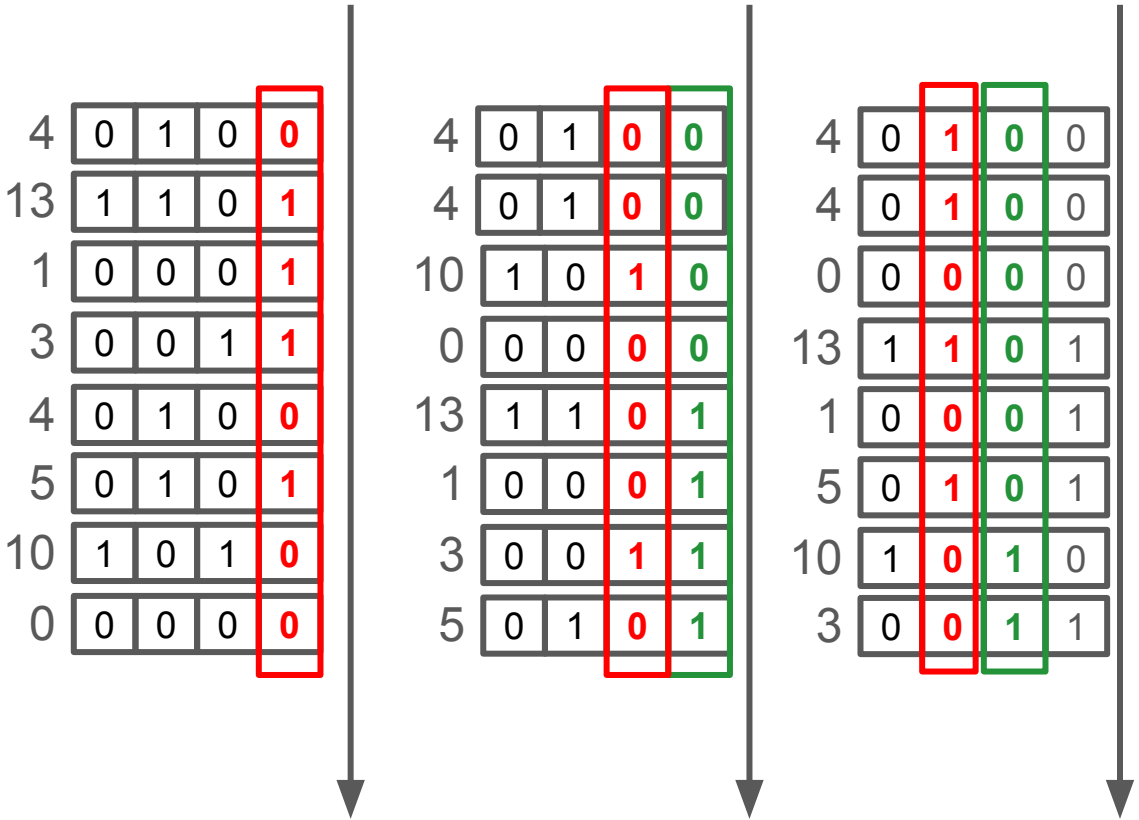
Radix sort (**стабильная** поразрядная сортировка)



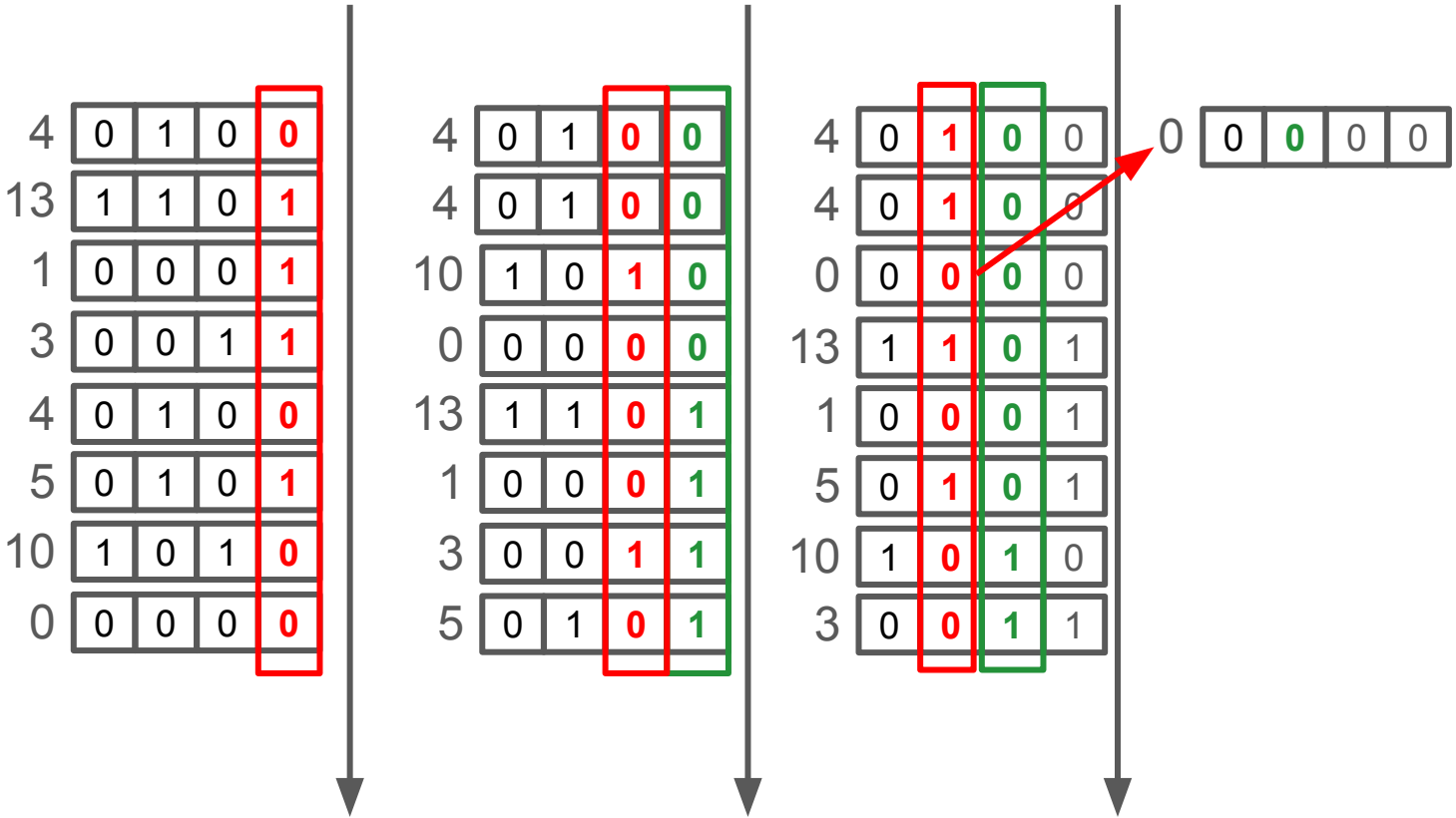
Упорядочены по второму биту

Но мы ведь потеряли упорядоченность по первому биту!

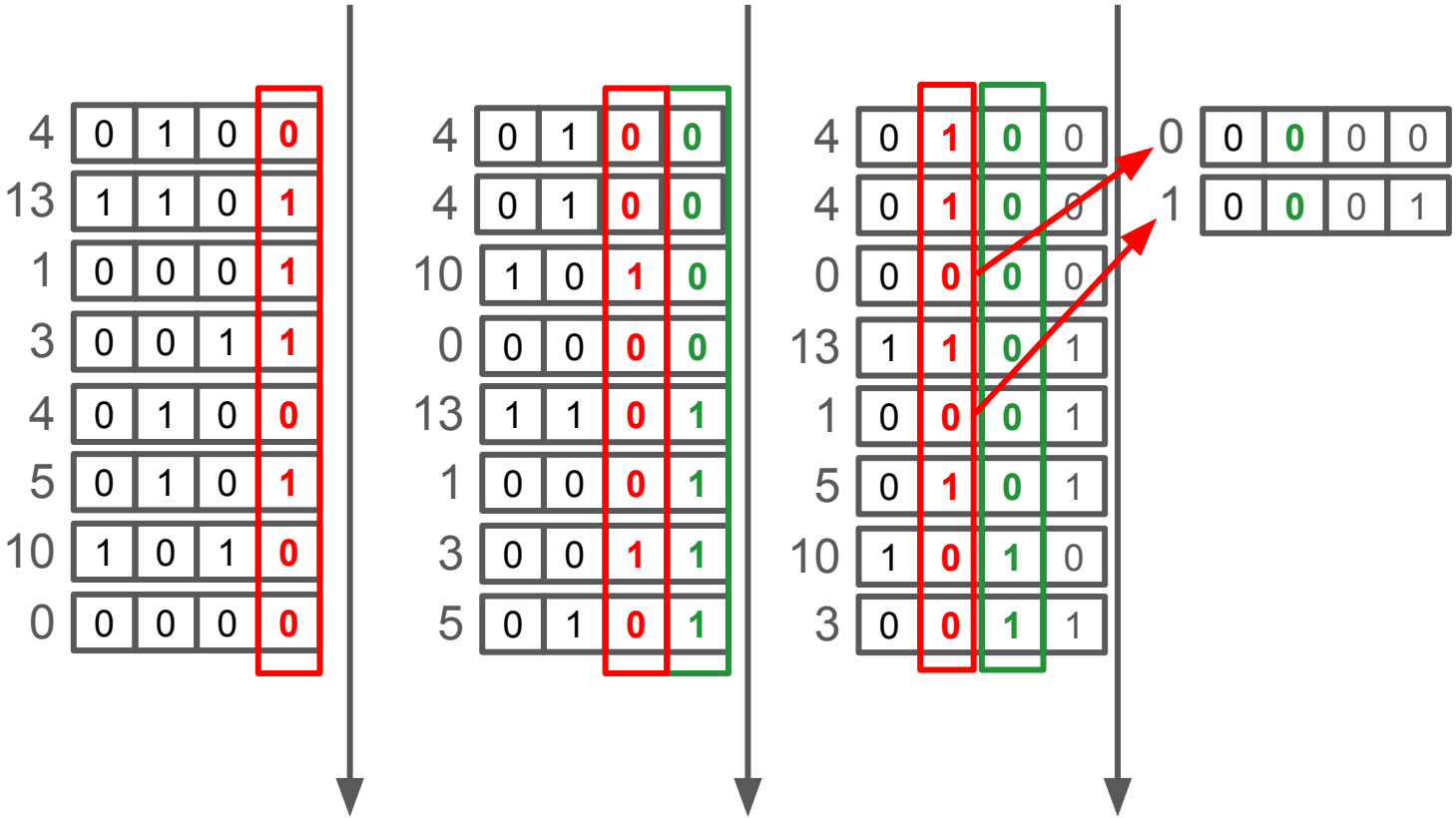
Radix sort (**стабильная** поразрядная сортировка)



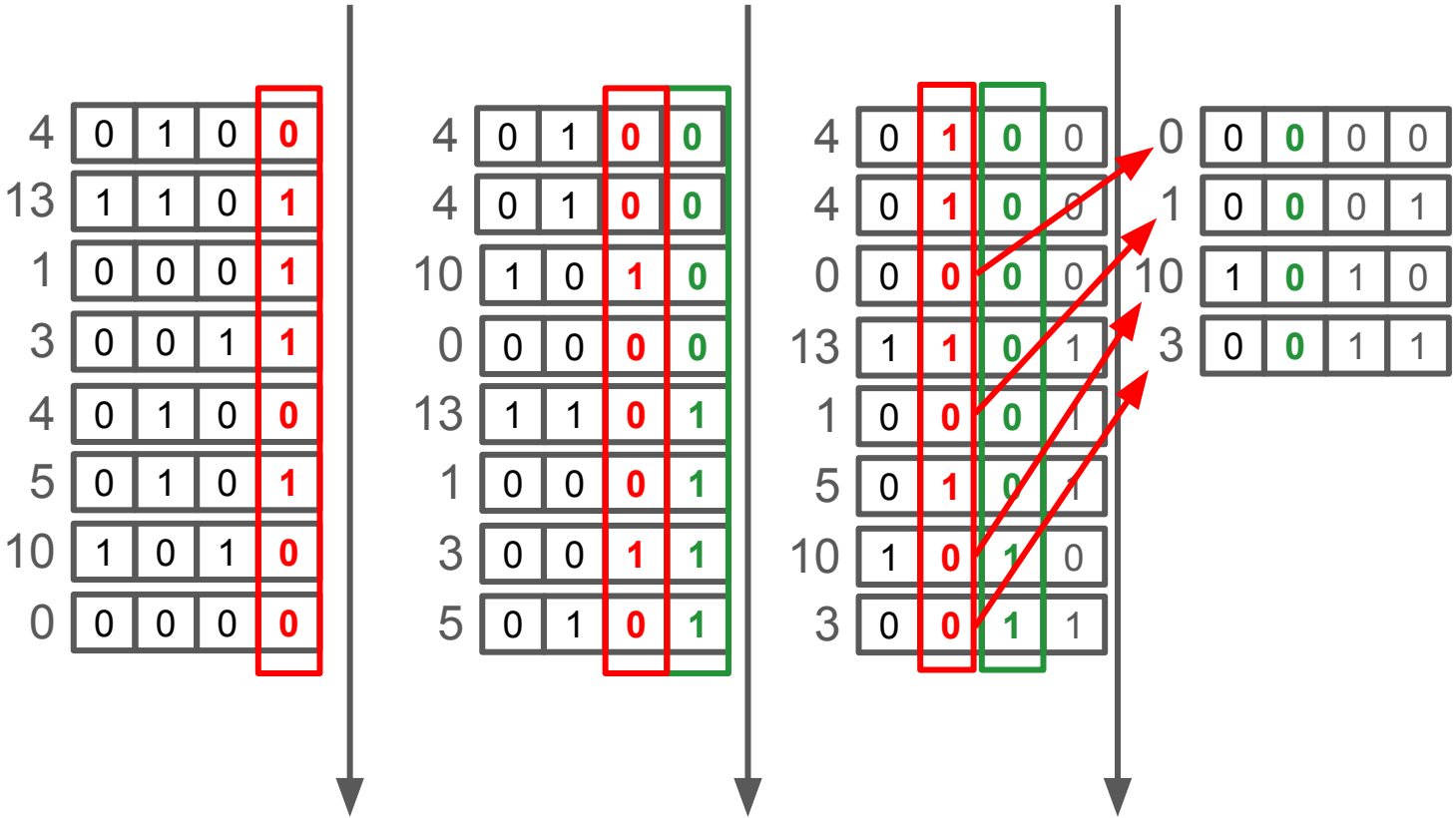
Radix sort (**стабильная** поразрядная сортировка)



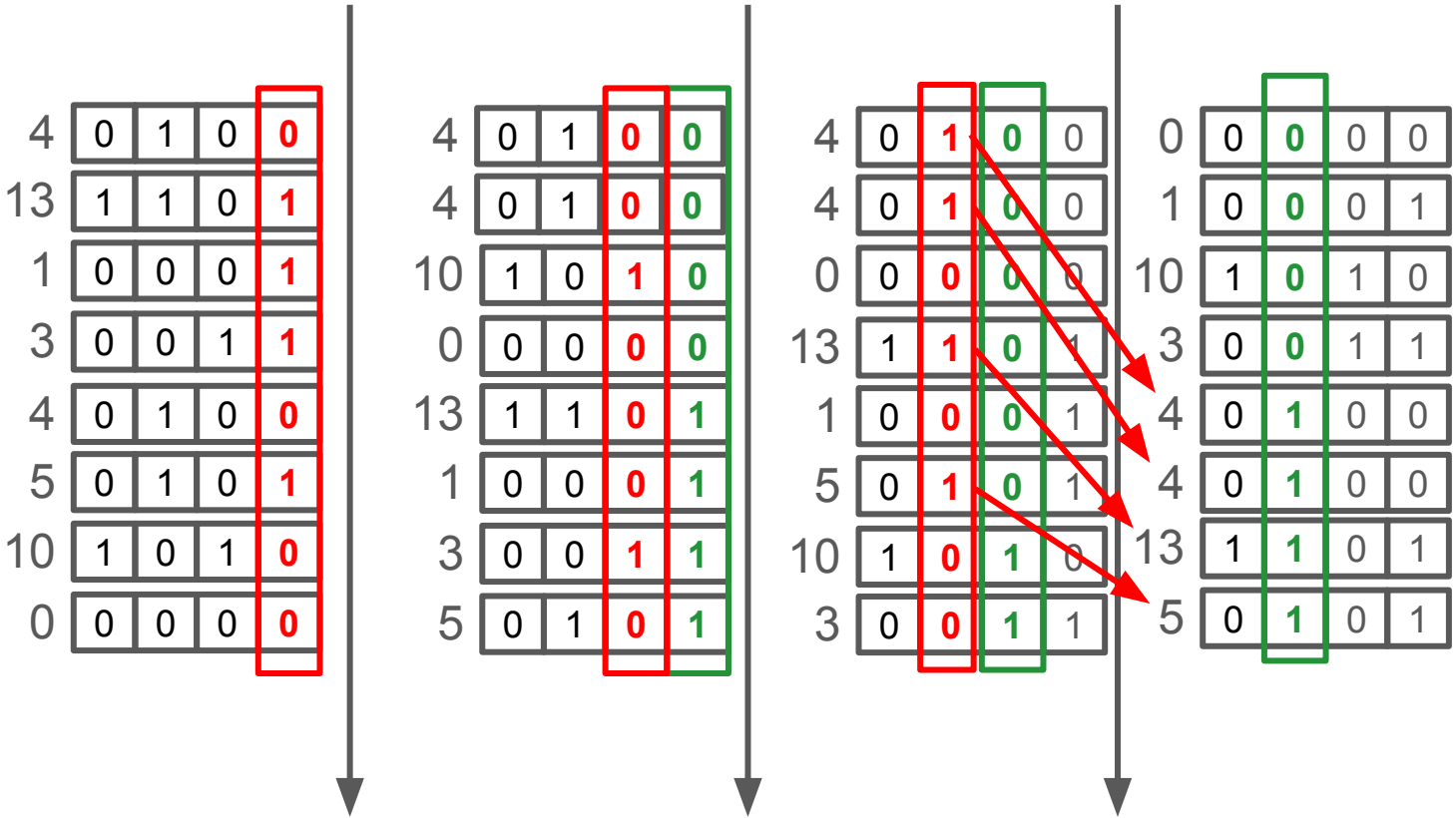
Radix sort (**стабильная** поразрядная сортировка)



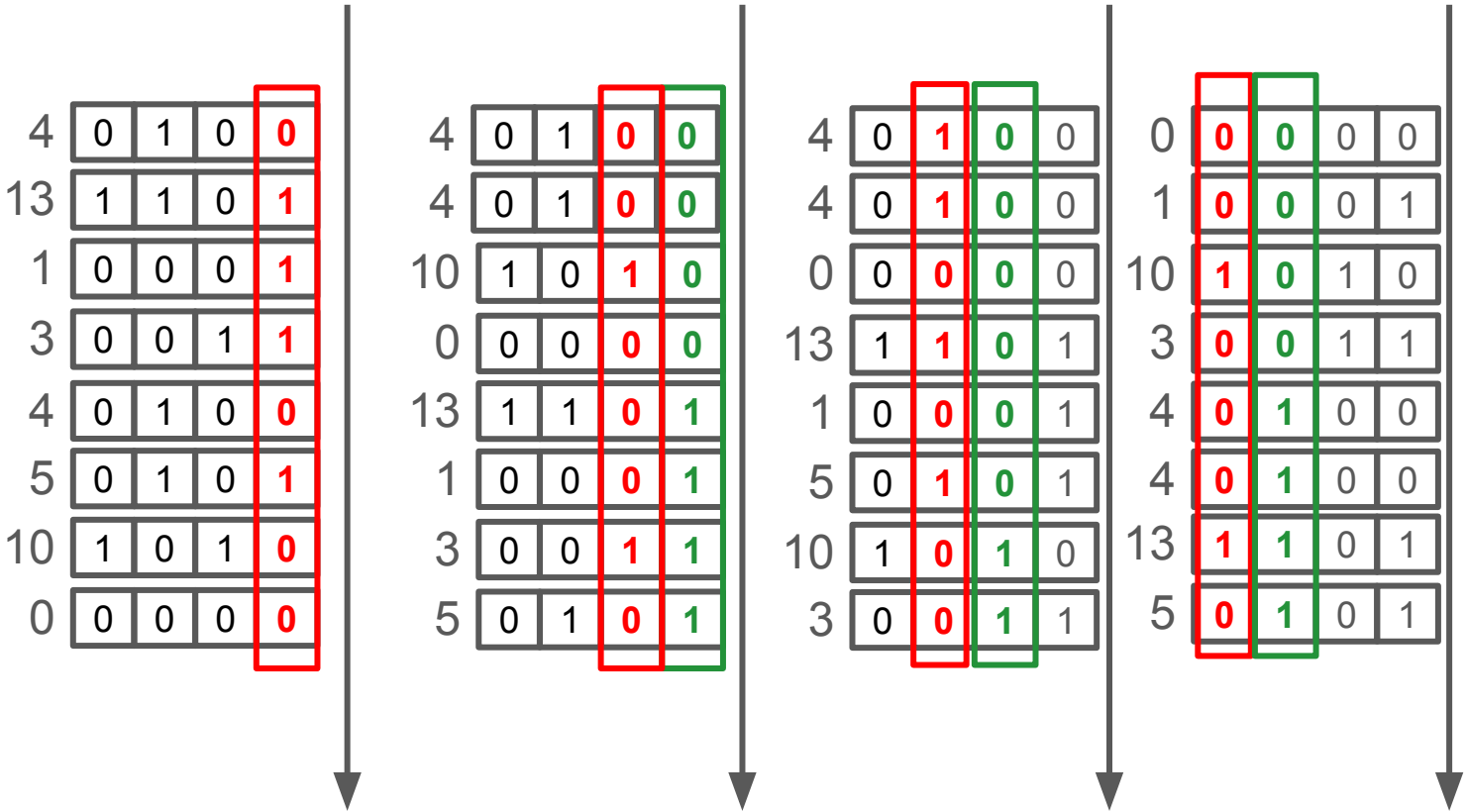
Radix sort (**стабильная** поразрядная сортировка)



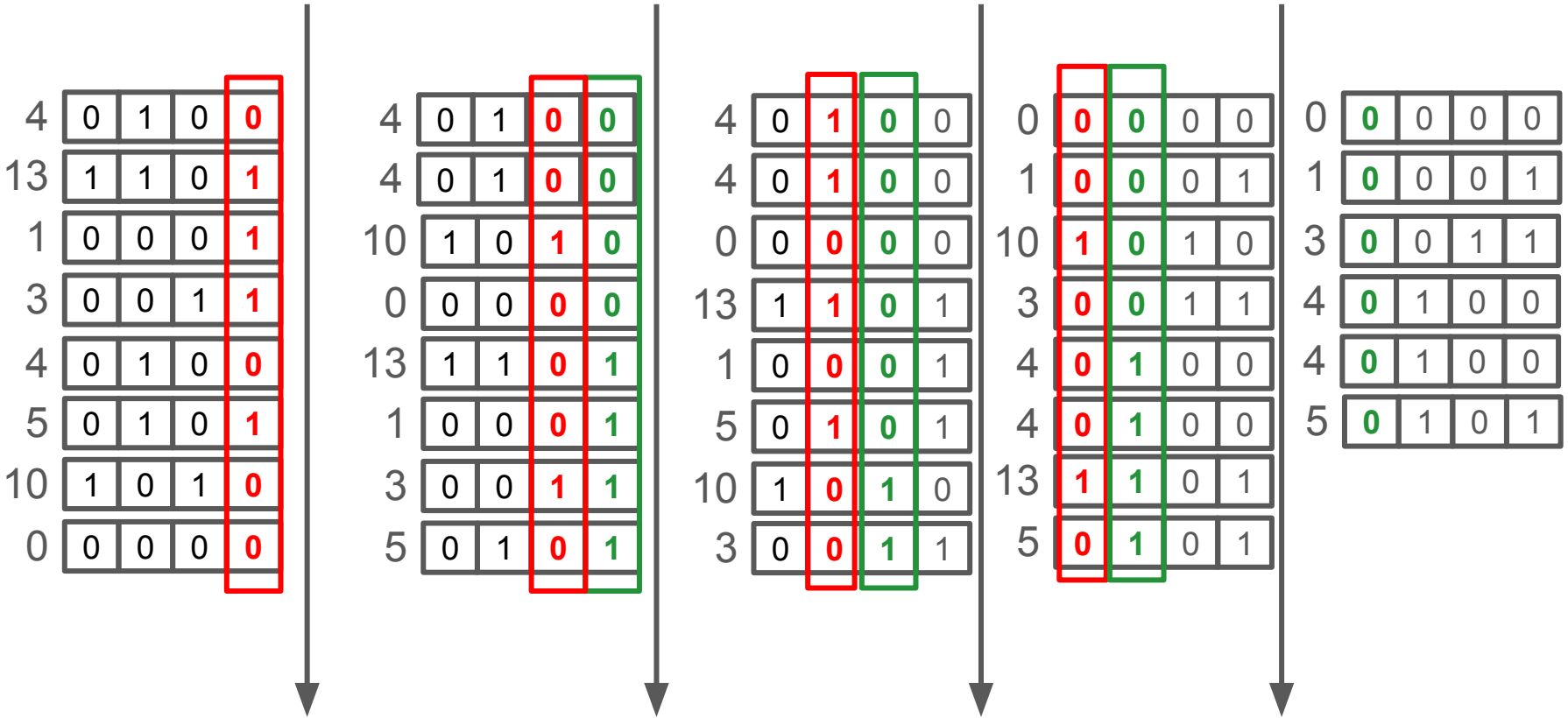
Radix sort (**стабильная** поразрядная сортировка)



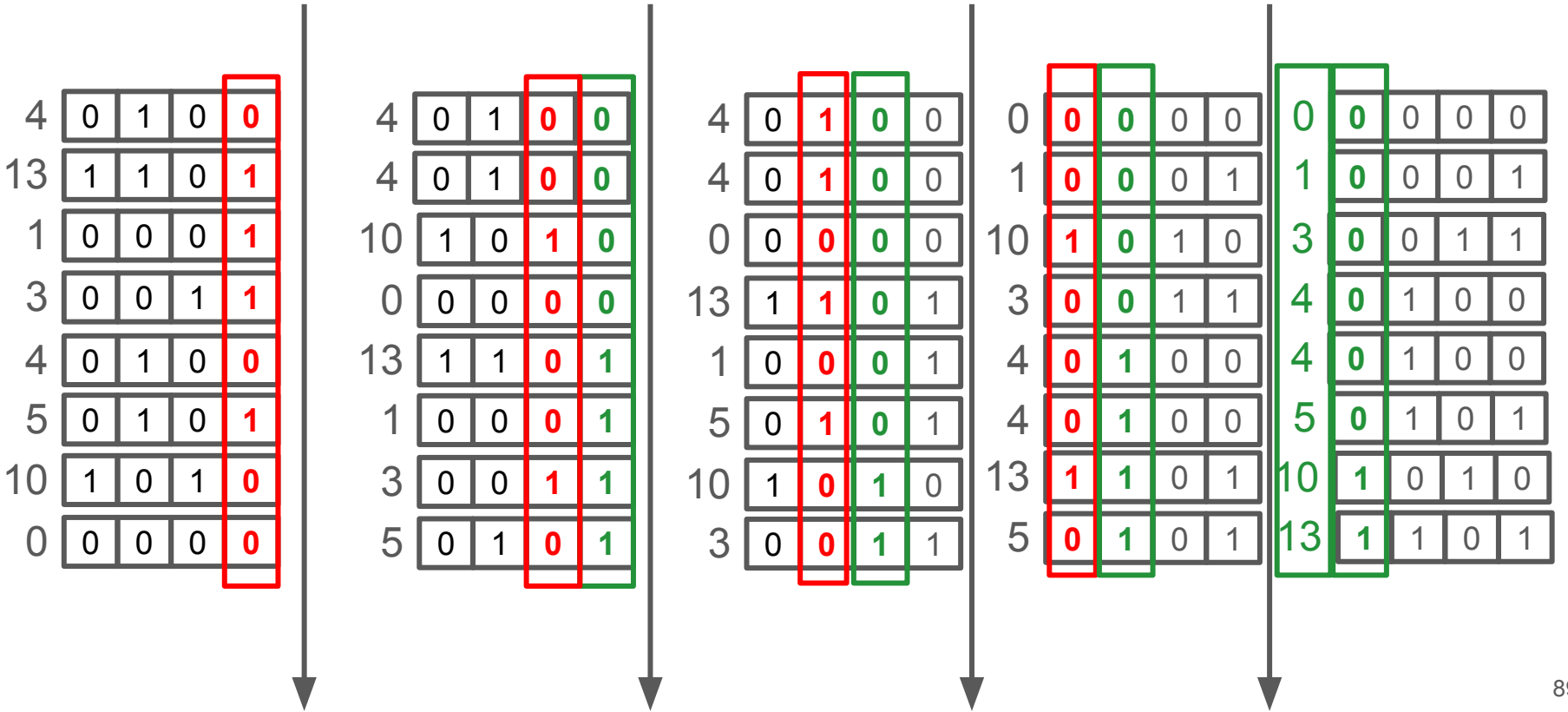
Radix sort (**стабильная** поразрядная сортировка)



Radix sort (**стабильная** поразрядная сортировка)



Radix sort (**стабильная** поразрядная сортировка)



Префиксные суммы: напоминание

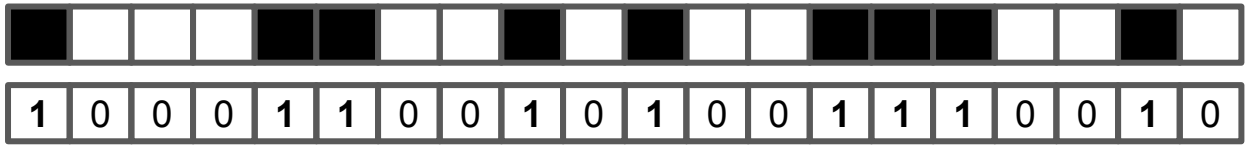
- Scan
- Что если каждый WorkItem детектирует ключевую точку в пикселе?
  - кто-то мог решить что пиксель плохой (без текстуры)
  - кто-то другой мог решить что тут есть ключевая точка - по какому индексу в выходному массиву ее записать?

Префиксные суммы: напоминание

- Scan
- Что если каждый WorkItem детектирует ключевую точку в пикселе?
  - кто-то мог решить что пиксель плохой (без текстуры)
  - кто-то другой мог решить что тут есть ключевая точка - по какому индексу в выходному массиву ее записать?
  - а чем плохо просто сохранять результат с пузырями (“здесь нет точки”)?

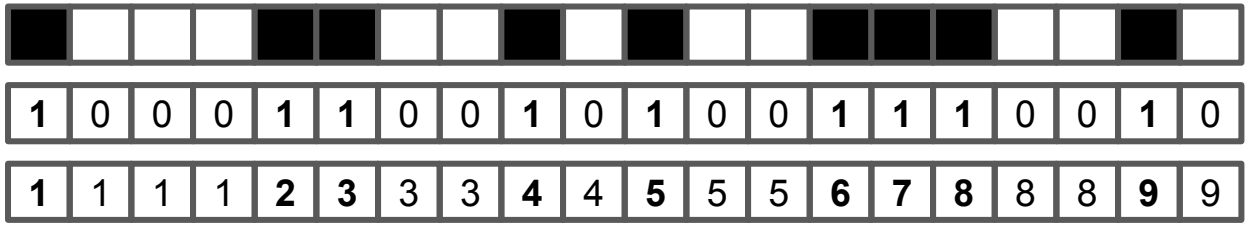
Префиксные суммы: напоминание

- Scan
- Генерация результата в три прохода:
  - **map** - каждый WorkItem записал 1 если результат есть, иначе 0



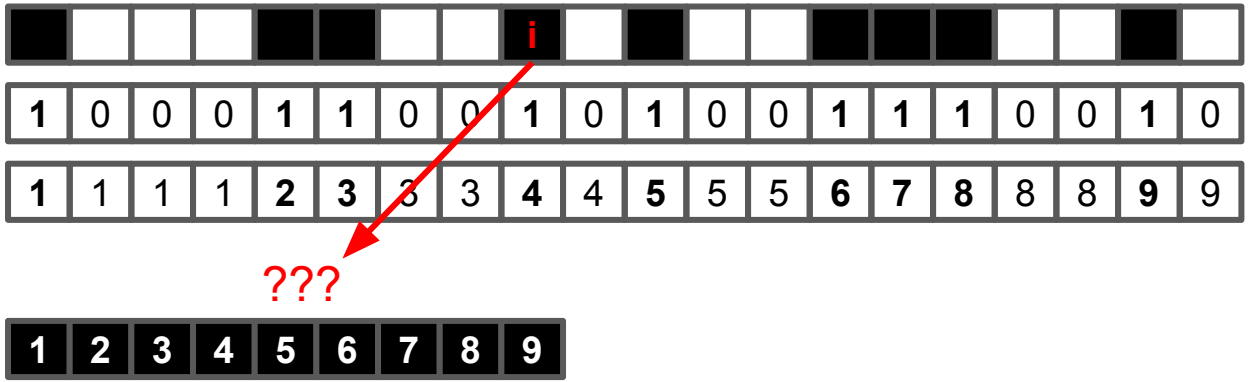
Префиксные суммы: напоминание

- Scan
- Генерация результата в три прохода:
  - **map** - каждый WorkItem записал 1 если результат есть, иначе 0
  - **scan** - посчитали префиксные суммы PrefixSum[i]



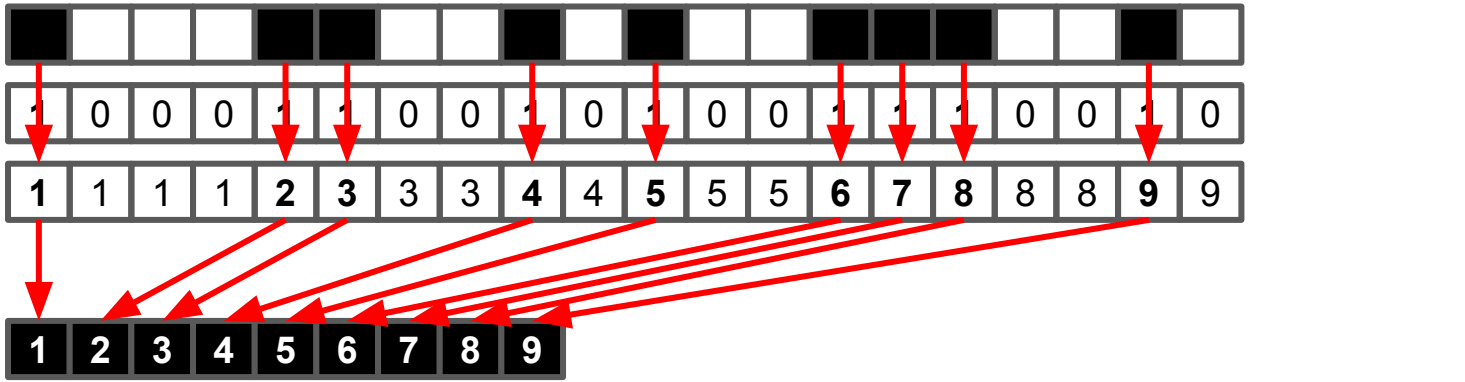
Префиксные суммы: напоминание

- Scan
- Генерация результата в три прохода:
  - **map** - каждый WorkItem записал 1 если результат есть, иначе 0
  - **scan** - посчитали префиксные суммы PrefixSum[i]
  - куда теперь должен писать свой результат WorkItem i?



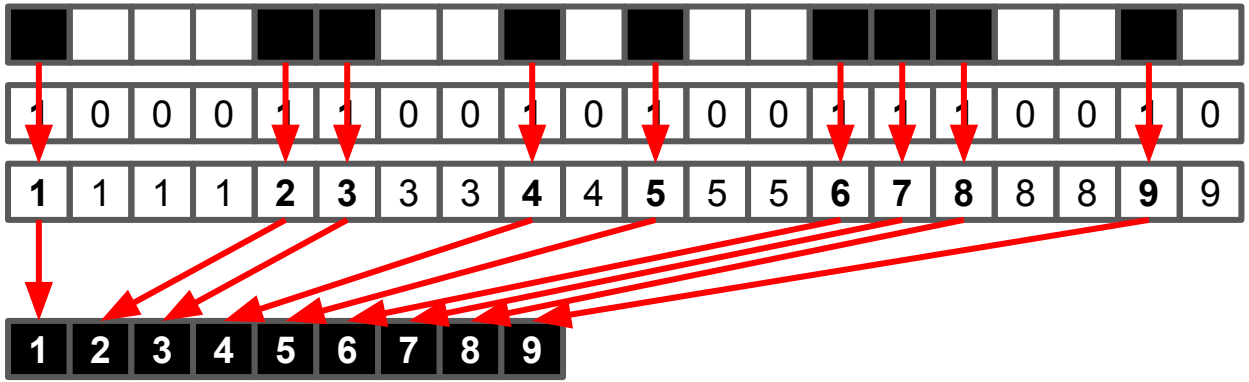
Префиксные суммы: напоминание

- Scan
- Генерация результата в три прохода:
  - **map** - каждый WorkItem записал 1 если результат есть, иначе 0
  - **scan** - посчитали префиксные суммы PrefixSum[i]
  - **scatter** - WorkItem i пишет свой *результат* в *Result*[**PrefixSum[i]**]

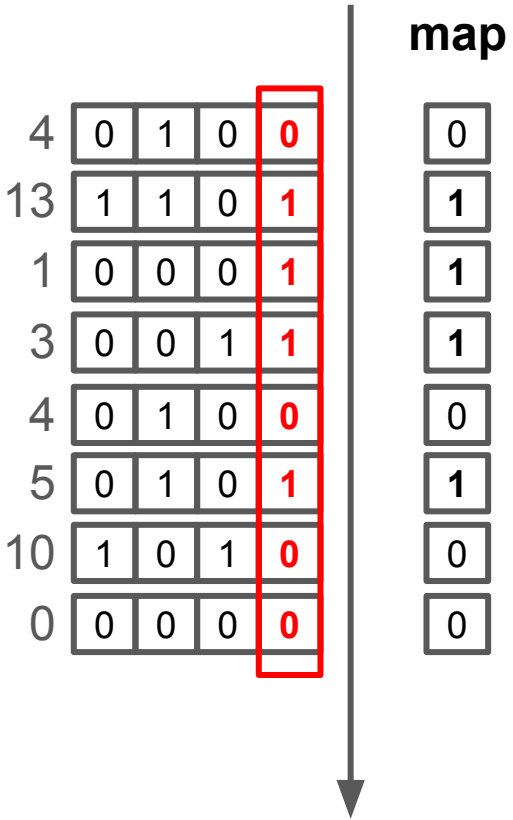


Префиксные суммы: напоминание

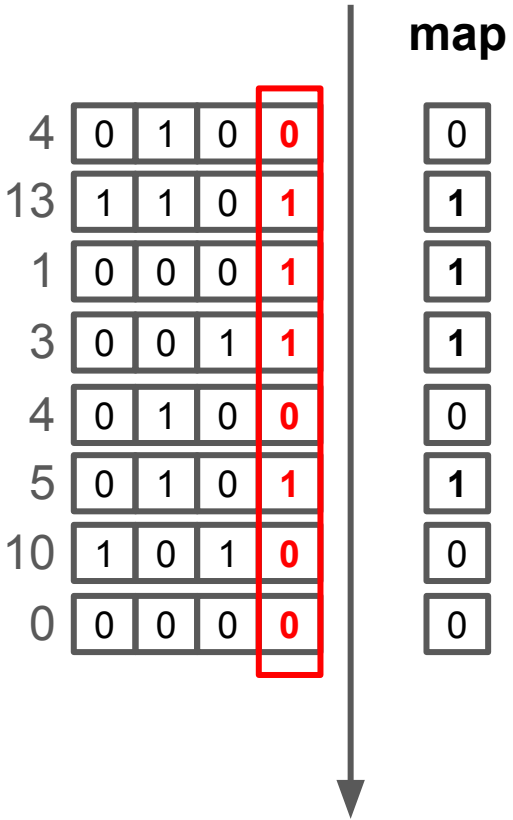
- Scan Как выразить поразрядную сортировку через map/scan/scatter?
- Генерация результата в три прохода:
  - **map** - каждый WorkItem записал 1 если результат есть, иначе 0
  - **scan** - посчитали префиксные суммы PrefixSum[i]
  - **scatter** - WorkItem i пишет свой *результат* в *Result*[**PrefixSum[i]**]



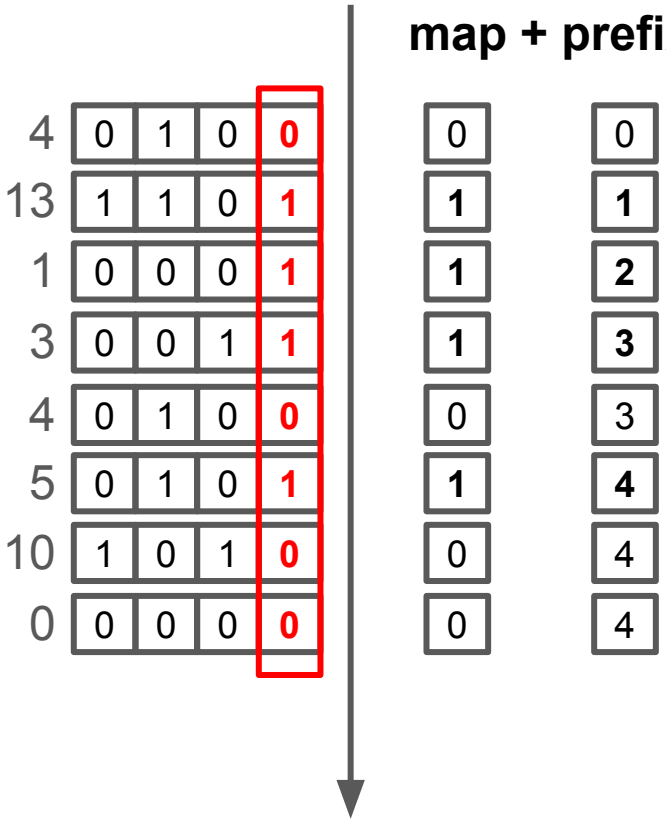
Radix sort: сведение к prefix scan



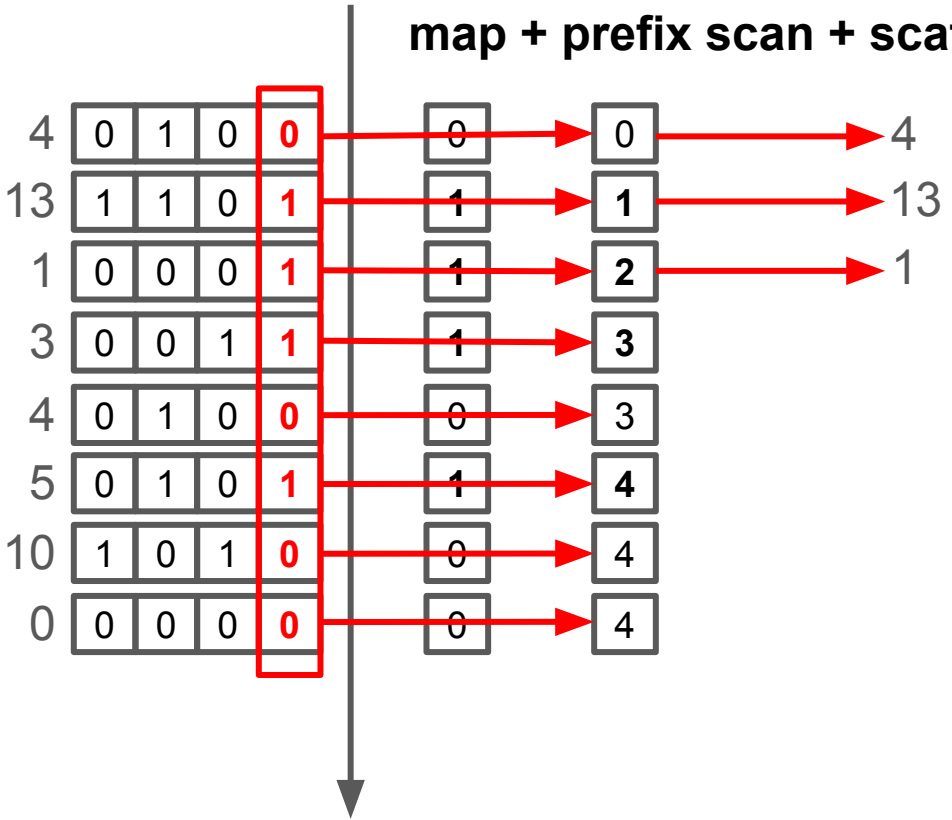
Radix sort: сведение к prefix scan



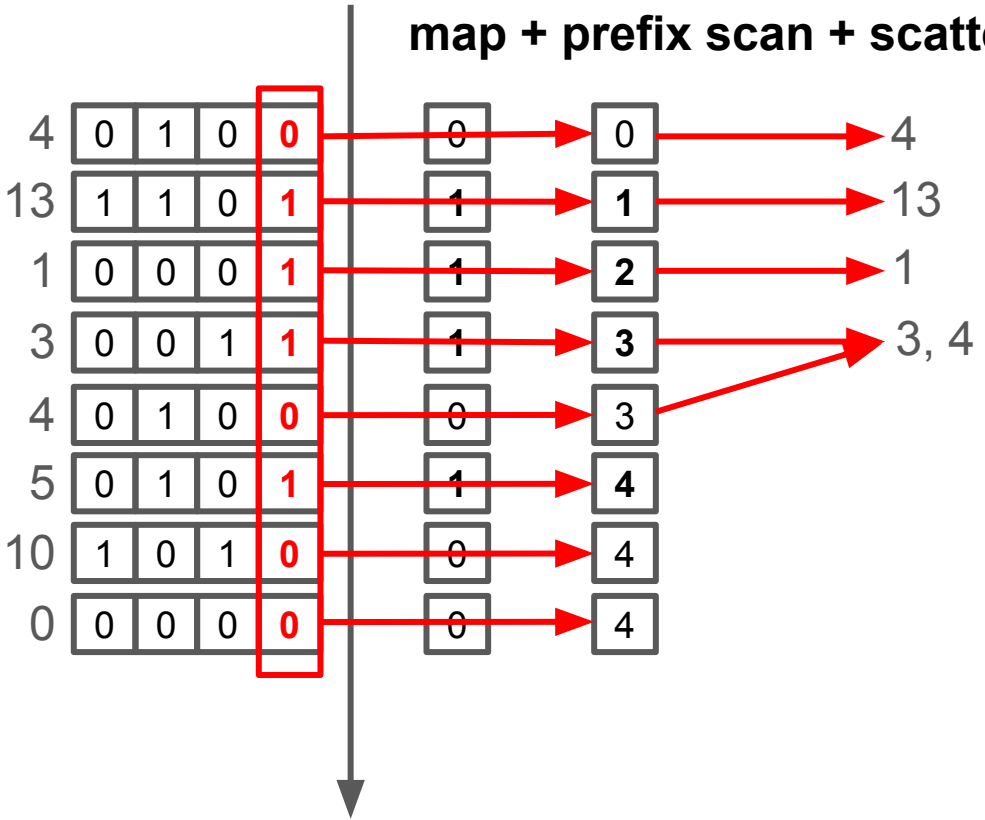
Radix sort: сведение к prefix scan



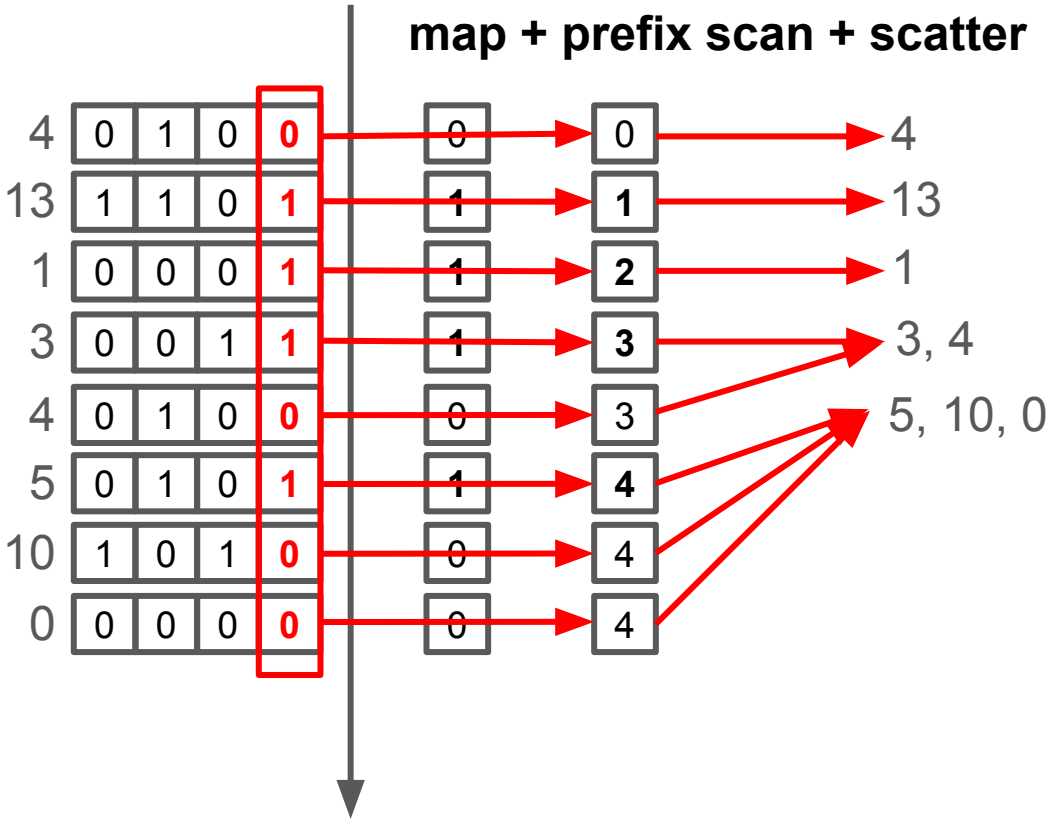
Radix sort: сведение к prefix scan



Radix sort: сведение к prefix scan

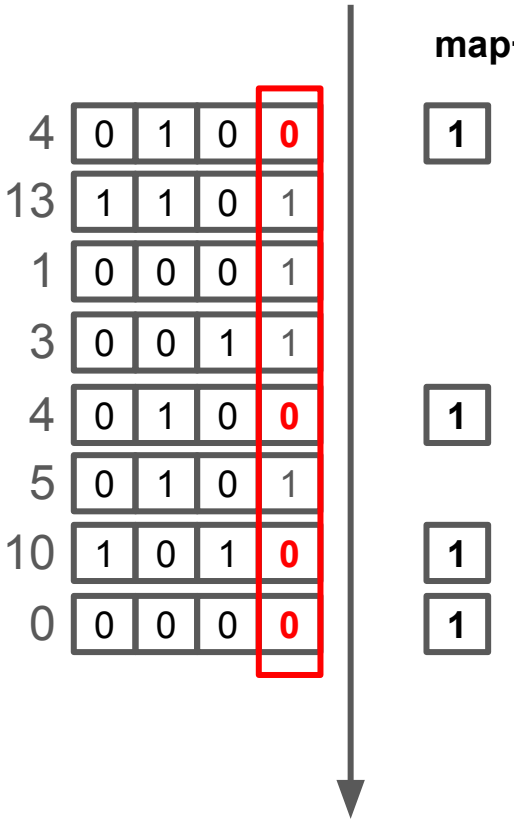


Radix sort: сведение к prefix scan

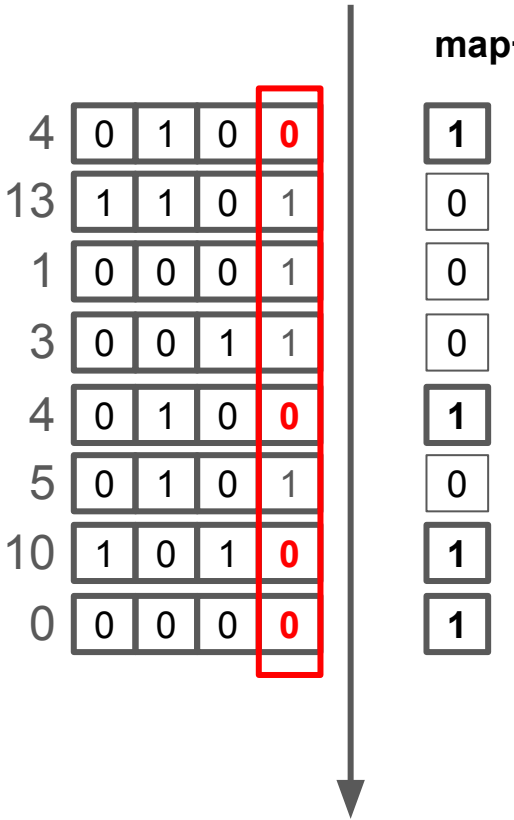


Какие есть идеи?

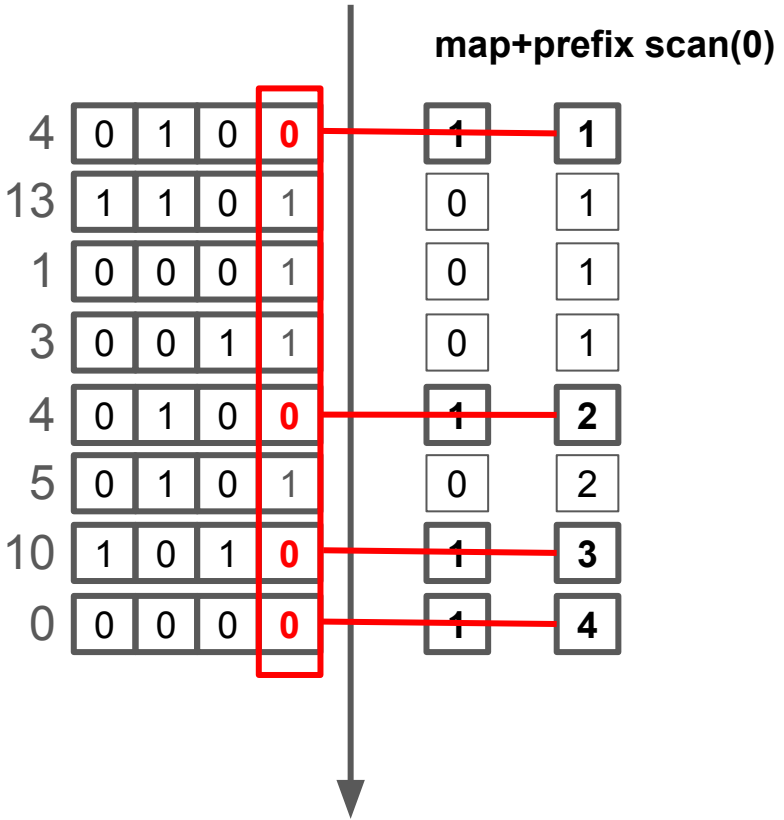
Radix sort: сведение к prefix scan



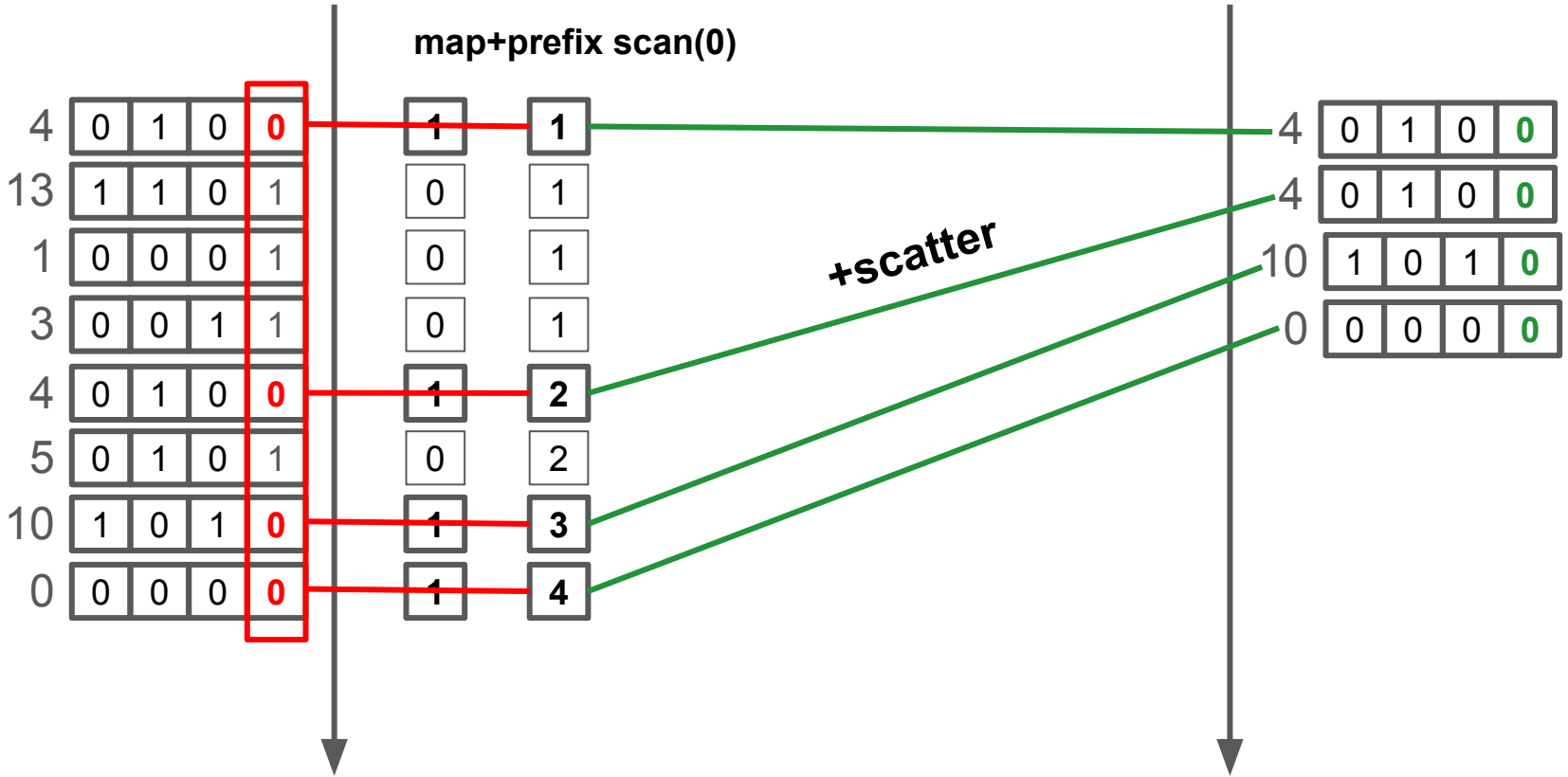
Radix sort: сведение к prefix scan



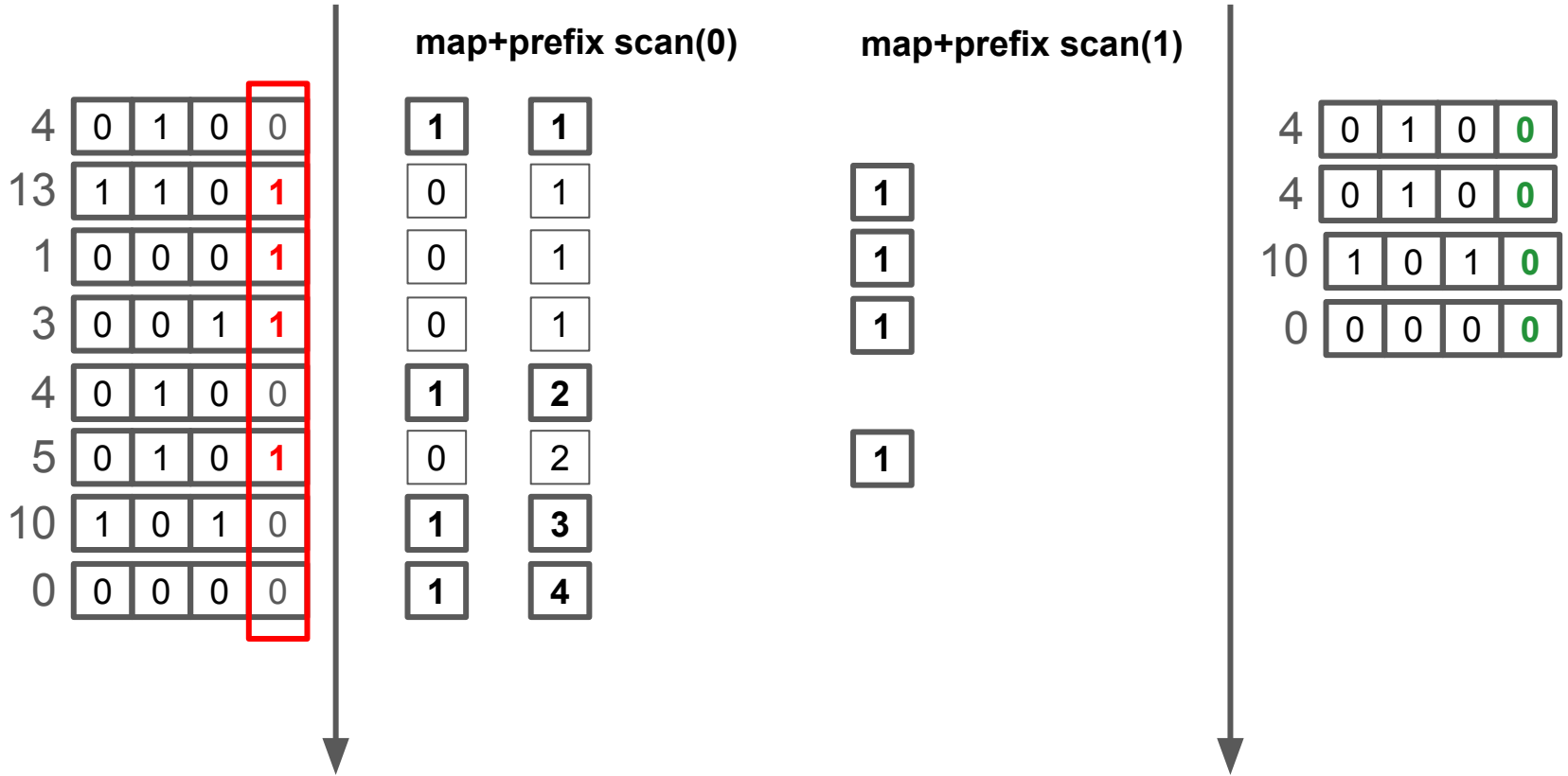
Radix sort: сведение к prefix scan



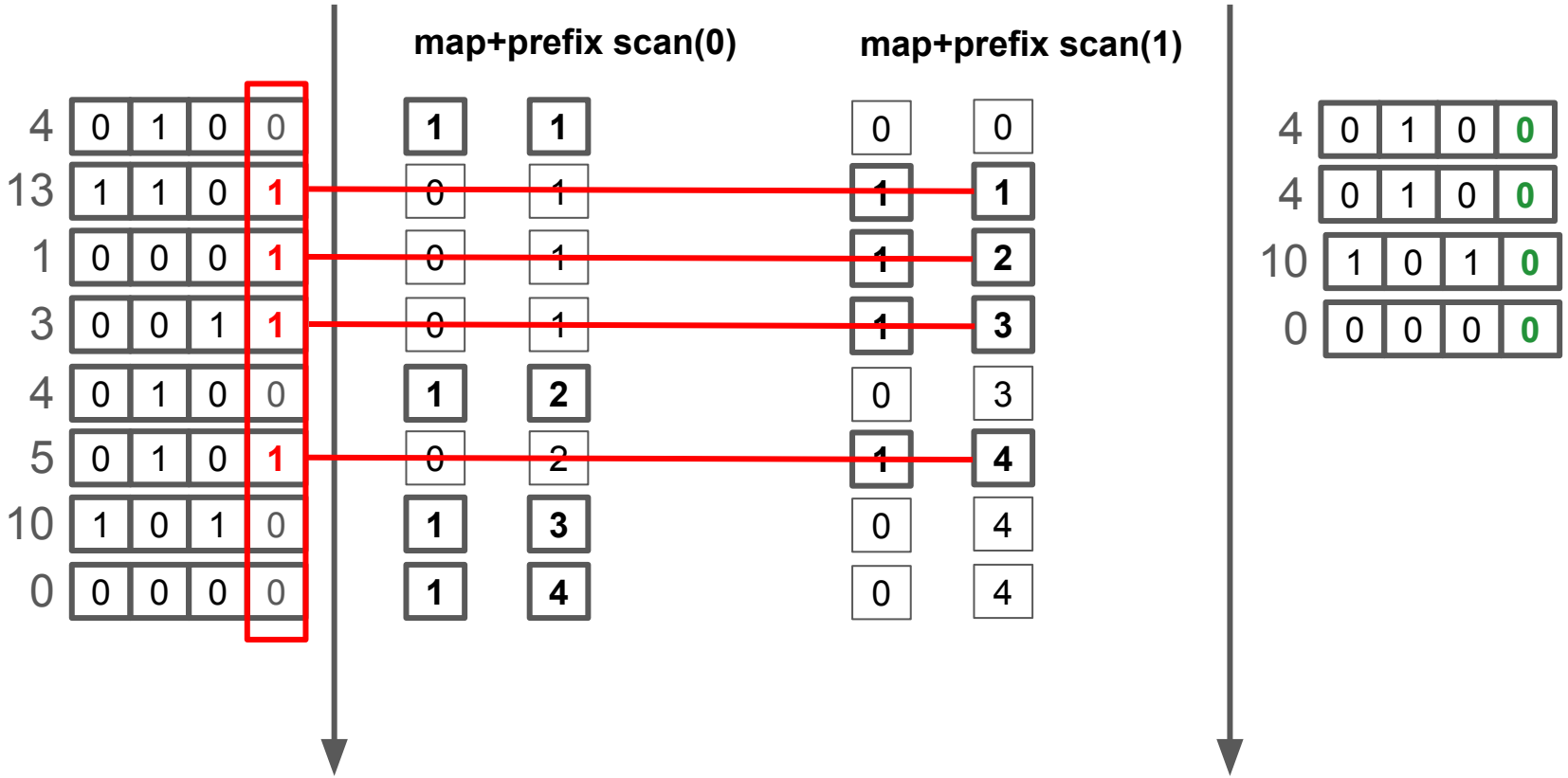
Radix sort: сведение к prefix scan



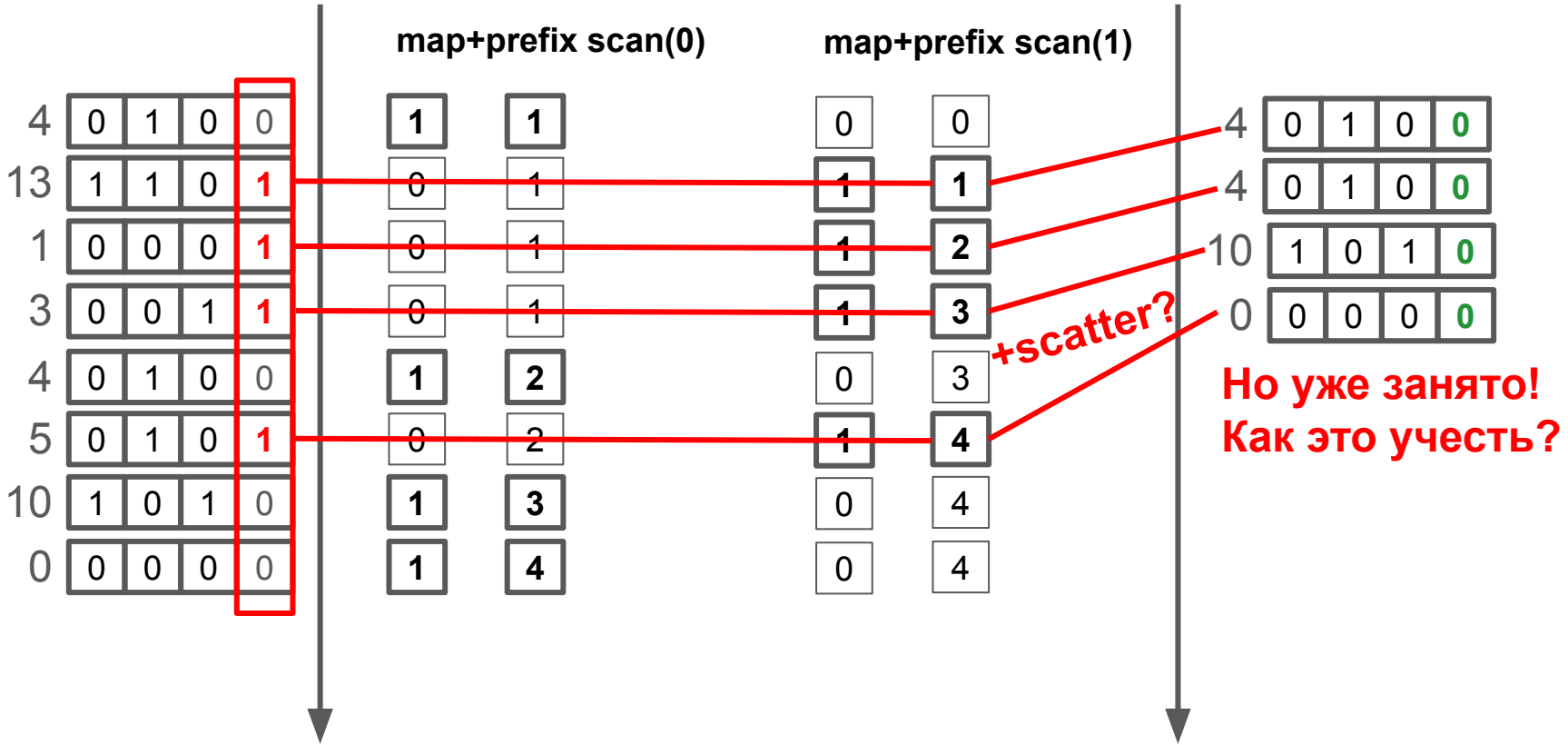
Radix sort: сведение к prefix scan



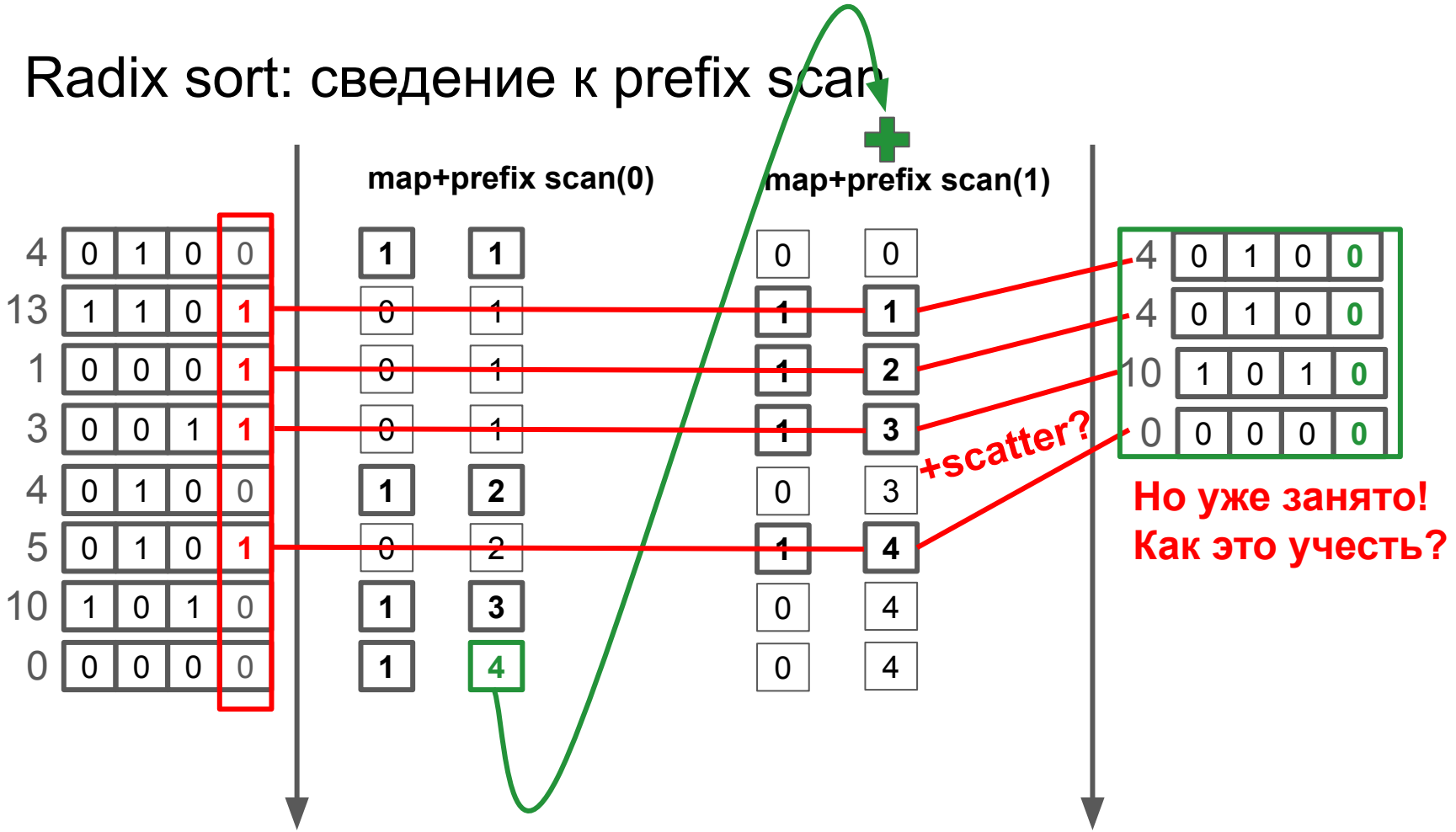
Radix sort: сведение к prefix scan



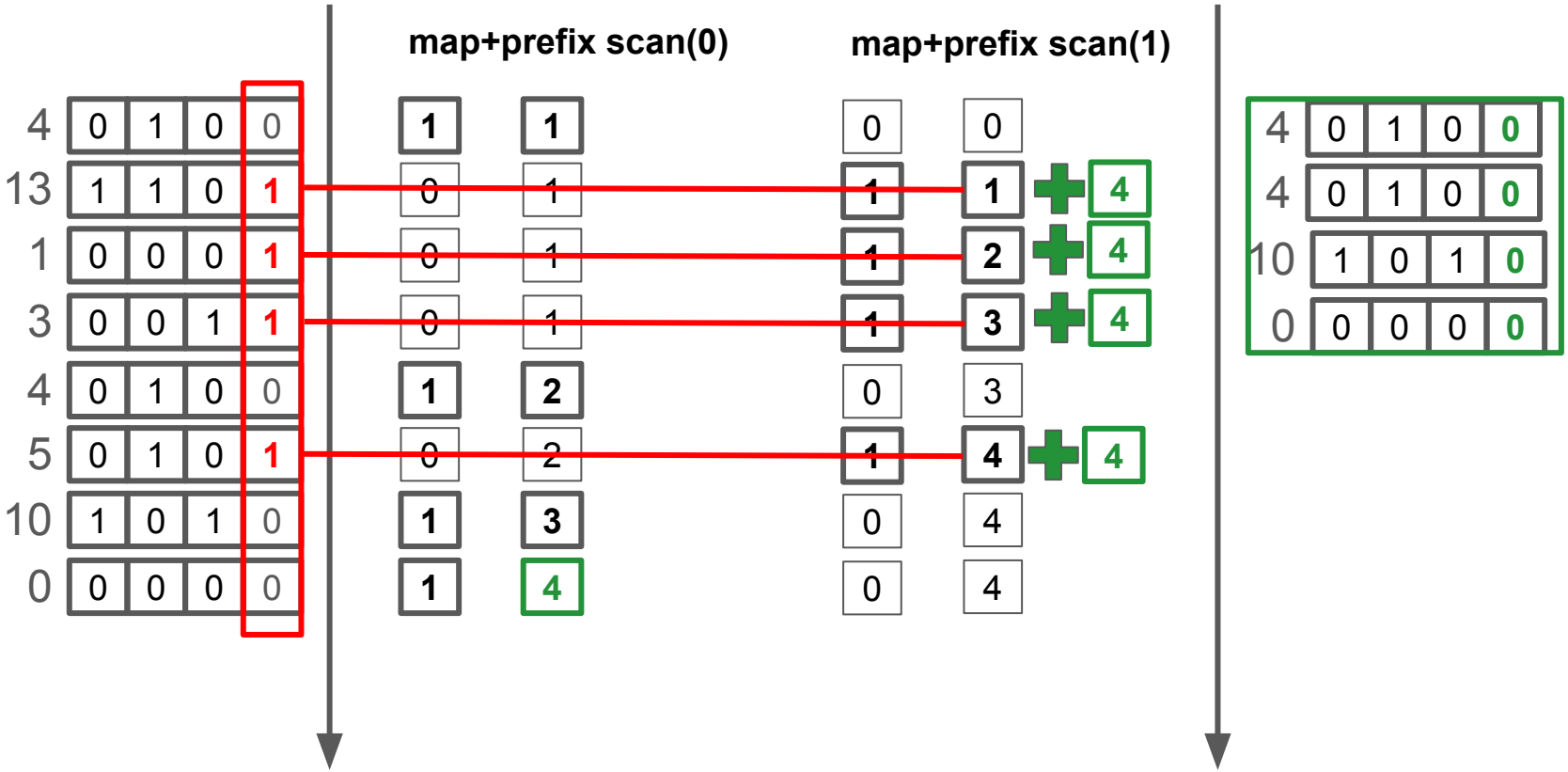
Radix sort: сведение к prefix scan



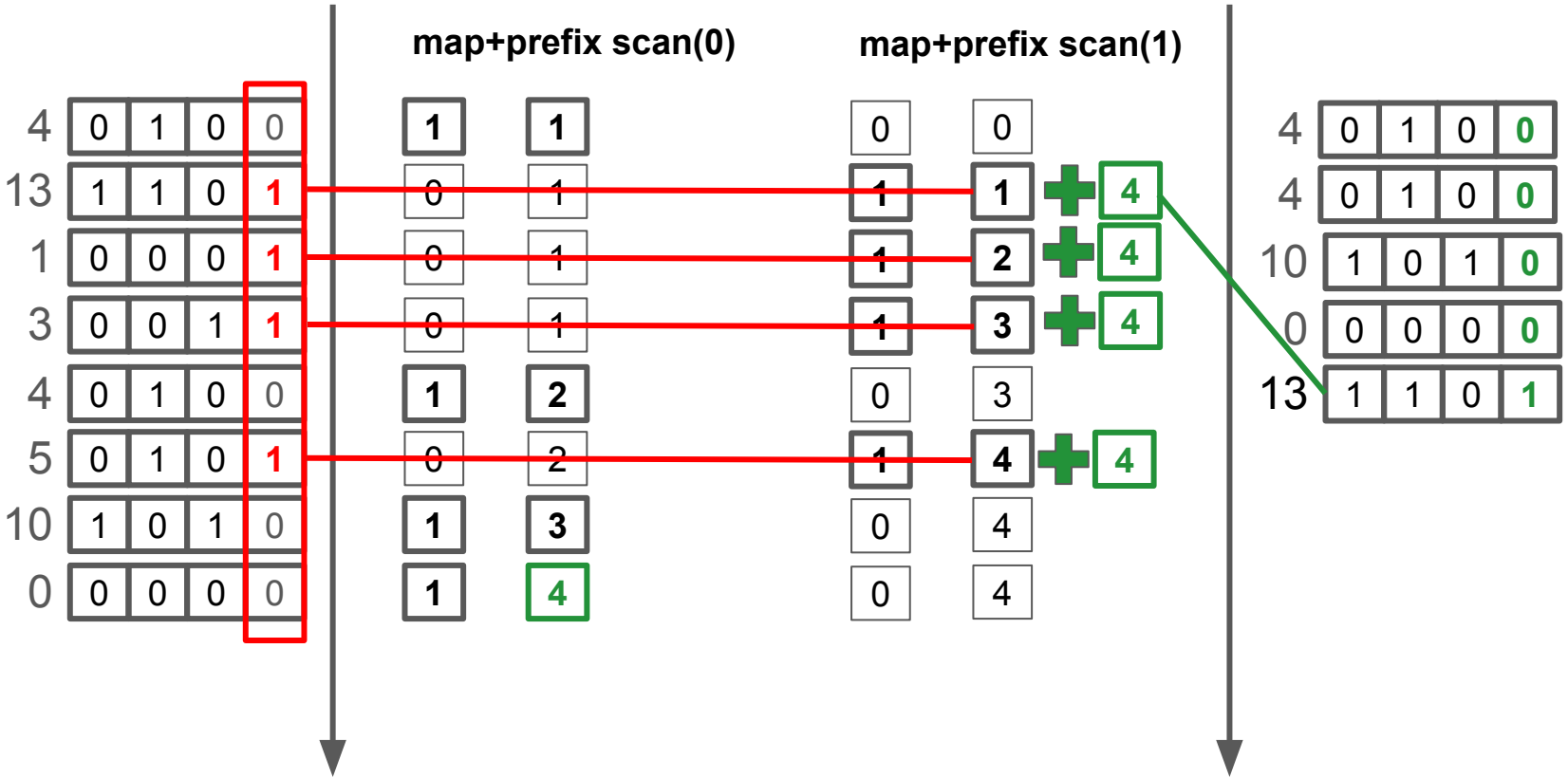
Radix sort: сведение к prefix scan



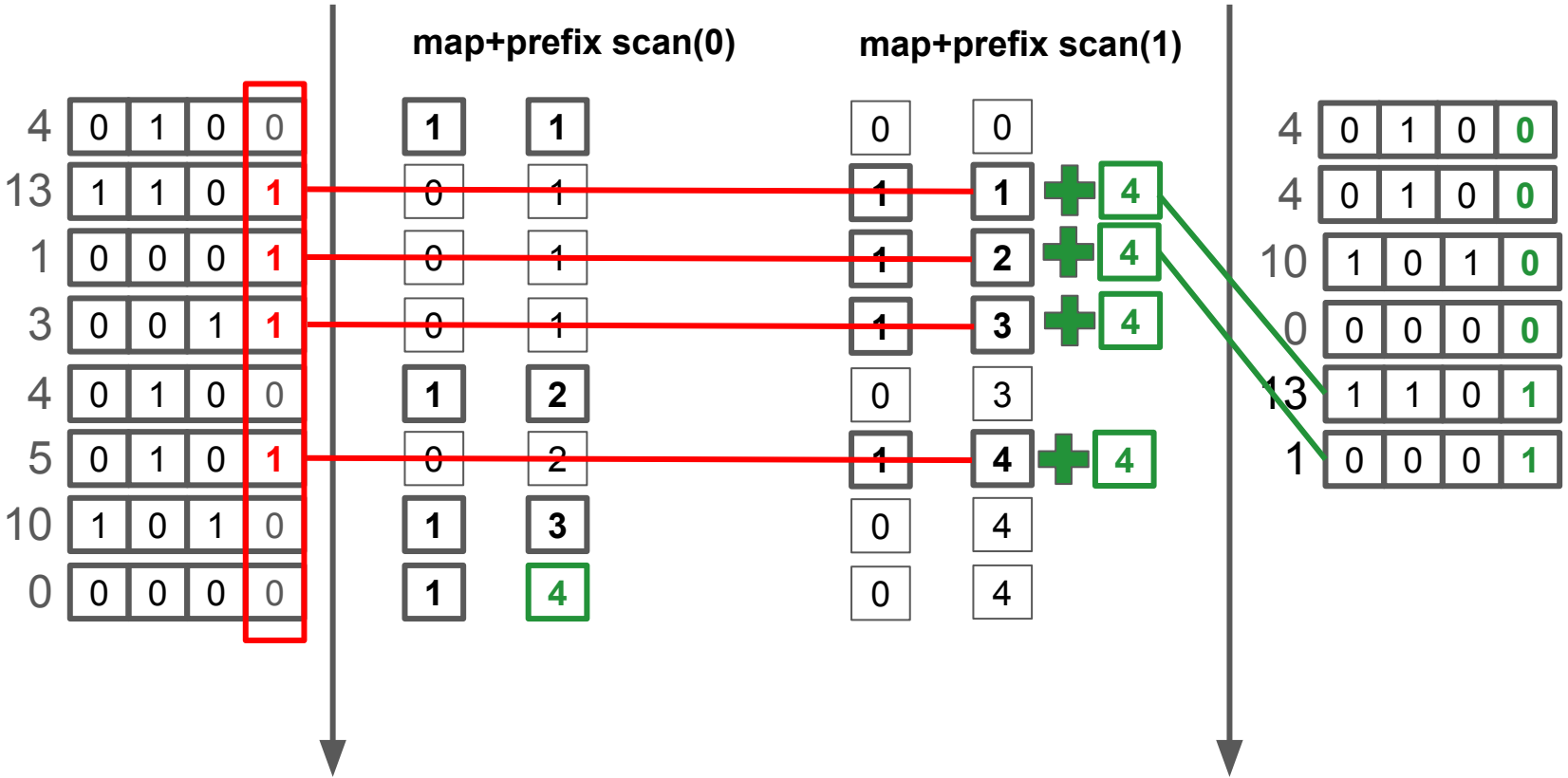
Radix sort: сведение к prefix scan



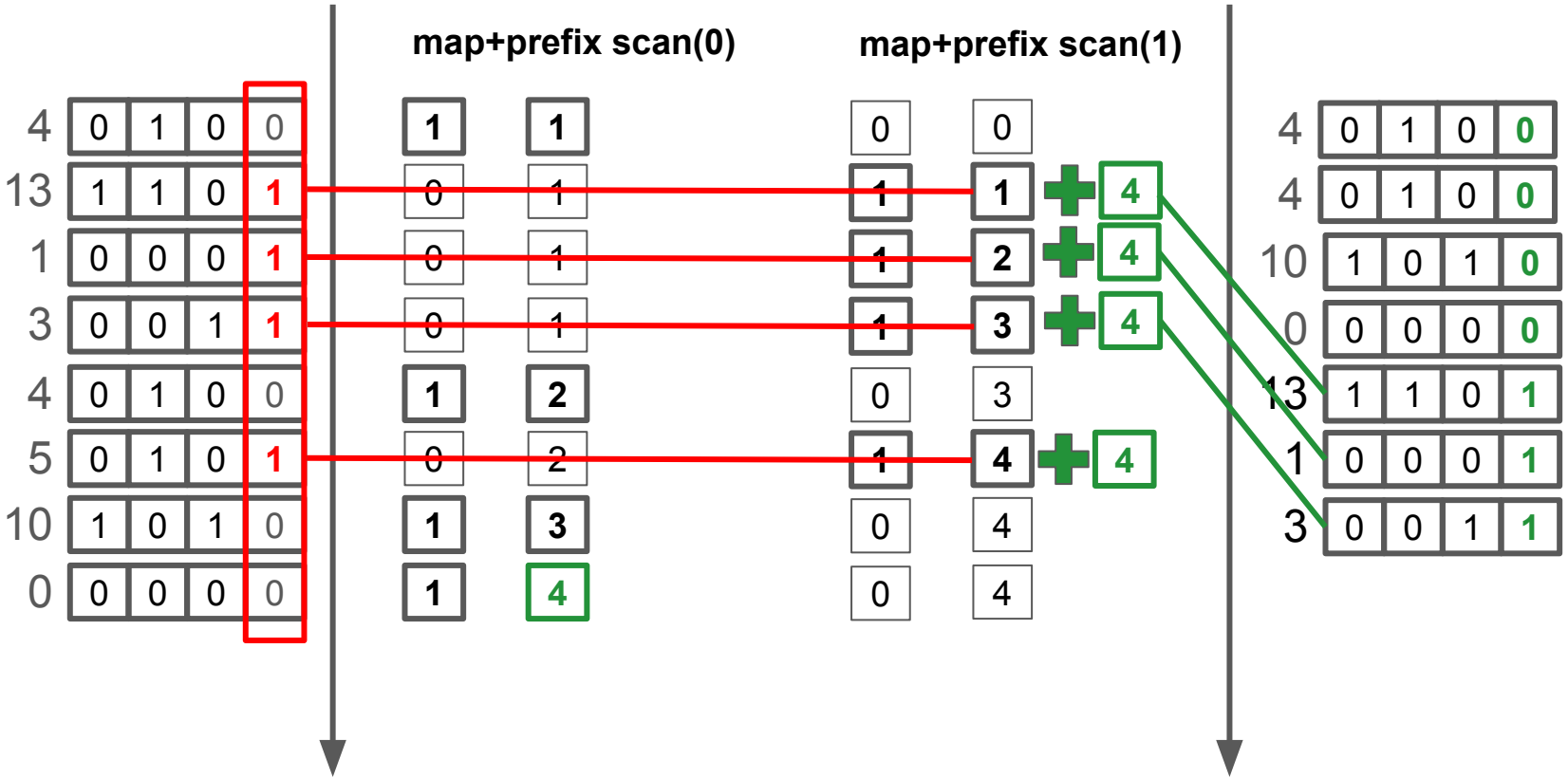
Radix sort: сведение к prefix scan



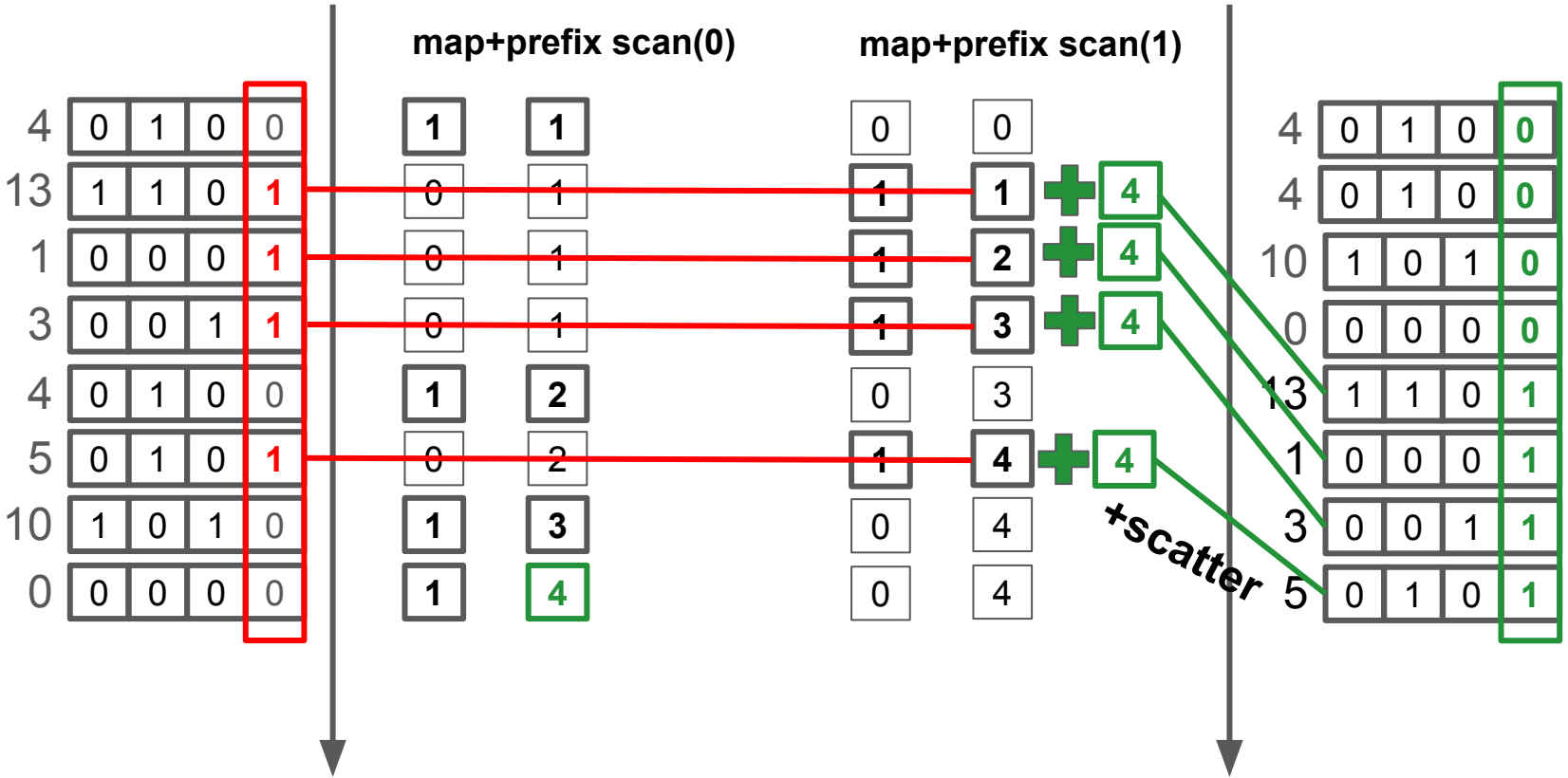
Radix sort: сведение к prefix scan



Radix sort: сведение к prefix scan

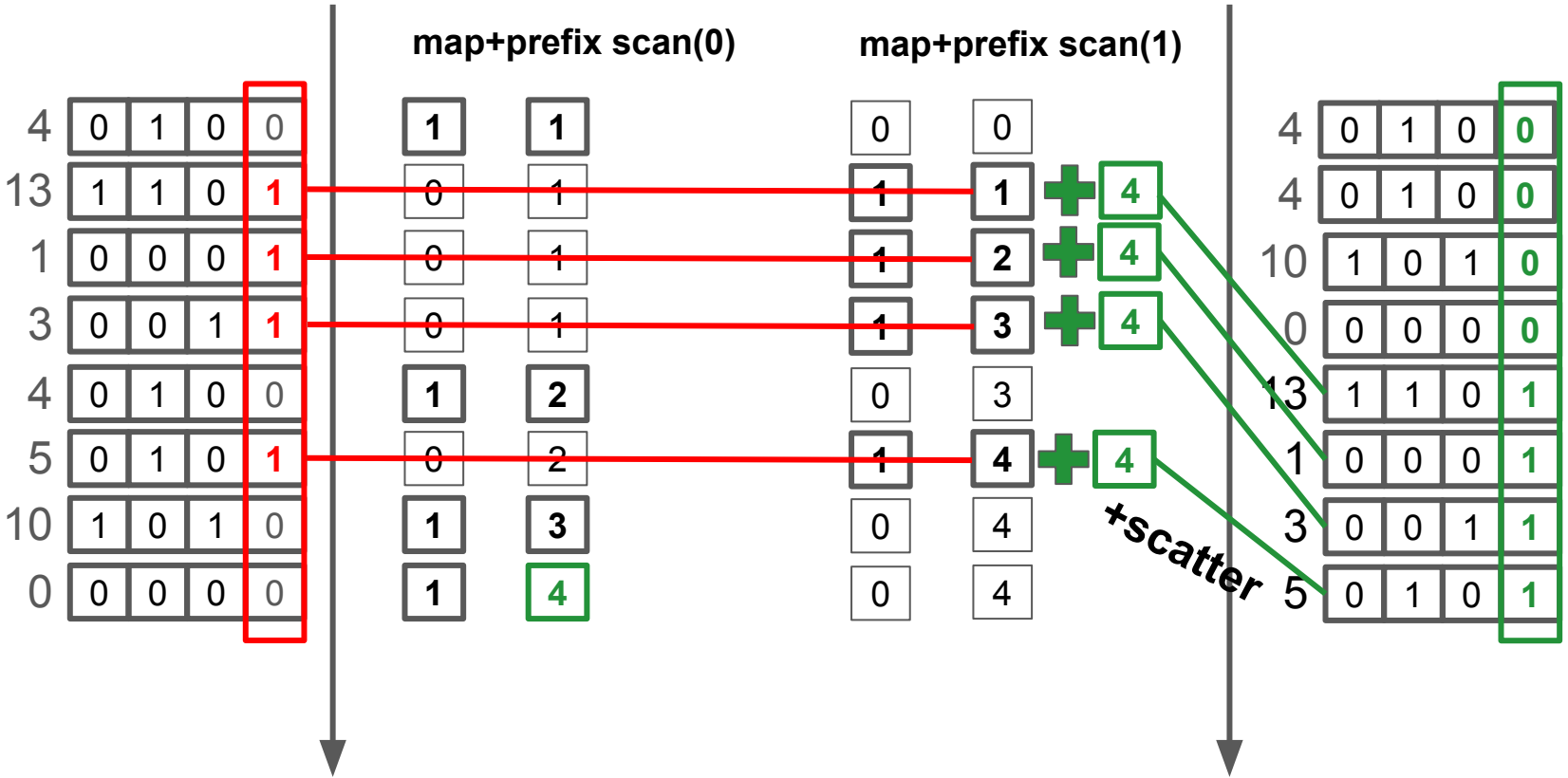


Radix sort: сведение к prefix scan



Вопросы?

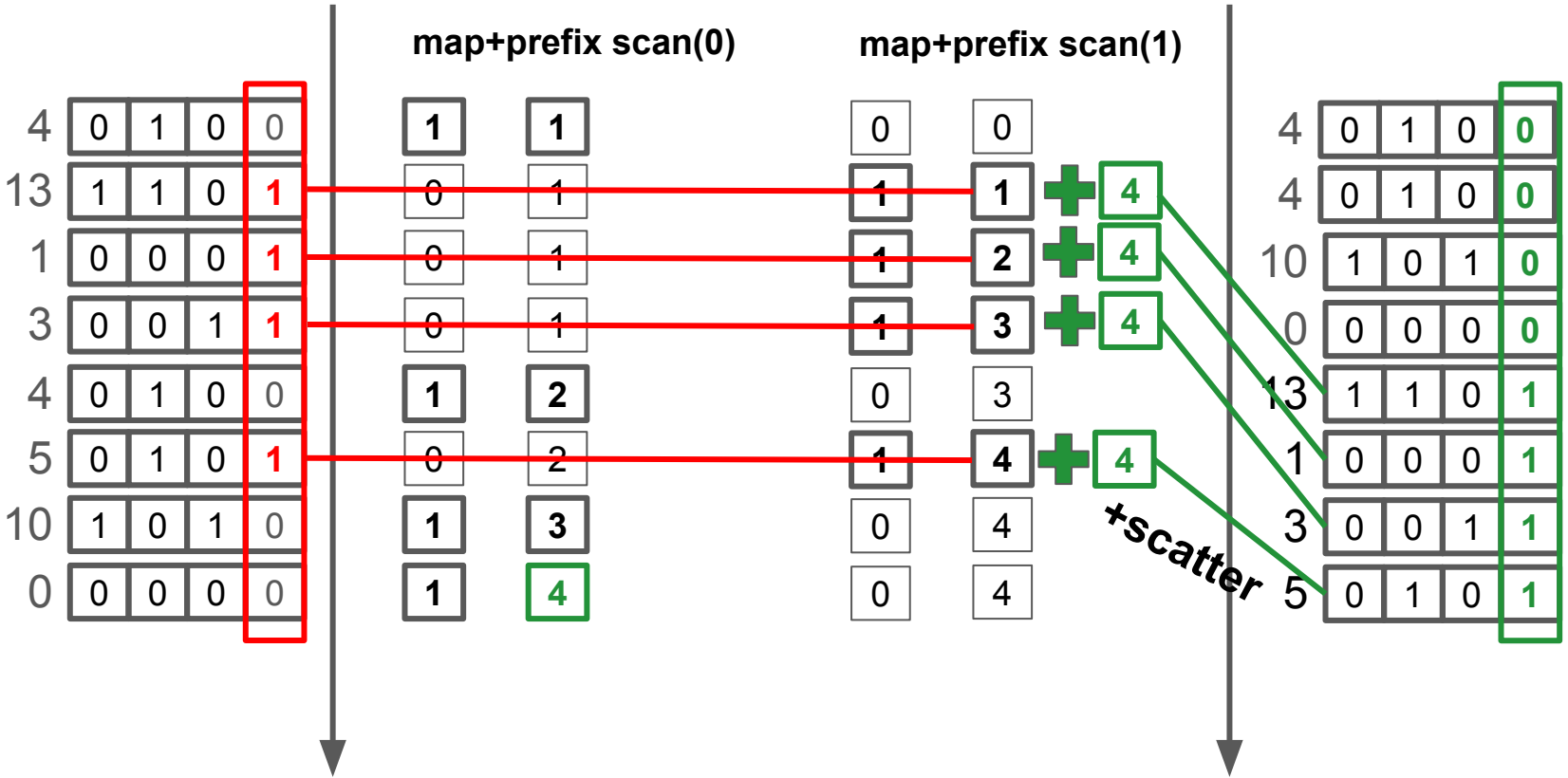
Radix sort: сведение к prefix scan



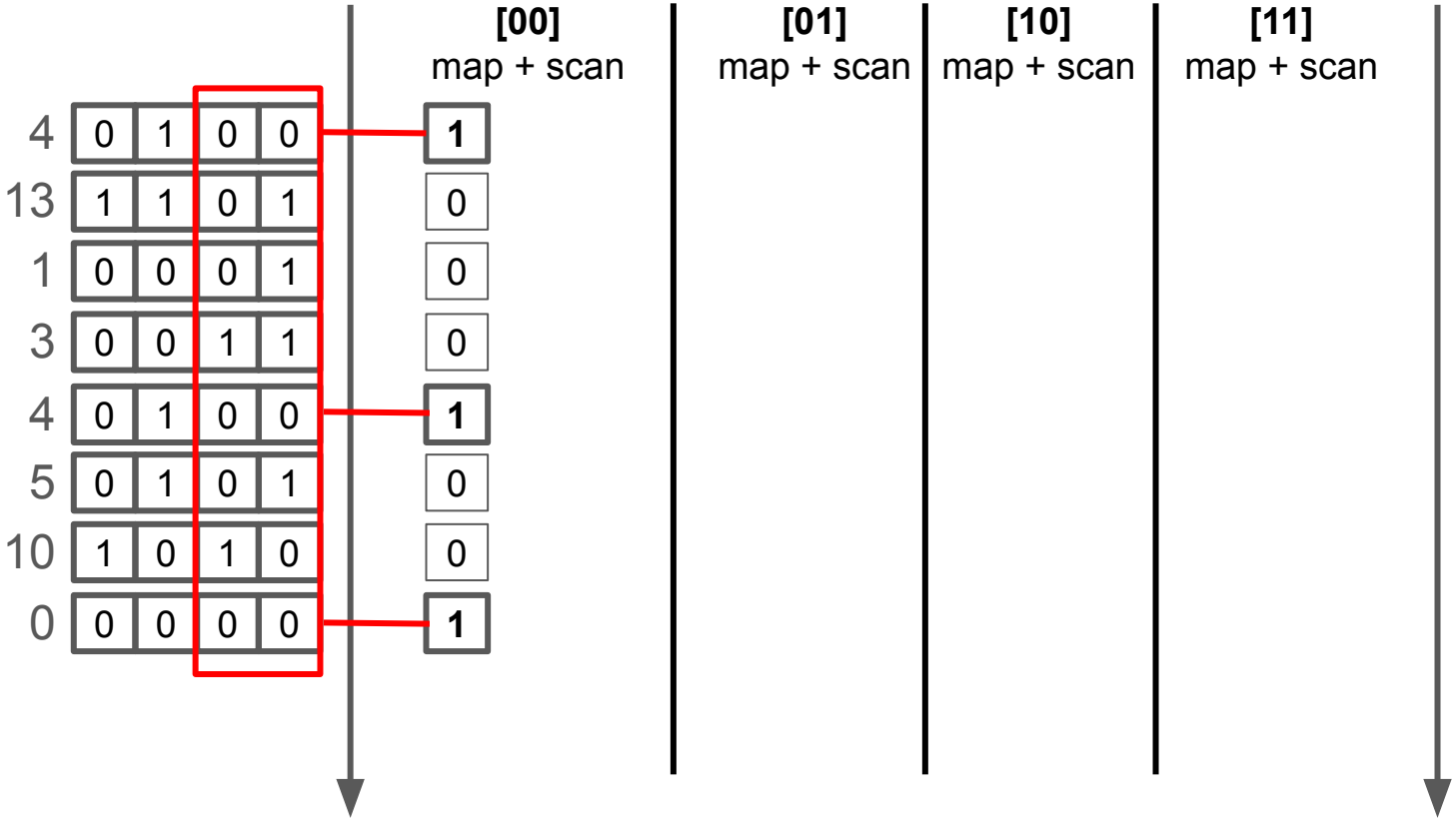

Перерыв!
Streaming
Pubiprocestor
VIDIA

В какой ресурс мы уперлись? Как ускорить?

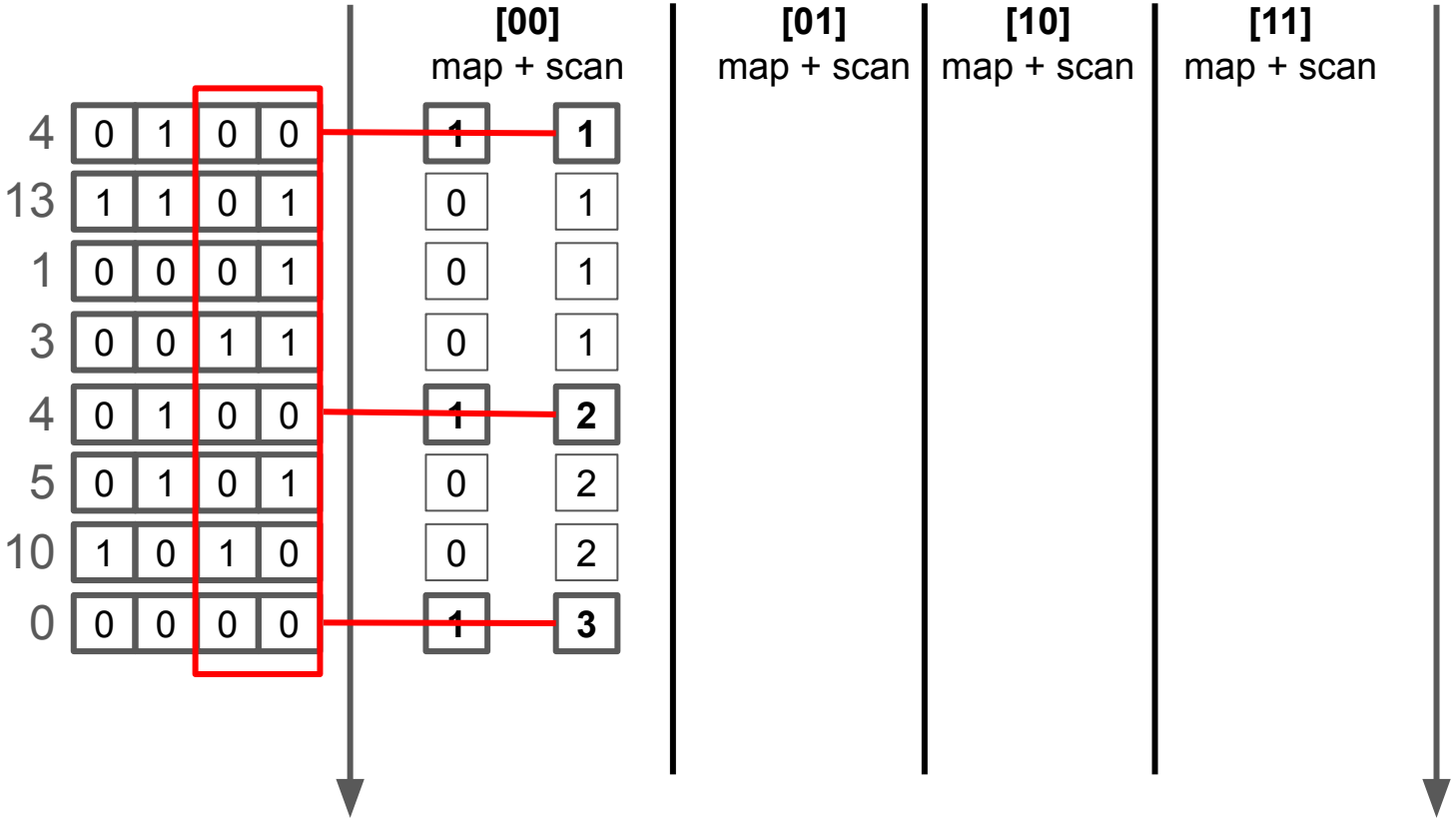
Radix sort: сведение к prefix scan



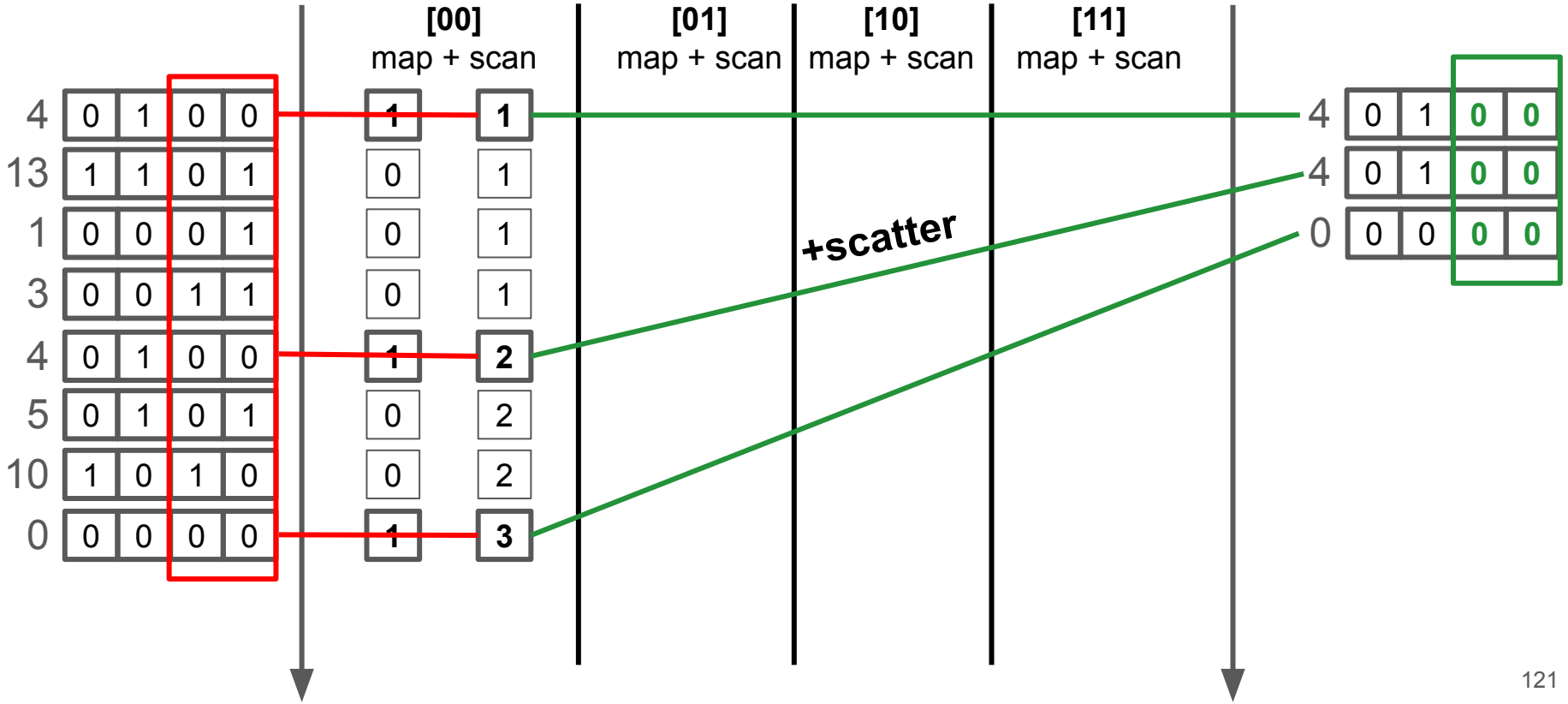
Radix sort: сортировка по **нескольким разрядам**



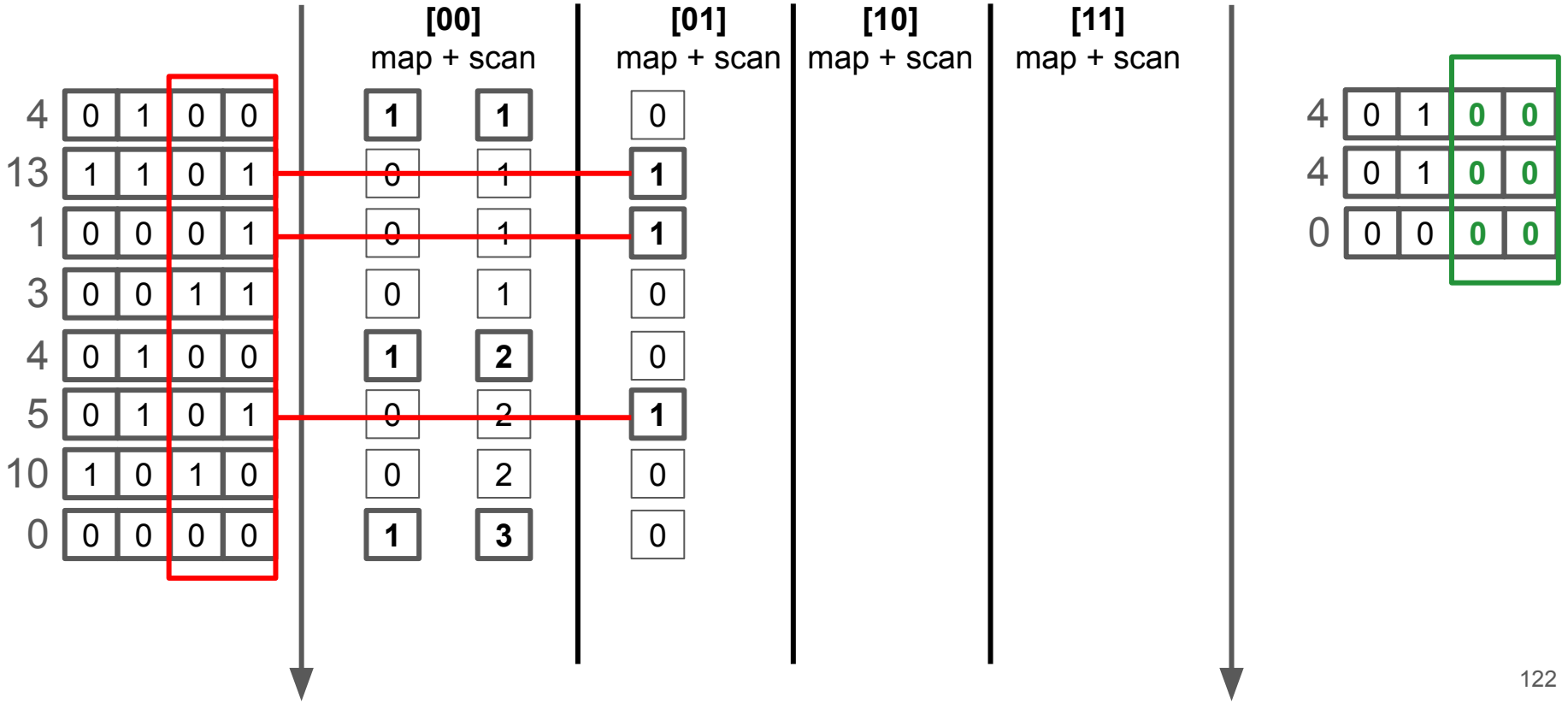
Radix sort: сортировка по **нескольким разрядам**



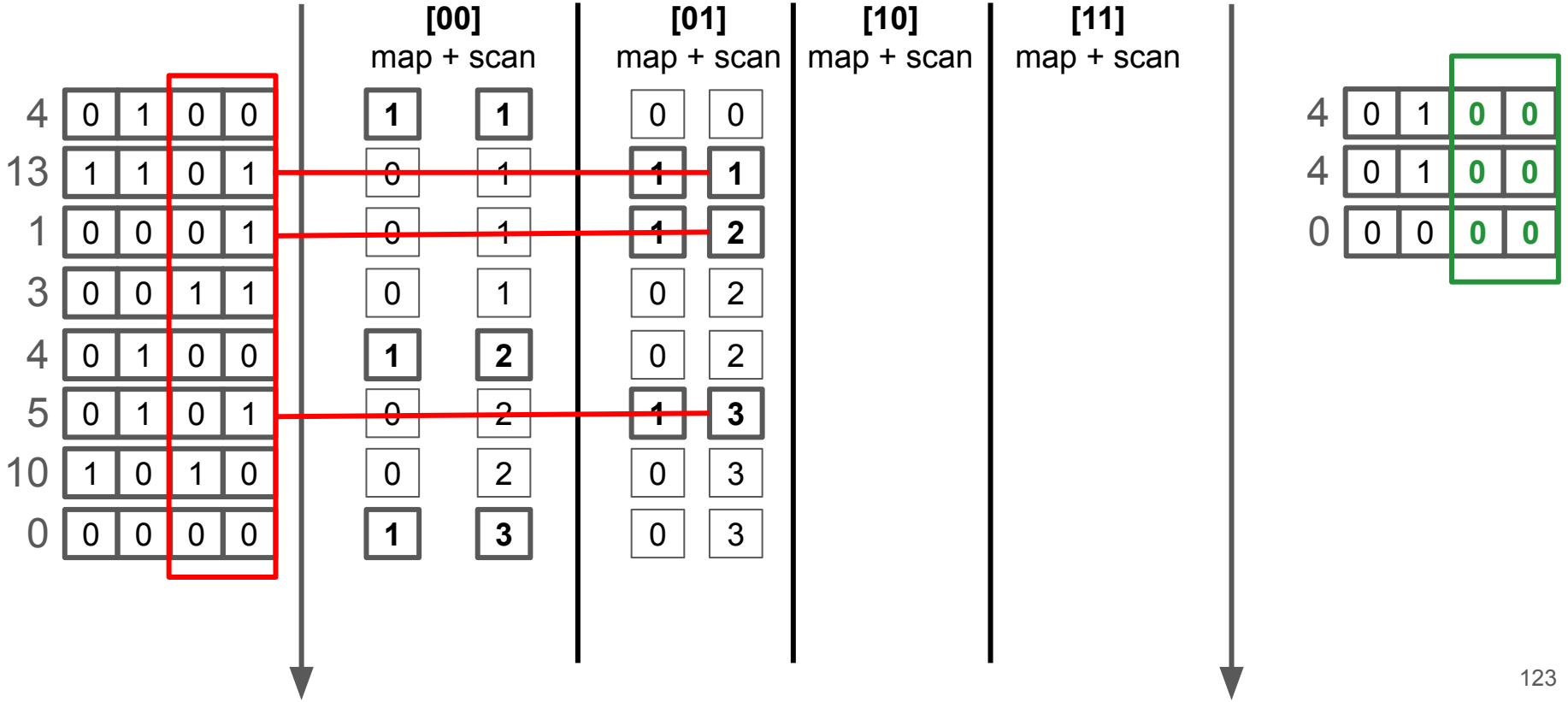
Radix sort: сортировка по **нескольким разрядам**



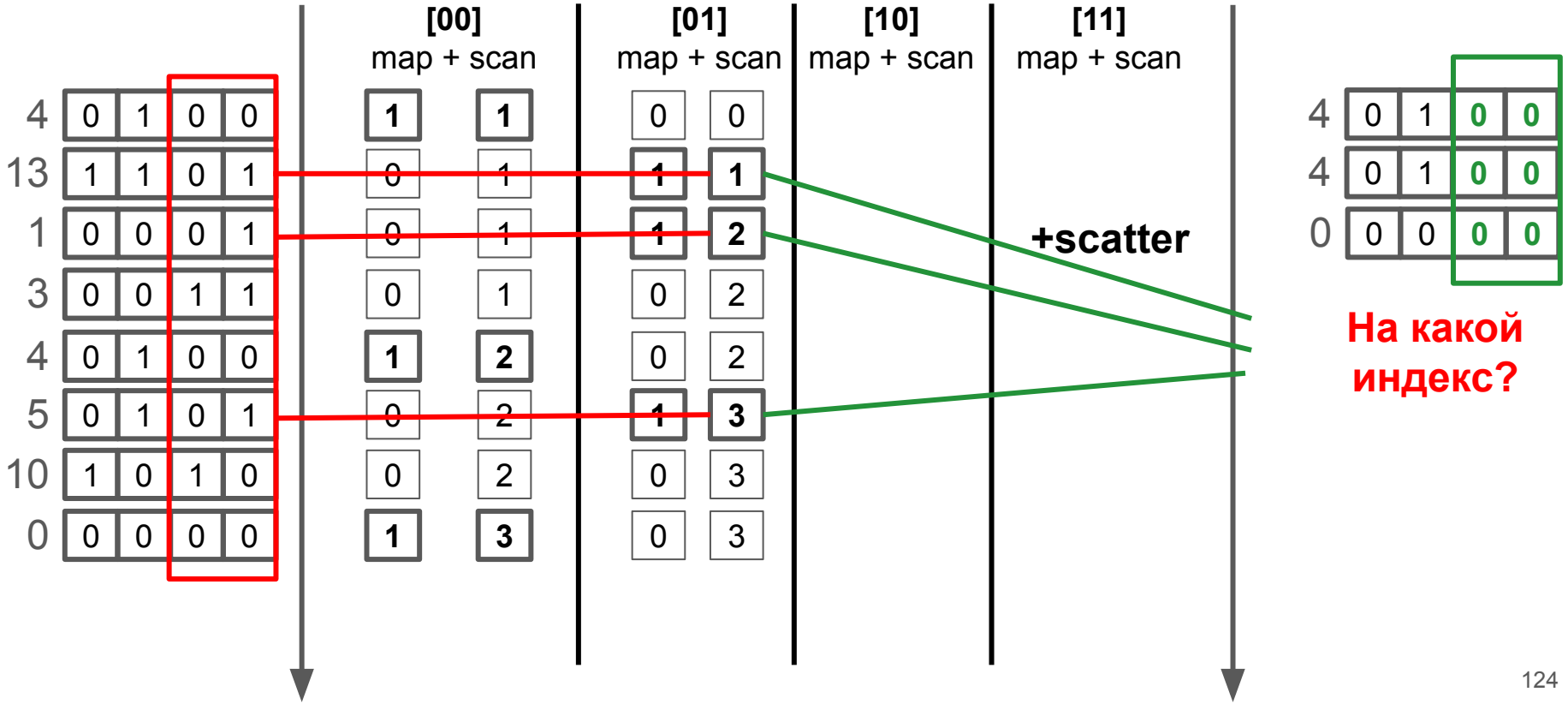
Radix sort: сортировка по **нескольким разрядам**



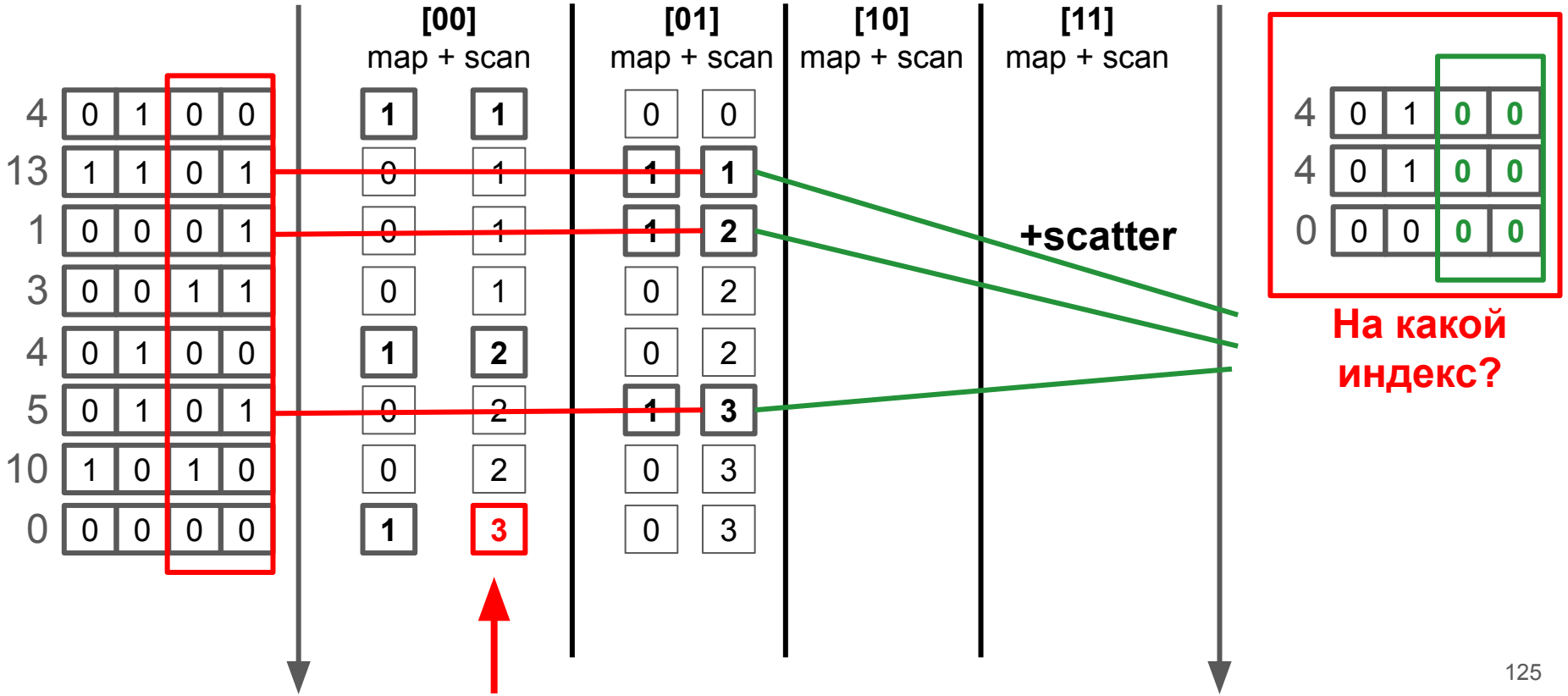
Radix sort: сортировка по **нескольким разрядам**



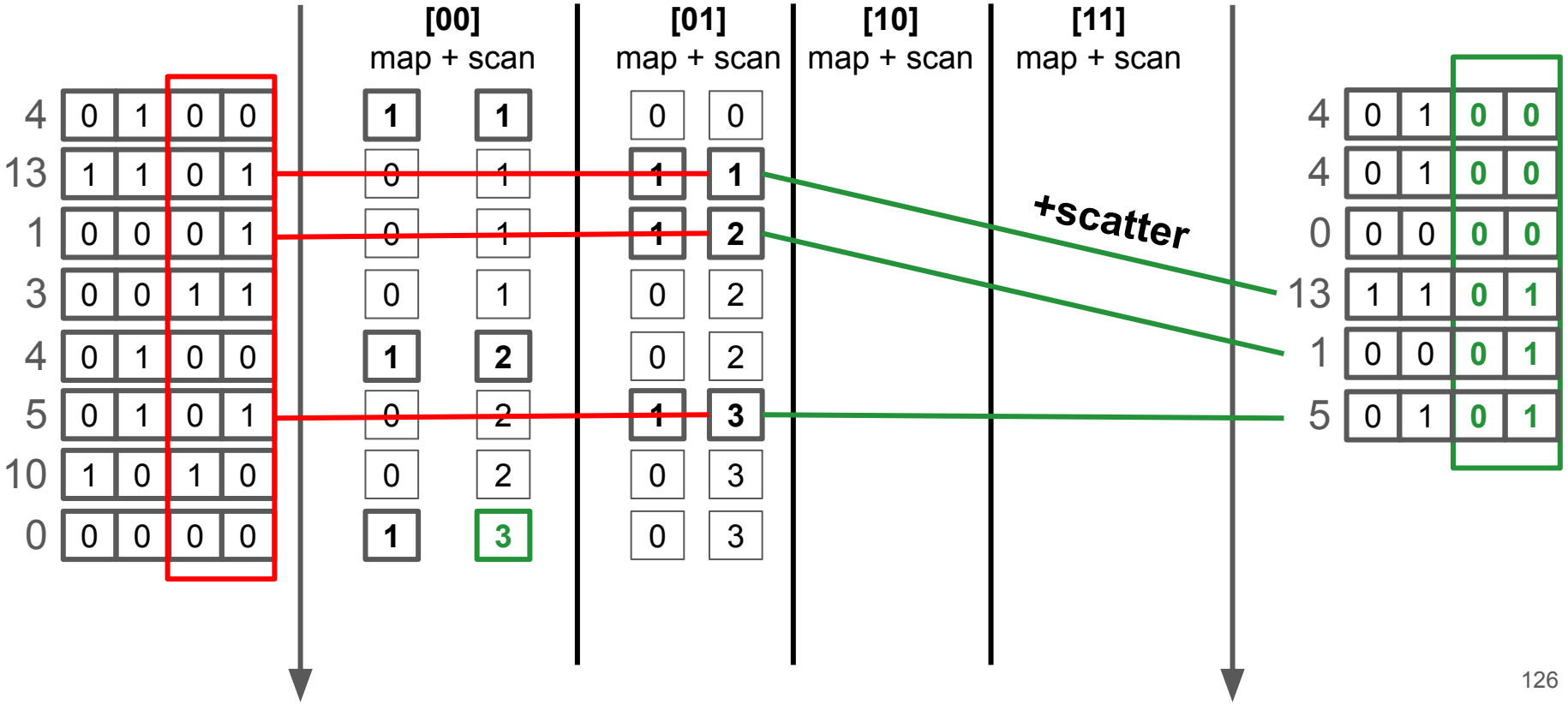
Radix sort: сортировка по **нескольким разрядам**



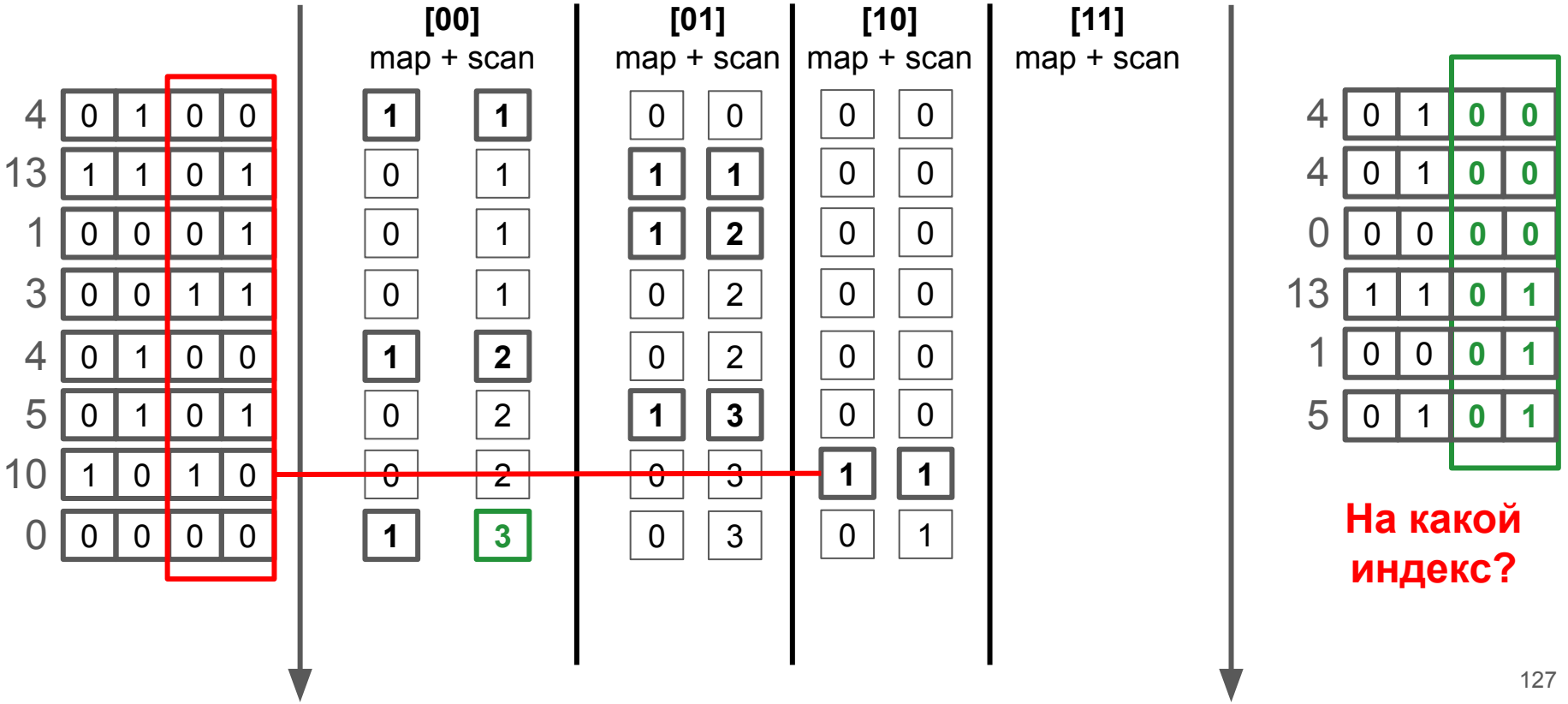
Radix sort: сортировка по **нескольким разрядам**



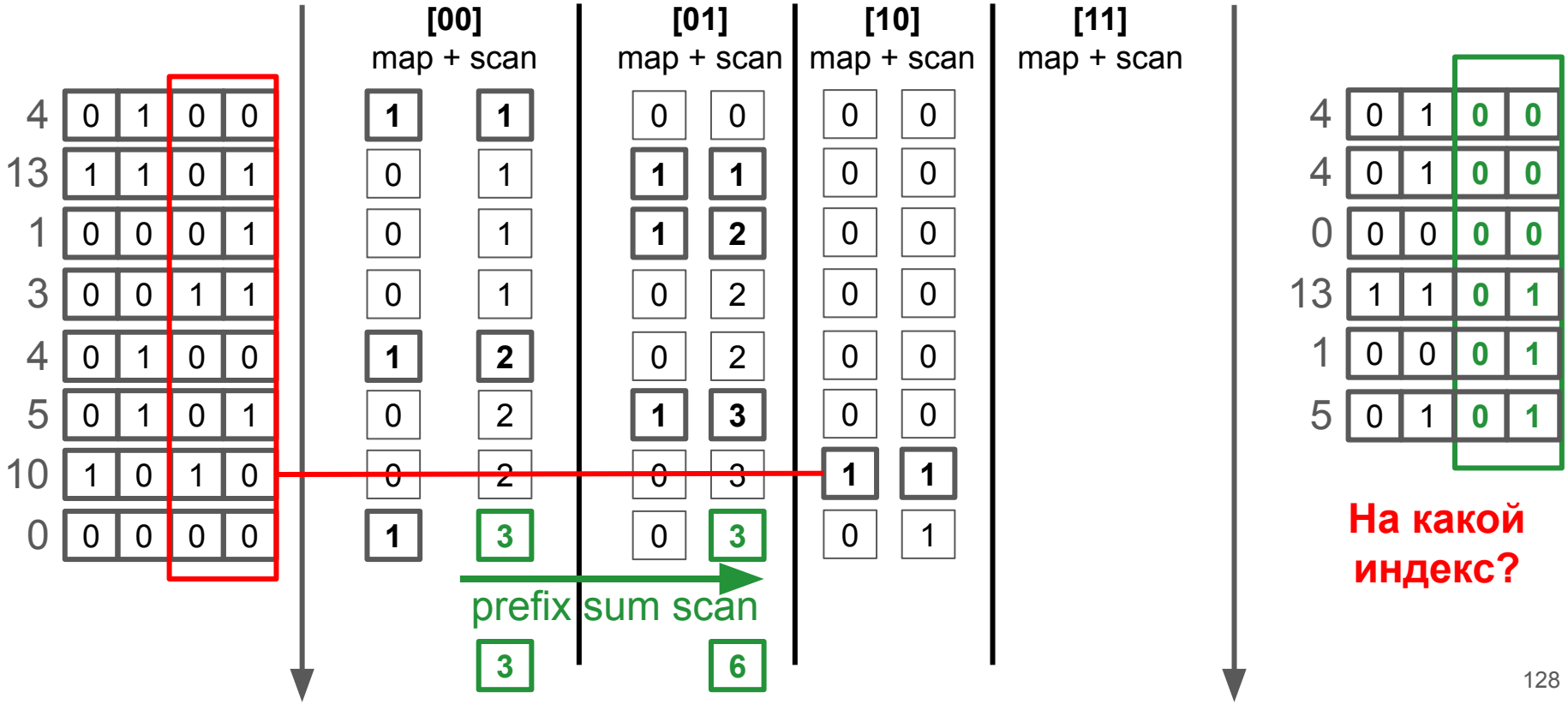
Radix sort: сортировка по **нескольким разрядам**



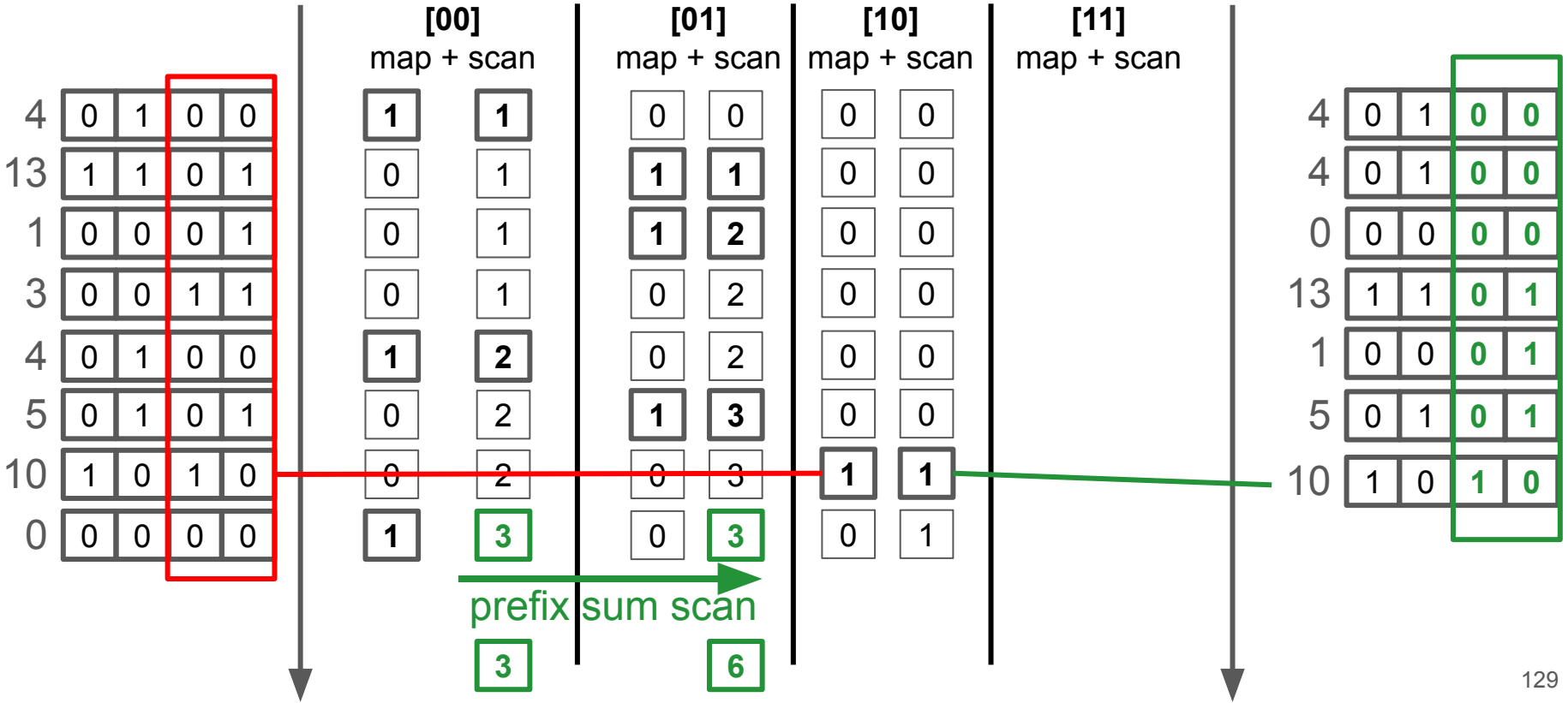
Radix sort: сортировка по **нескольким разрядам**



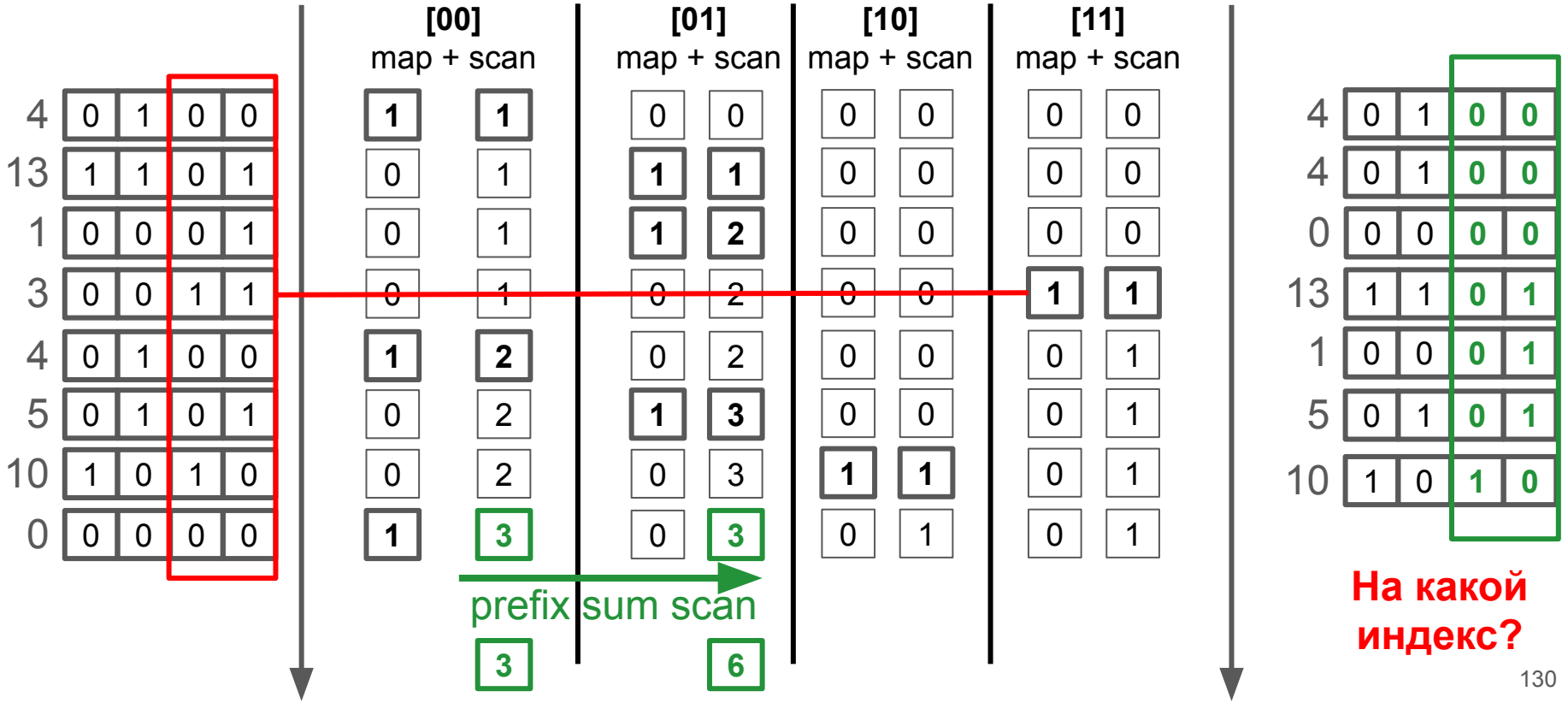
Radix sort: сортировка по **нескольким разрядам**



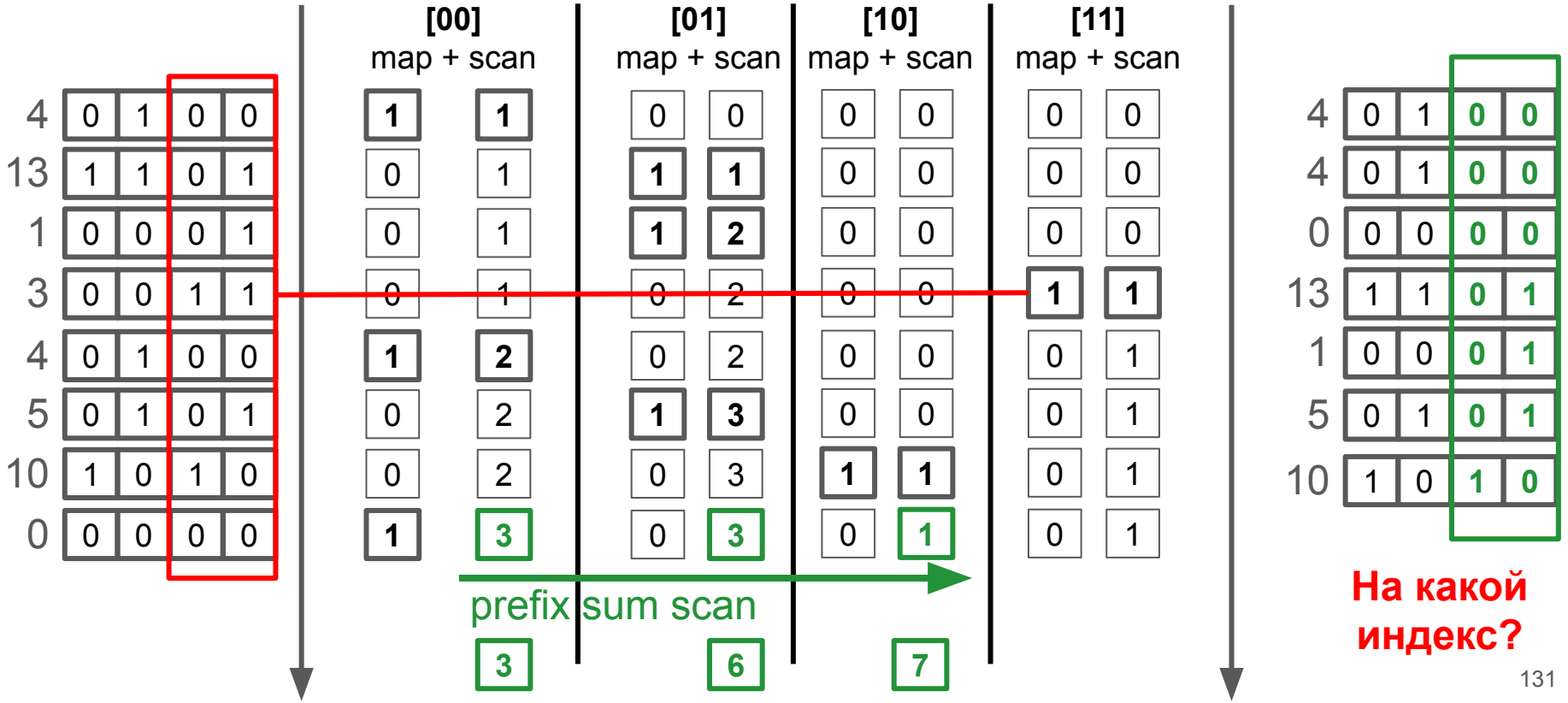
Radix sort: сортировка по **нескольким разрядам**



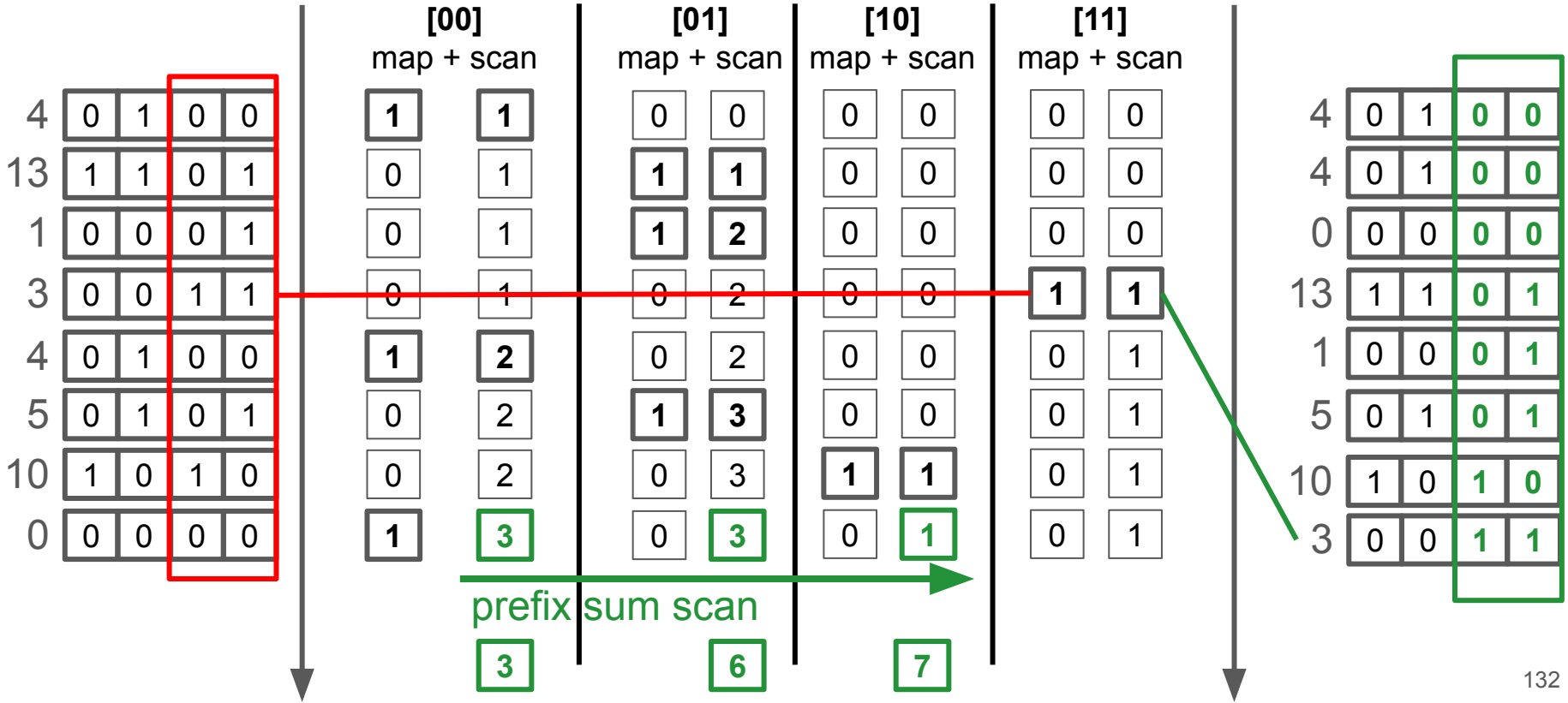
Radix sort: сортировка по **нескольким разрядам**



Radix sort: сортировка по **нескольким разрядам**



Radix sort: сортировка по **нескольким разрядам**



Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[???]***

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **???**

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **???**

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **???**

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем

   ???

   = ***global index*** на который мы кладем наше число

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем
   *сколько каждого случая в группе (**local offset**)*
   *+ смотрим сколько каждого случая до нашей группы (**global offset**)*
   = *global index* на который мы кладем наше число

Вопросы?

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем
   *сколько каждого случая в группе (**local offset**)*
   *+ смотрим сколько каждого случая до нашей группы (**global offset**)*
   = *global index* на который мы кладем наше число

Сколько проходов нужно?

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем
   *сколько каждого случая в группе (**local offset**)*
   *+ смотрим сколько каждого случая до нашей группы (**global offset**)*
   = *global index* на который мы кладем наше число

Можно ли сразу отсортировать по 8 бит? А по 32 бита?

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем
   *сколько каждого случая в группе (**local offset**)*
   *+ смотрим сколько каждого случая до нашей группы (**global offset**)*
   = *global index* на который мы кладем наше число

Как сортировать пары key (uint32) + value (большое)?

Radix sort: сортировка по **нескольким разрядам**

for ***sorted_bit_offset*** in ***[0, 4, 8, …, 28]***

1) **local counting**: в каждой группе считаем
   сколько каждого случая в группе
2) **global prefix scan**: префиксными суммами по группам находим
   сколько каждого случая до нашей группы
3) **global scater**: в каждой группе вновь считаем
   *сколько каждого случая в группе (**local offset**)*
   *+ смотрим сколько каждого случая до нашей группы (**global offset**)*
   = *global index* на который мы кладем наше число

Radix sort: результаты (проход - 4 бита)

```
Using device #1: API: CUDA+OpenCL+Vulkan. GPU. NVIDIA GeForce RTX 4090 (CUDA 12090). Free memory: 22994/24563 Mb.
Using CUDA API...
n=100000000 max_value=2147483647
sorting on CPU...
CPU std::sort finished in 7.59589 sec
CPU std::sort effective RAM bandwidth: 0.0980841 GB/s (13.1645 uint millions/s)
GPU radix-sort times (in seconds) - 10 values (min=0.0513104 10%=0.0517715 median=0.0521597 90%=0.176323 max=0.176323)
GPU radix-sort median effective VRAM bandwidth: 14.2842 GB/s (1917.19 uint millions/s)
```

```
Using device #0: API: CUDA+OpenCL+Vulkan. GPU. Tesla T4 (CUDA 12020). Free memory: 14822/14930 Mb.
Using CUDA API...
n=100000000 max_value=2147483647
sorting on CPU...
CPU std::sort finished in 11.0169 sec
CPU std::sort effective RAM bandwidth: 0.0676285 GB/s (9.07694 uint millions/s)
GPU radix-sort times (in seconds) - 10 values (min=0.222695 10%=0.222791 median=0.223857 90%=0.361715 max=0.361715)
GPU radix-sort median effective VRAM bandwidth: 3.32827 GB/s (446.713 uint millions/s)
```

Radix sort: результаты (проход - 4 бита)

```
Using device #1: API: CUDA+OpenCL+Vulkan. GPU. NVIDIA GeForce RTX 4090 (CUDA 12090). Free memory: 22994/24563 Mb.
Using CUDA API...
n=100000000 max_value=2147483647
sorting on CPU...
CPU std::sort finished in 7.59589 sec
CPU std::sort effective RAM bandwidth: 0.0980841 GB/s (13.1645 uint millions/s)
GPU radix-sort times (in seconds) - 10 values (min=0.0513104 10%=0.0517715 median=0.0521597 90%=0.176323 max=0.176323)
GPU radix-sort median effective VRAM bandwidth: 14.2842 GB/s (1917.19 uint millions/s)
    prefix sum median effective VRAM bandwidth: 40.3089 GB/s (5410.17 uint millions/s)
```

```
Using device #0: API: CUDA+OpenCL+Vulkan. GPU. Tesla T4 (CUDA 12020). Free memory: 14822/14930 Mb.
Using CUDA API...
n=100000000 max_value=2147483647
sorting on CPU...
CPU std::sort finished in 11.0169 sec
CPU std::sort effective RAM bandwidth: 0.0676285 GB/s (9.07694 uint millions/s)
GPU radix-sort times (in seconds) - 10 values (min=0.222695 10%=0.222791 median=0.223857 90%=0.361715 max=0.361715)
GPU radix-sort median effective VRAM bandwidth: 3.32827 GB/s (446.713 uint millions/s)
    prefix sum median effective VRAM bandwidth: 10.6824 GB/s (1433.76 uint millions/s)
```

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 5 values: [4 6 6 5 4]
```

```
__________________bits offset: 0_____________________________________________________
                      a_gpu: 5 values: [4 6 6 5 4]
                      a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
```

```
__________________bits offset: 0__________________________________________________________
                       a_gpu: 5 values: [4 6 6 5 4]
                       a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
```

```
__________________bits offset: 0__________________________________________________________
                       a_gpu: 5 values: [4 6 6 5 4]
                       a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
```

```
__________________bits offset: 0__________________________________________________________
                       a_gpu: 5 values: [4 6 6 5 4]
                       a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                       b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                       b_gpu: 5 values: [4 6 6 4 5]
```

```
__________________bits offset: 0____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
         per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                        b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                        b_gpu: 5 values: [4 6 6 4 5]
__________________bits offset: 1____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [01[0]0 01[1]0 01[1]0 01[0]1 01[0]0]
         per_chunk_counts_gpu: 2 values: [3 253]
per_chunk_counts_prefixes_gpu: 2 values: [3 253]
                        b_gpu: 5 values: [01[0]0 01[0]1 01[0]0 01[1]0 01[1]0]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 2____________________________________________________
rassert code=345214321 line=40
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
         per_chunk_counts_gpu: 2 values: [0 0]
per_chunk_counts_prefixes_gpu: 2 values: [0 0]
rassert code=345214321 line=64
                        b_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 3____________________________________________________
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
         per_chunk_counts_gpu: 2 values: [5 251]
per_chunk_counts_prefixes_gpu: 2 values: [5 251]
                        b_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
                        b_gpu: 5 values: [4 5 4 6 6]
```

```
__________________bits offset: 0____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                        b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                        b_gpu: 5 values: [4 6 6 4 5]
__________________bits offset: 1____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [01[0]0 01[1]0 01[1]0 01[0]1 01[0]0]
        per_chunk_counts_gpu: 2 values: [3 253]
per_chunk_counts_prefixes_gpu: 2 values: [3 253]
                        b_gpu: 5 values: [01[0]0 01[0]1 01[0]0 01[1]0 01[1]0]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 2____________________________________________________
rassert code=345214321 line=40
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
        per_chunk_counts_gpu: 2 values: [0 0]
per_chunk_counts_prefixes_gpu: 2 values: [0 0]
rassert code=345214321 line=64
                        b_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 3____________________________________________________
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
        per_chunk_counts_gpu: 2 values: [5 251]
per_chunk_counts_prefixes_gpu: 2 values: [5 251]
                        b_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
                        b_gpu: 5 values: [4 5 4 6 6]
radix sort times (in seconds) - 1 values (min=0.0139145 10%=0.0139145 median=0.0139145
GPU-radix sort median effective VRAM bandwidth: 2.67727e-06 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 4 5 1" failed at line 233
```

```
__________________bits offset: 0____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                        b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                        b_gpu: 5 values: [4 6 6 4 5]
__________________bits offset: 1____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [01[0]0 01[1]0 01[1]0 01[0]1 01[0]0]
        per_chunk_counts_gpu: 2 values: [3 253]
per_chunk_counts_prefixes_gpu: 2 values: [3 253]
                        b_gpu: 5 values: [01[0]0 01[0]1 01[0]0 01[1]0 01[1]0]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 2____________________________________________________
rassert code=345214321 line=40
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
        per_chunk_counts_gpu: 2 values: [0 0]
per_chunk_counts_prefixes_gpu: 2 values: [0 0]
rassert code=345214321 line=64
                        b_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 3____________________________________________________
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
        per_chunk_counts_gpu: 2 values: [5 251]
per_chunk_counts_prefixes_gpu: 2 values: [5 251]
                        b_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
                        b_gpu: 5 values: [4 5 4 6 6]
radix sort times (in seconds) - 1 values (min=0.0139145 10%=0.0139145 median=0.0139145
GPU-radix sort median effective VRAM bandwidth: 2.67727e-06 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 4 5 1" failed at line 233
```

```
__________________bits offset: 0____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                        b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                        b_gpu: 5 values: [4 6 6 4 5]
__________________bits offset: 1____________________________________________________
                        a_gpu: 5 values: [4 6 6 5 4]
                        a_gpu: 5 values: [01[0]0 01[1]0 01[0]1 01[0]0]
        per_chunk_counts_gpu: 2 values: [3 253]
per_chunk_counts_prefixes_gpu: 2 values: [3 253]
                        b_gpu: 5 values: [01[0]0 01[0]1 01[0]0 01[1]0 01[1]0]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 2____________________________________________________
rassert code=345214321 line=40
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
        per_chunk_counts_gpu: 2 values: [0 0]
per_chunk_counts_prefixes_gpu: 2 values: [0 0]
rassert code=345214321 line=64
                        b_gpu: 5 values: [0[1]00 0[1]01 0[1]00 0[1]10 0[1]10]
                        b_gpu: 5 values: [4 5 4 6 6]
__________________bits offset: 3____________________________________________________
                        a_gpu: 5 values: [4 5 4 6 6]
                        a_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
        per_chunk_counts_gpu: 2 values: [5 251]
per_chunk_counts_prefixes_gpu: 2 values: [5 251]
                        b_gpu: 5 values: [[0]100 [0]101 [0]100 [0]110 [0]110]
                        b_gpu: 5 values: [4 5 4 6 6]
radix sort times (in seconds) - 1 values (min=0.0139145 10%=0.0139145 median=0.0139145
GPU-radix sort median effective VRAM bandwidth: 2.67727e-06 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 4 5 1" failed at line 233
```

```
__________________bits offset: 0____________________________________________________
                       a_gpu: 5 values: [4 6 6 5 4]
                       a_gpu: 5 values: [010[0] 011[0] 011[0] 010[1] 010[0]]
        per_chunk_counts_gpu: 2 values: [4 252]
per_chunk_counts_prefixes_gpu: 2 values: [4 252]
                       b_gpu: 5 values: [010[0] 011[0] 011[0] 010[0] 010[1]]
                       b_gpu: 5 values: [4 6 6 4 5]
__________________bits offset: 1____________________________________________________
                       a_gpu: 5 values: [4 6 6 4 5]
                       a_gpu: 5 values: [01[0]0 01[1]0 01[1]0 01[0]0 01[0]1]
        per_chunk_counts_gpu: 2 values: [3 253]
per_chunk_counts_prefixes_gpu: 2 values: [3 253]
                       b_gpu: 5 values: [01[0]0 01[0]0 01[0]1 01[1]0 01[1]0]
                       b_gpu: 5 values: [4 4 5 6 6]
__________________bits offset: 2____________________________________________________
rassert code=345214321 line=40
                       a_gpu: 5 values: [4 4 5 6 6]
                       a_gpu: 5 values: [0[1]00 0[1]00 0[1]01 0[1]10 0[1]10]
        per_chunk_counts_gpu: 2 values: [0 0]
per_chunk_counts_prefixes_gpu: 2 values: [0 0]
rassert code=345214321 line=64
                       b_gpu: 5 values: [0[1]00 0[1]00 0[1]01 0[1]10 0[1]10]
                       b_gpu: 5 values: [4 4 5 6 6]
__________________bits offset: 3____________________________________________________
                       a_gpu: 5 values: [4 4 5 6 6]
                       a_gpu: 5 values: [[0]100 [0]100 [0]101 [0]110 [0]110]
        per_chunk_counts_gpu: 2 values: [5 251]
per_chunk_counts_prefixes_gpu: 2 values: [5 251]
                       b_gpu: 5 values: [[0]100 [0]100 [0]101 [0]110 [0]110]
                       b_gpu: 5 values: [4 4 5 6 6]
```

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
```

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
        per_chunk_counts_gpu: 4 values: [2 0 4 250]
```

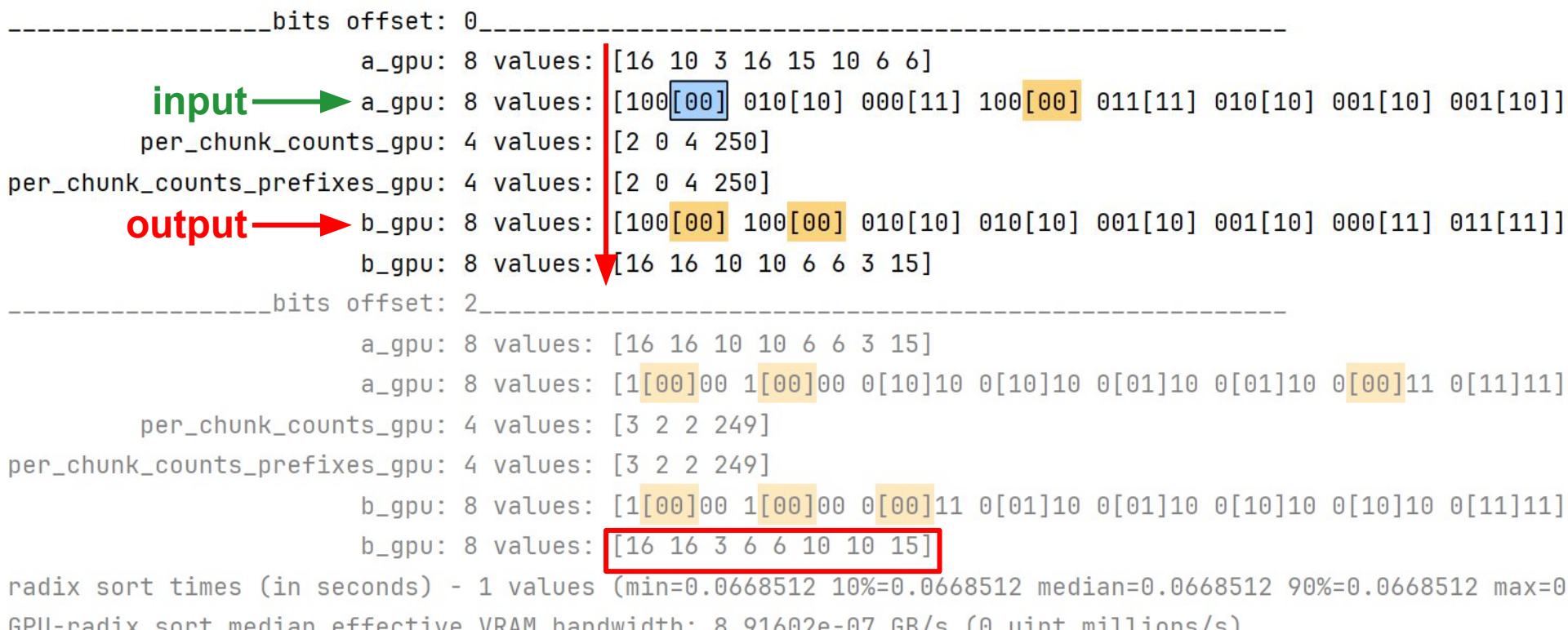
```
__________________bits offset: 0__________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
         per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
```

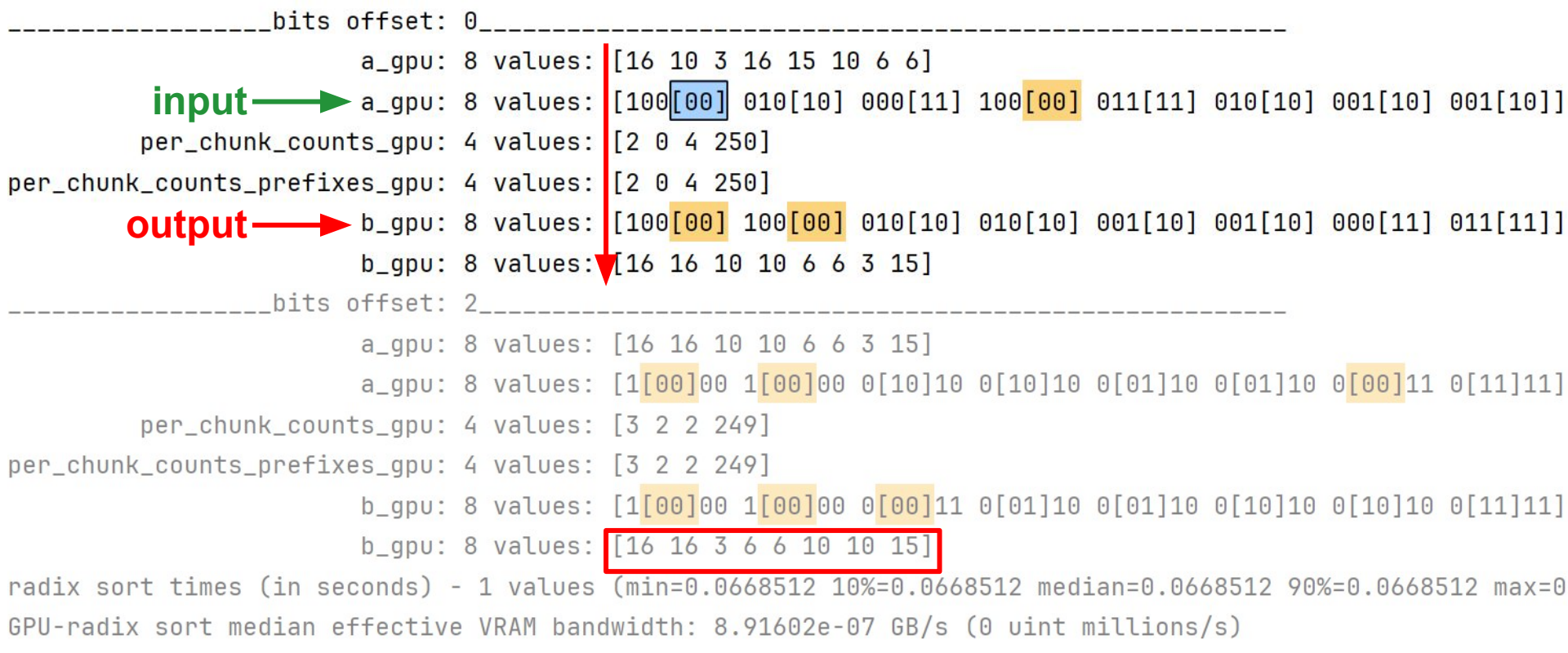
```
__________________bits offset: 0_____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
         per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                      b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                      b_gpu: 8 values: [16 16 10 10 6 6 3 15]
```

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
         per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                      b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                      b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________
                      a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                      a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
         per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                      b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                      b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```

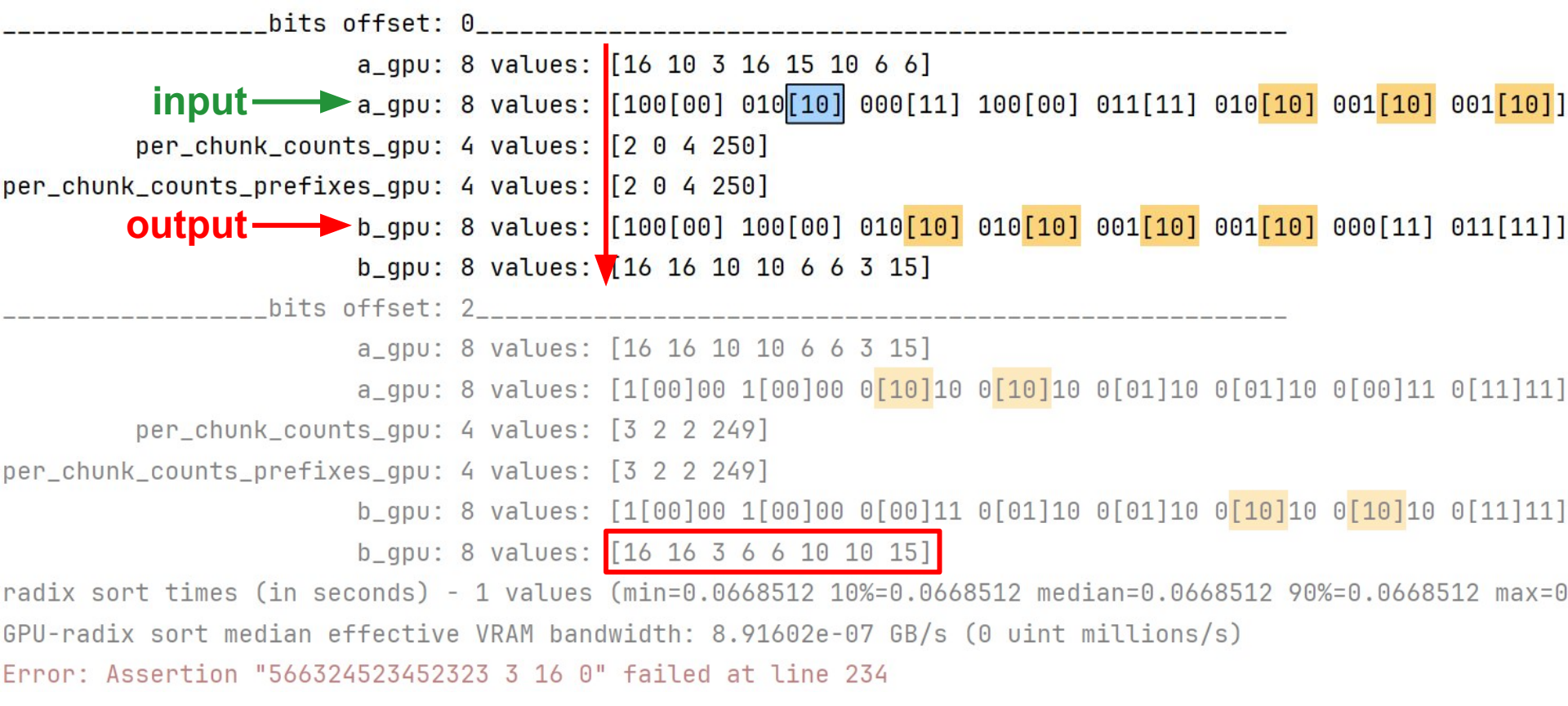
```
__________________bits offset: 0____________________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
input ──────────────► a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
          per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
output ─────────────► b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                      b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________________
                      a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                      a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
          per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                      b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                      b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```

Как быстро проверить?

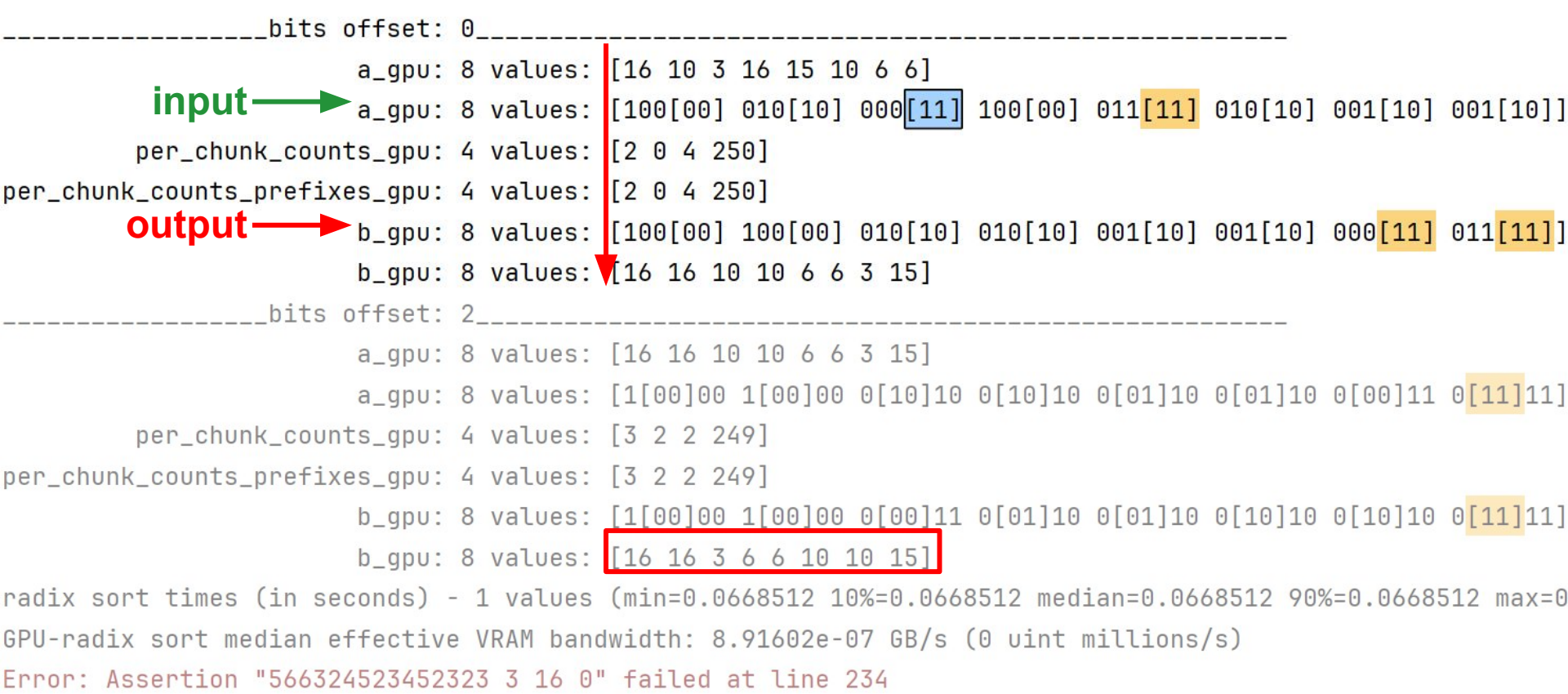

_____bits offset: 0_____
a_gpu: 8 values: [16 10 3 16 15 10 6 6]
input
a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
output
b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____bits offset: 2_____
a_gpu: 8 values: [16 16 10 10 6 6 3 15]
a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234



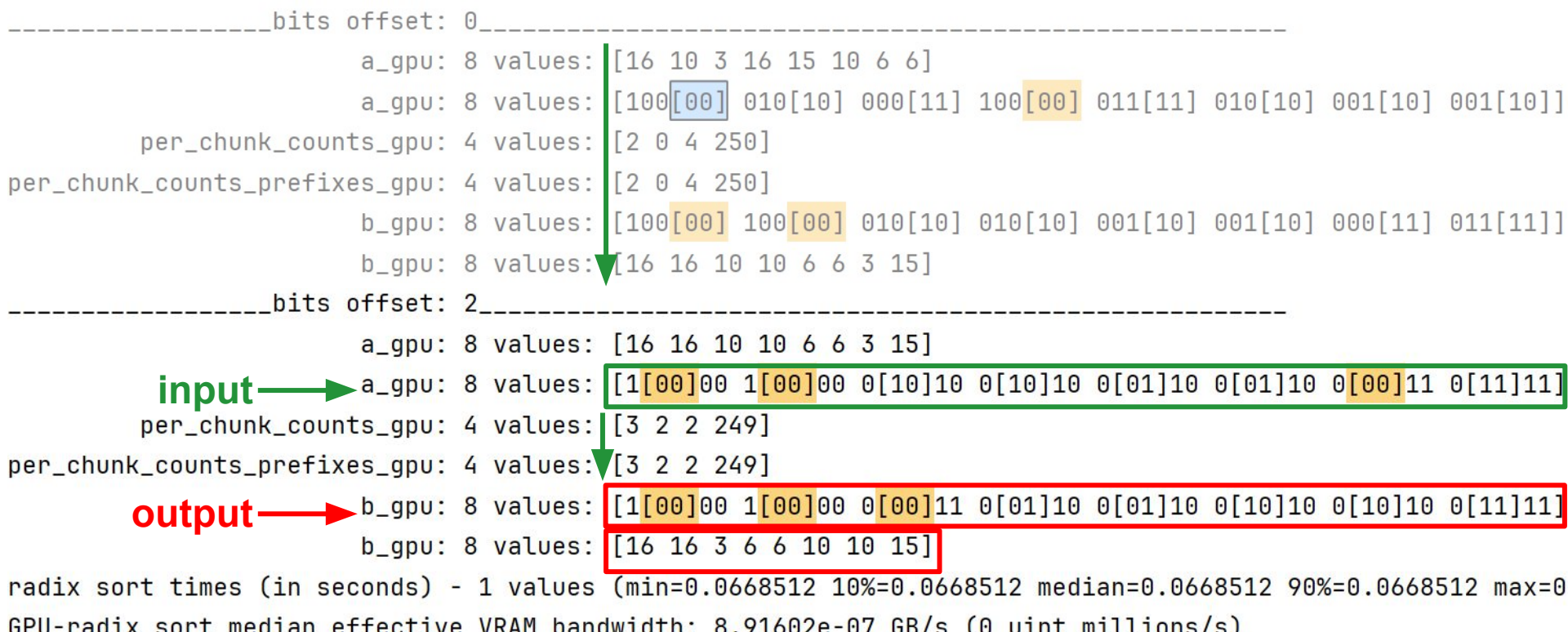

```
__________________bits offset: 0____________________________________________________
                     a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                     a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
          per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                     b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                     b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________
                     a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                     a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
          per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                     b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                     b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```



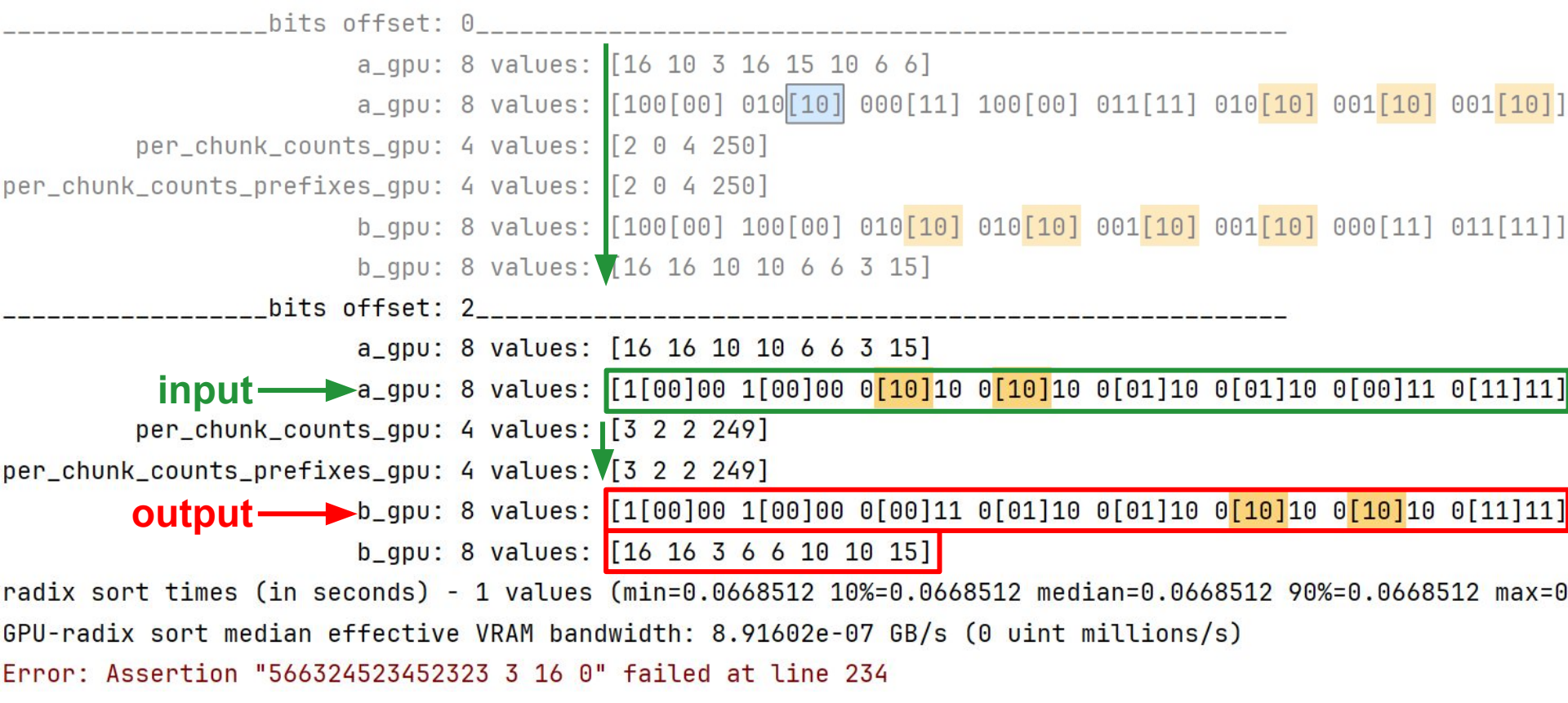

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
        per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                      b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                      b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________
                      a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                      a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
        per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                      b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                      b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```

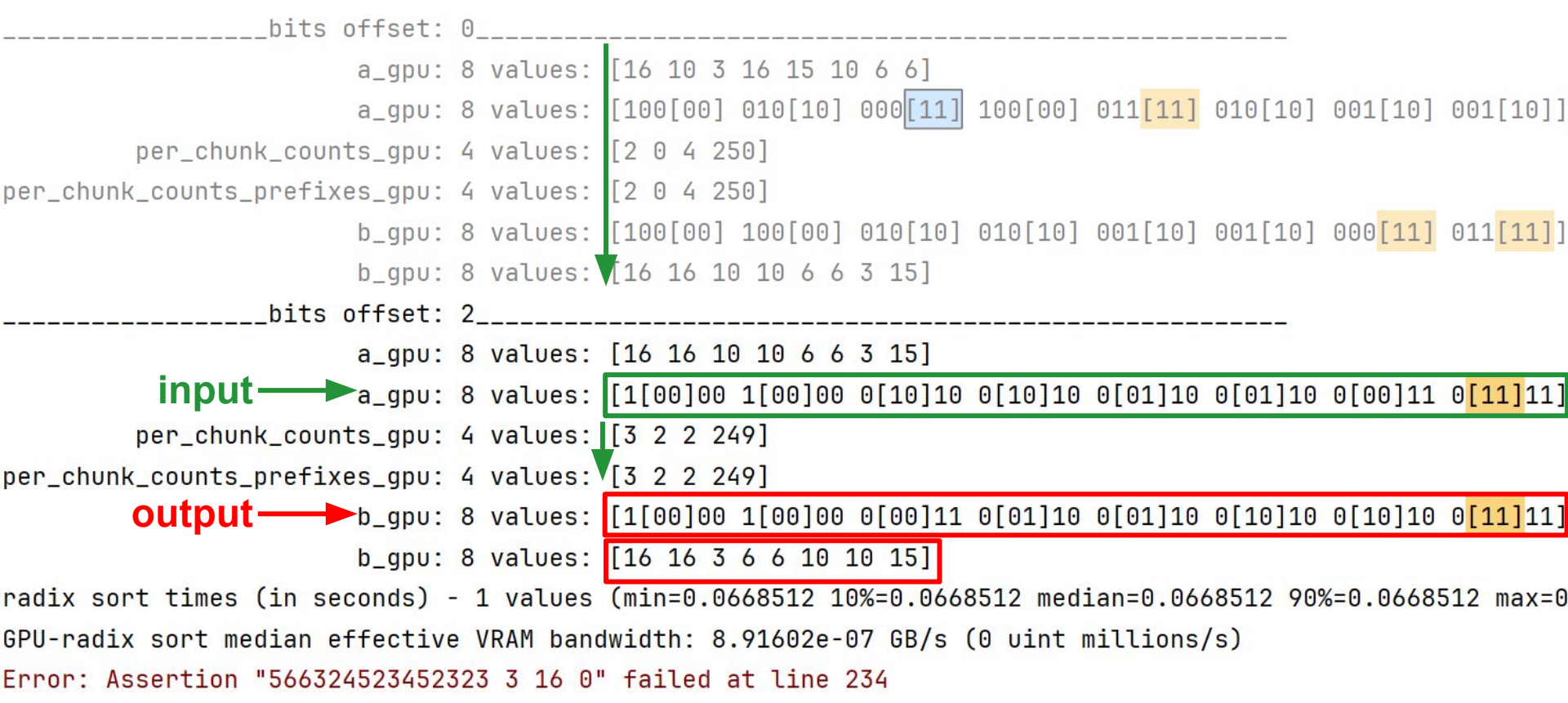


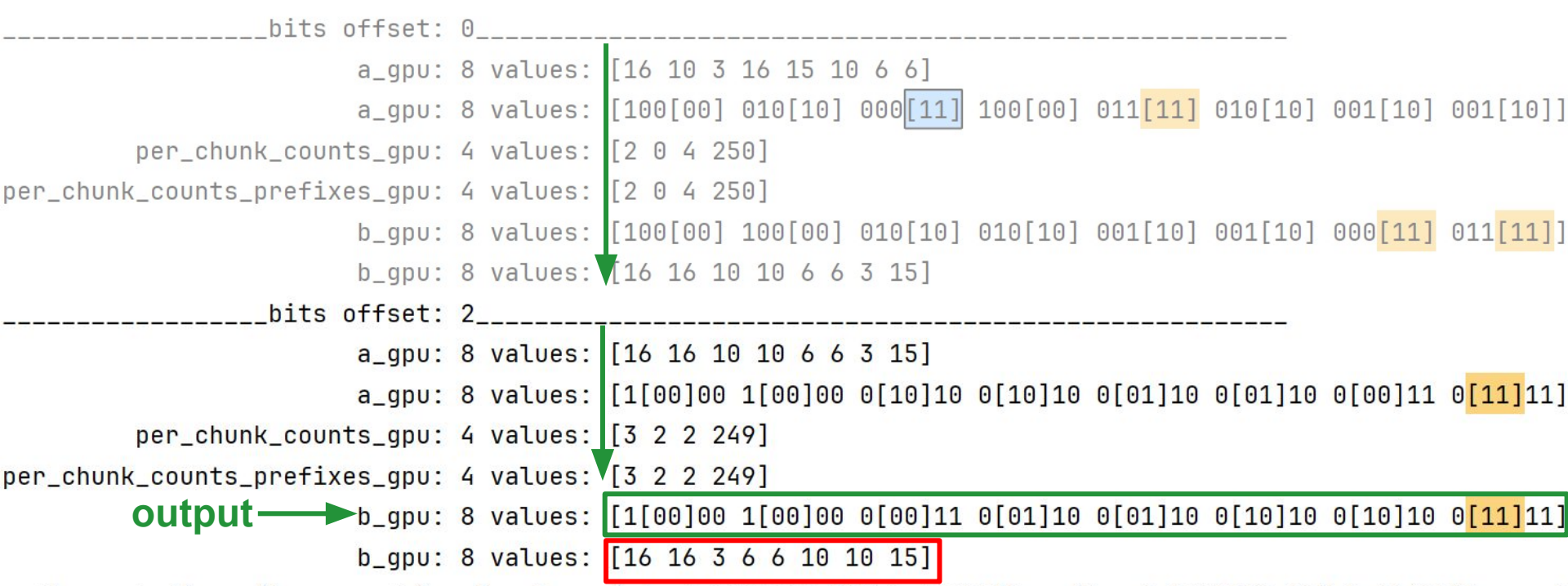

```
_____________bits offset: 0_____________________________________________________
                  a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                  a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
        per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                  b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                  b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____________bits offset: 2_____________________________________________________
                  a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                  a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
        per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                  b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                  b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```


_____bits offset: 0_____
a_gpu: 8 values: [16 10 3 16 15 10 6 6]
a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____bits offset: 2_____
a_gpu: 8 values: [16 16 10 10 6 6 3 15]
input
a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
output
b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234

```
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```


_____bits offset: 0_____
a_gpu: 8 values: [16 10 3 16 15 10 6 6]
a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____bits offset: 2_____
a_gpu: 8 values: [16 16 10 10 6 6 3 15]
input
a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
output
b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234


_____bits offset: 0_____
a_gpu: 8 values: [16 10 3 16 15 10 6 6]
a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____bits offset: 2_____
a_gpu: 8 values: [16 16 10 10 6 6 3 15]
input
a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
output
b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234


_____bits offset: 0_____
a_gpu: 8 values: [16 10 3 16 15 10 6 6]
a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
b_gpu: 8 values: [16 16 10 10 6 6 3 15]
_____bits offset: 2_____
a_gpu: 8 values: [16 16 10 10 6 6 3 15]
a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
output
b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234

```
__________________bits offset: 0____________________________________________________________
                    a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                    a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
          per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                    b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                    b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________________
                    a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                    a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
          per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
output ->           b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                    b_gpu: 8 values: [16 16 3 6 6 10 10 15]
radix sort times (in seconds) - 1 values (min=0.0668512 10%=0.0668512 median=0.0668512 90%=0.0668512 max=0
GPU-radix sort median effective VRAM bandwidth: 8.91602e-07 GB/s (0 uint millions/s)
Error: Assertion "566324523452323 3 16 0" failed at line 234
```

В чем же бага?

```
__________________bits offset: 0____________________________________________________
                      a_gpu: 8 values: [16 10 3 16 15 10 6 6]
                      a_gpu: 8 values: [100[00] 010[10] 000[11] 100[00] 011[11] 010[10] 001[10] 001[10]]
         per_chunk_counts_gpu: 4 values: [2 0 4 250]
per_chunk_counts_prefixes_gpu: 4 values: [2 0 4 250]
                      b_gpu: 8 values: [100[00] 100[00] 010[10] 010[10] 001[10] 001[10] 000[11] 011[11]]
                      b_gpu: 8 values: [16 16 10 10 6 6 3 15]
__________________bits offset: 2____________________________________________________
                      a_gpu: 8 values: [16 16 10 10 6 6 3 15]
                      a_gpu: 8 values: [1[00]00 1[00]00 0[10]10 0[10]10 0[01]10 0[01]10 0[00]11 0[11]11]
         per_chunk_counts_gpu: 4 values: [3 2 2 249]
per_chunk_counts_prefixes_gpu: 4 values: [3 2 2 249]
                      b_gpu: 8 values: [1[00]00 1[00]00 0[00]11 0[01]10 0[01]10 0[10]10 0[10]10 0[11]11]
                      b_gpu: 8 values: [16 16 3 6 6 10 10 15]
__________________bits offset: 4____________________________________________________
                      a_gpu: 8 values: [16 16 3 6 6 10 10 15]
                      a_gpu: 8 values: [[1]0000 [1]0000 [0]0011 [0]0110 [0]0110 [0]1010 [0]1010 [0]1111]
         per_chunk_counts_gpu: 4 values: [6 2 0 248]
per_chunk_counts_prefixes_gpu: 4 values: [6 2 0 248]
                      b_gpu: 8 values: [[0]0011 [0]0110 [0]0110 [0]1010 [0]1010 [0]1111 [1]0000 [1]0000]
                      b_gpu: 8 values: [3 6 6 10 10 15 16 16]
```

Усложнил случай - теперь **две** рабочие группы **по 32**

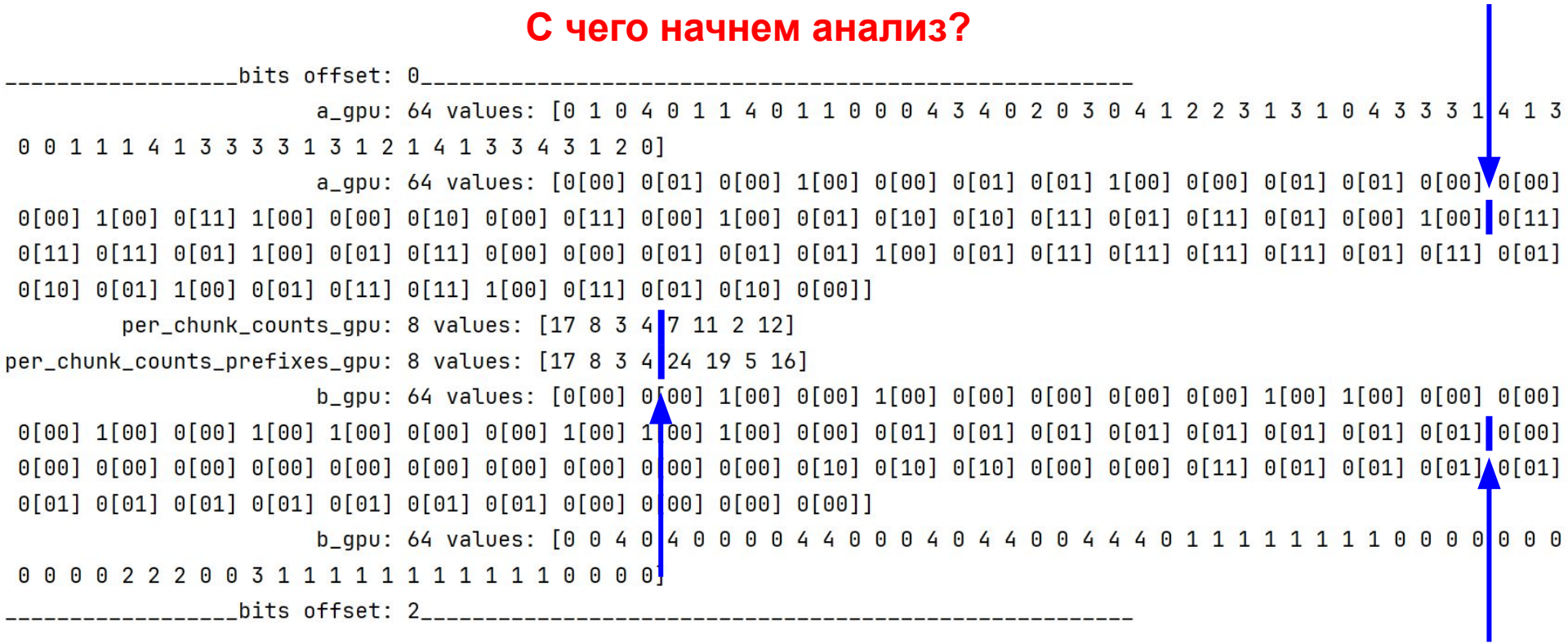
Усложнил случай - теперь **две** рабочие группы **по 32**

С чего начнем анализ?

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
        per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                        b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

Усложним случай - теперь **две** рабочие группы **по 32**

С чего начнем анализ?



Усложнил случай - теперь **две** рабочие группы **по 32**

```
__________________bits offset: 0____________________________________________________________
                       a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                       a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                       b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                       b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

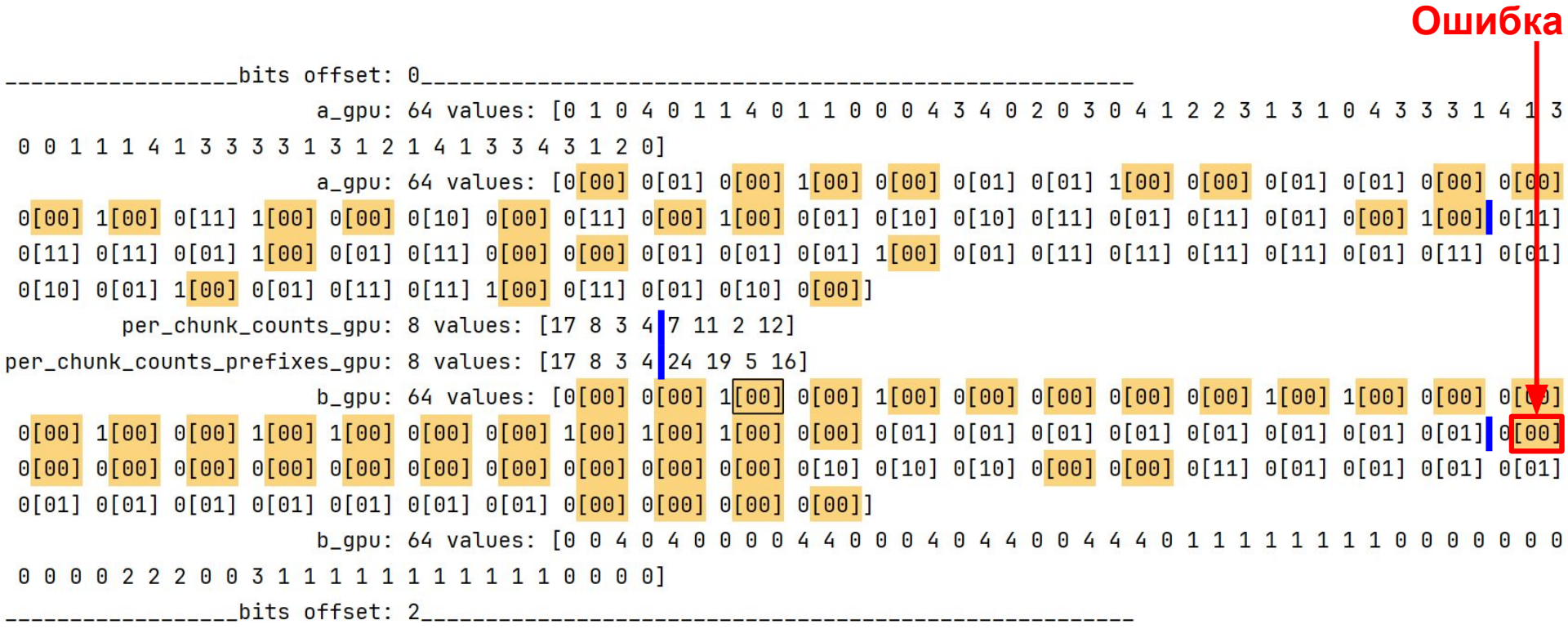
Где мы ожидаем видеть [00]?

Усложнил случай - теперь **две** рабочие группы **по 32**

```
__________________bits offset: 0____________________________________________________________
                       a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                       a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                       b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                       b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

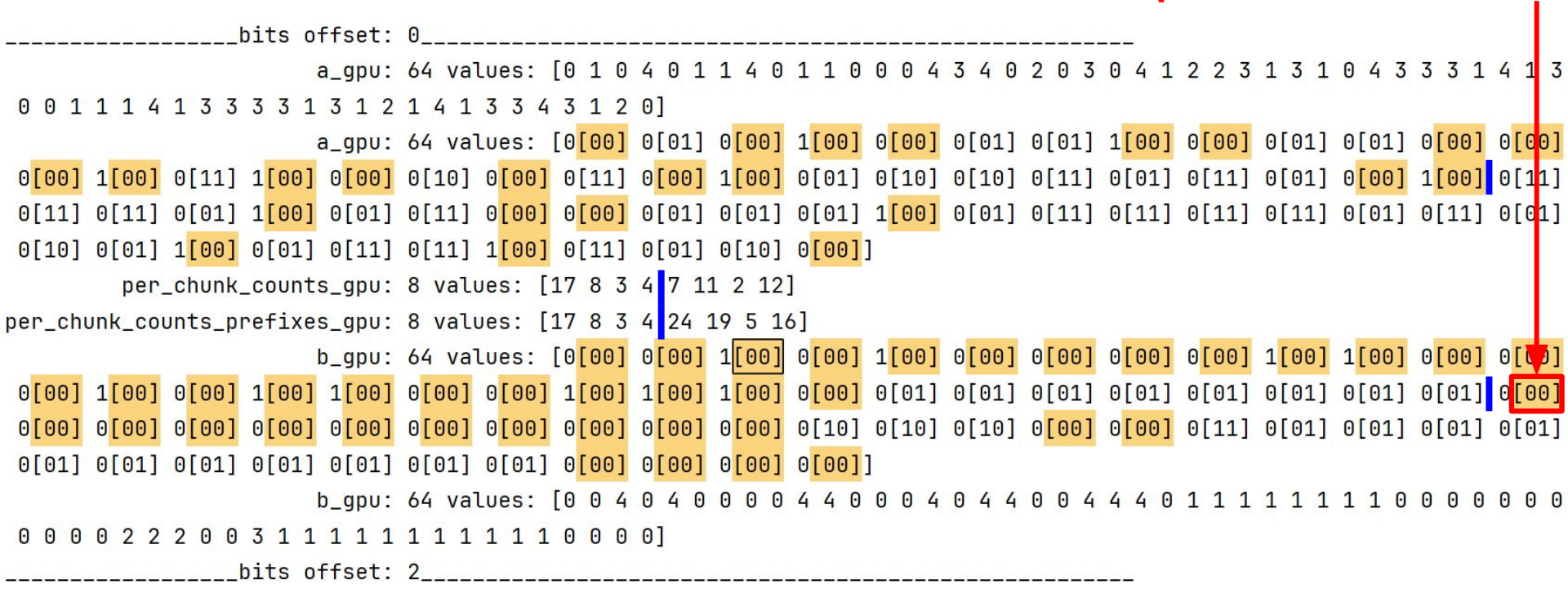
Все ли верно?

Усложнил случай - теперь **две** рабочие группы **по 32**



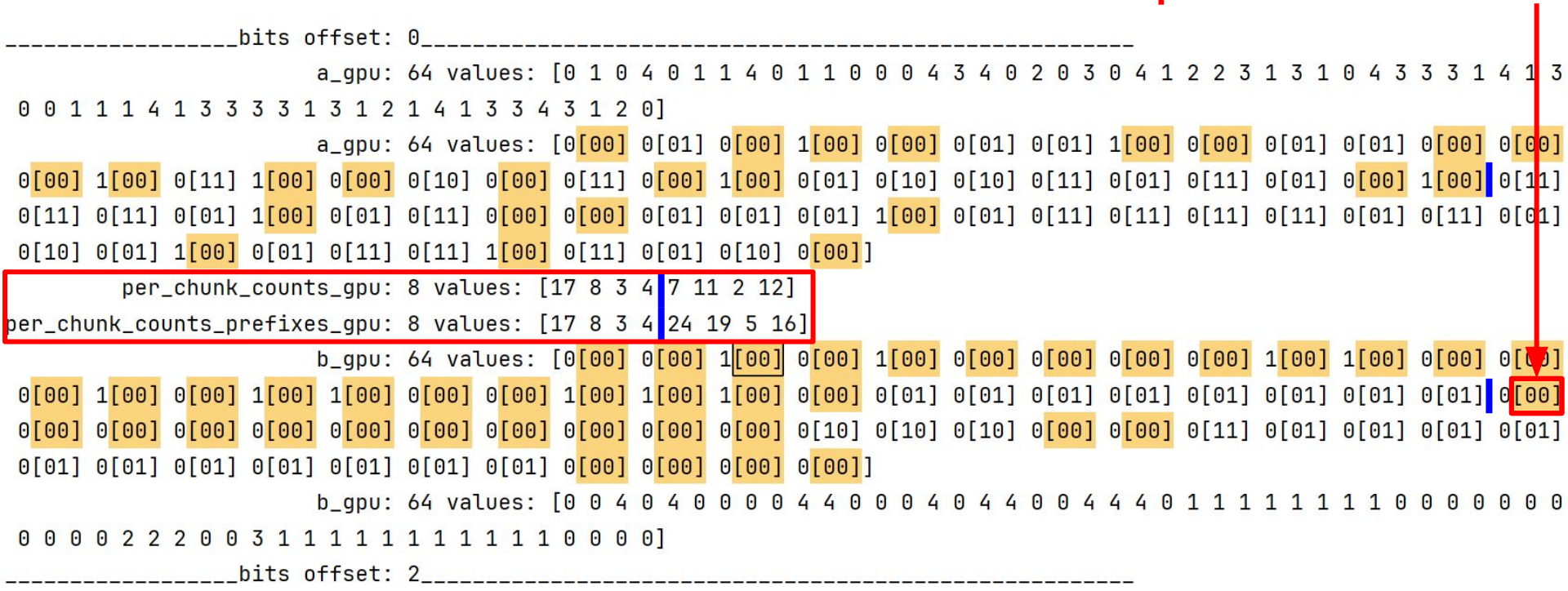
Усложнил случай - теперь **две** рабочие группы **по 32**

Первая ли это ошибка?



Усложнил случай - теперь **две** рабочие группы **по 32**

Первая ли это ошибка?



Усложнил случай - теперь **две** рабочие группы **по 32**

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                        b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

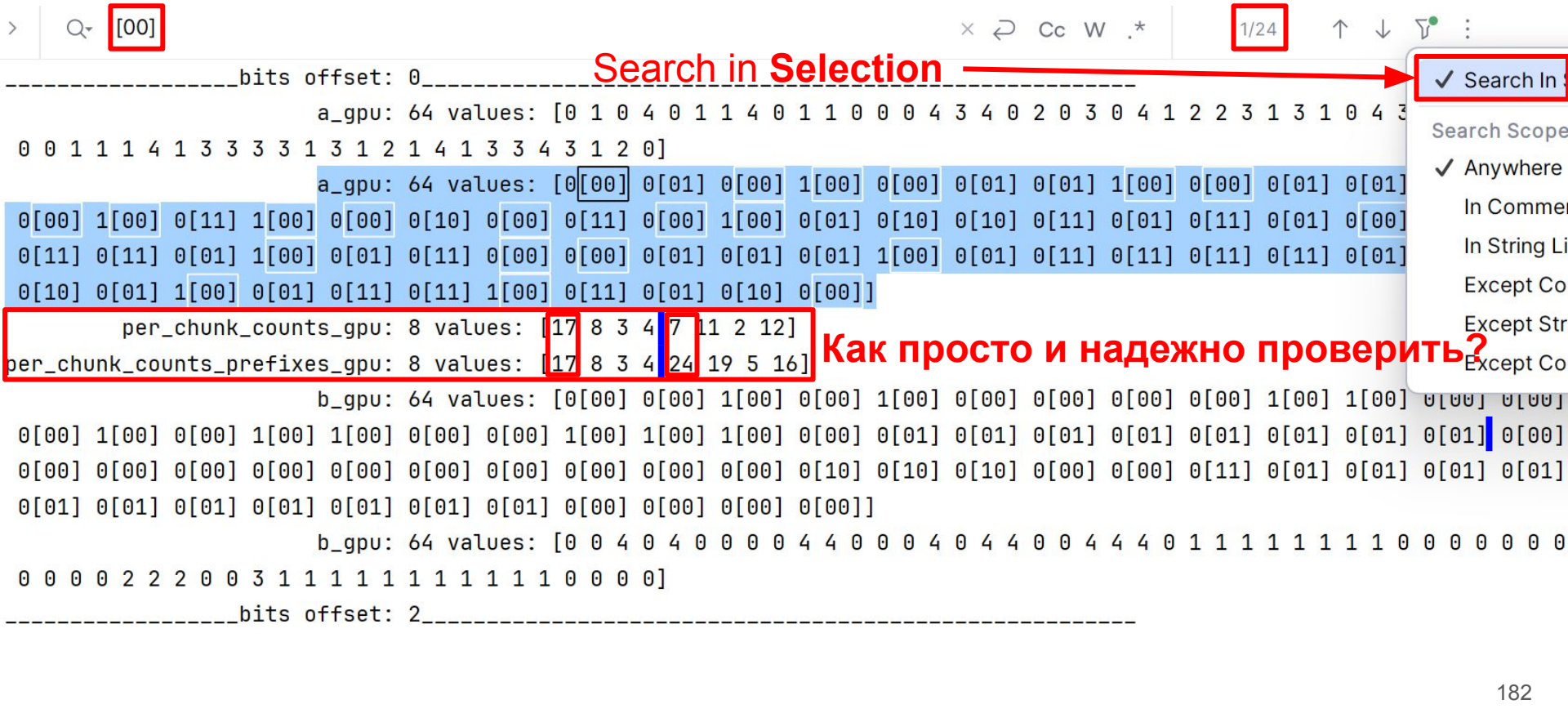
Как просто и надежно проверить?

Усложнил случай - теперь **две** рабочие группы **по 32**

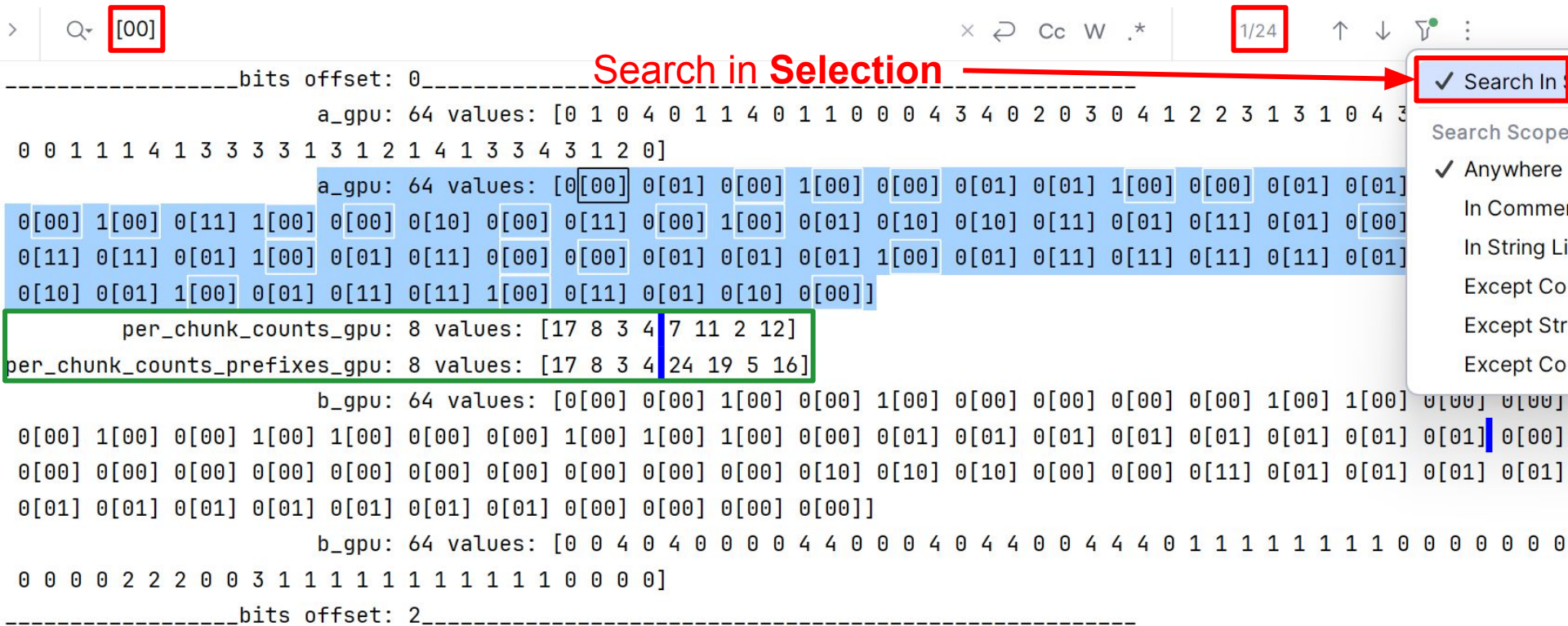
```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                        b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

Как просто и надежно проверить?

Усложнил случай - теперь **две** рабочие группы **по 32**



Усложнил случай - теперь **две** рабочие группы **по 32**

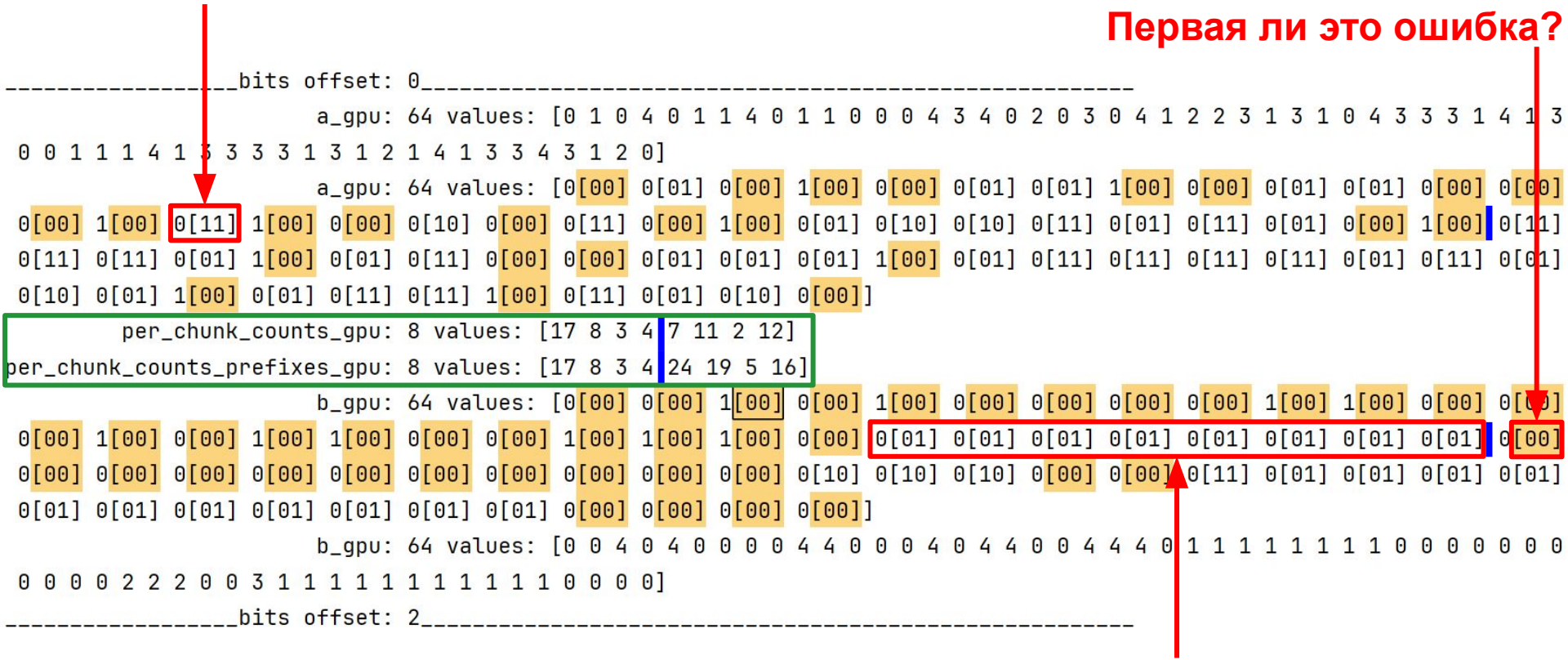


Усложнил случай - теперь **две** рабочие группы **по 32**

Первая ли это ошибка?

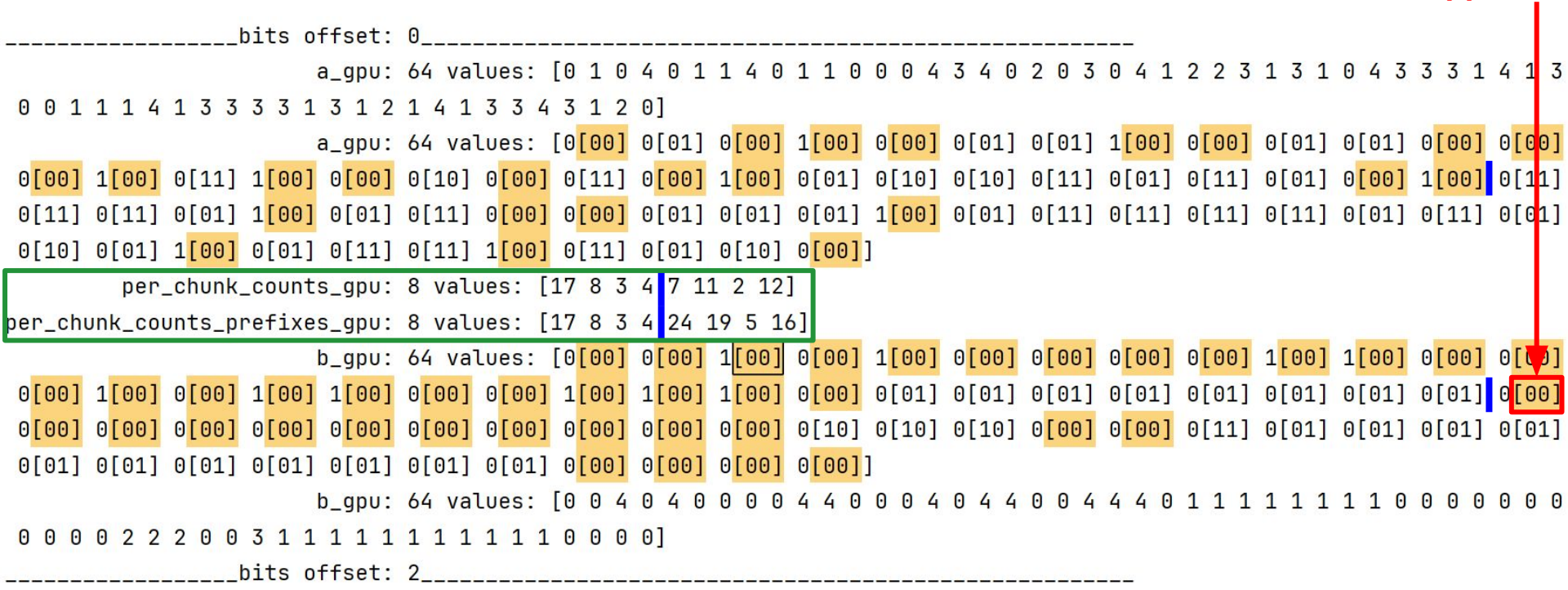
```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
                        b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 2 2 2 0 0 3 1 1 1 1 1 1 1 1 1 1 0 0 0 0]
__________________bits offset: 2____________________________________________________________
```

Усложнил случай - теперь **две** рабочие группы **по 32**



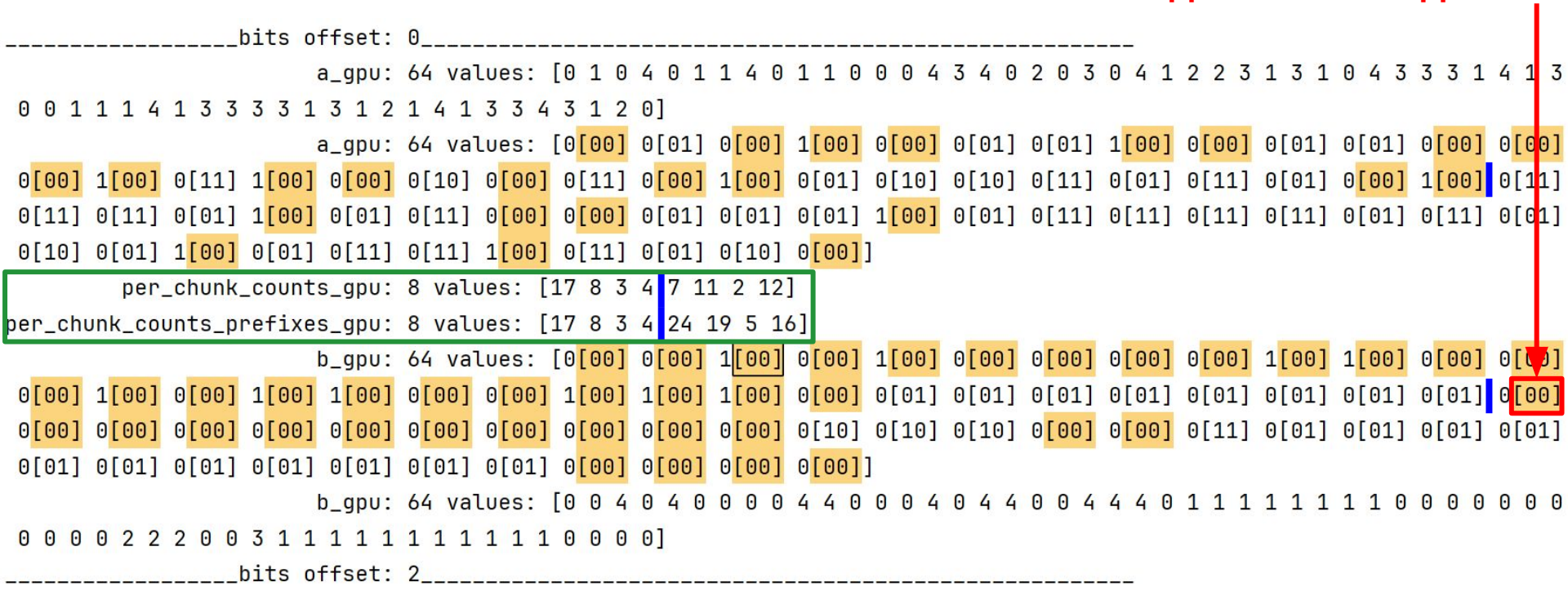
Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить?



Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить? Его индекс 32



Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить? Его индекс 32

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
```

```
output_values[output_index] = value;
```

Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить? Его индекс 32

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
```

```
unsigned int output_index = output_indices[i * WG_SIZE + local_id];
if (output_index == 32) printf("TEST239 lid=%d wg=%d ???\n", threadIdx.x, blockIdx.x);
output_values[output_index] = value;
```

Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить? Его индекс 32

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
```

```
unsigned int output_index = output_indices[i * WG_SIZE + local_id];
if (output_index == 32) printf("TEST239 lid=%d wg=%d ???\n", threadIdx.x, blockIdx.x);
output_values[output_index] = value;
```

Сообщение не выводится! Что делать?

Усложнил случай - теперь **две** рабочие группы **по 32**

Как его отладить? Его индекс 32

```
__________________bits offset: 0____________________________________________________________
                        a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                        a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                        b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00]
 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[00] 0[10] 0[10] 0[10] 0[00] 0[00] 0[11] 0[01] 0[01] 0[01] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[00] 0[00] 0[00] 0[00]]
```

```
printf("TEST239 output_index=%d value=%d\n", output_index, value);
output_values[output_index] = value;
```

Выведем все случаи этой строки!

Усложнил случай - теперь **две** рабочие группы **по 32**

TEST239 Cc W .* 1/50

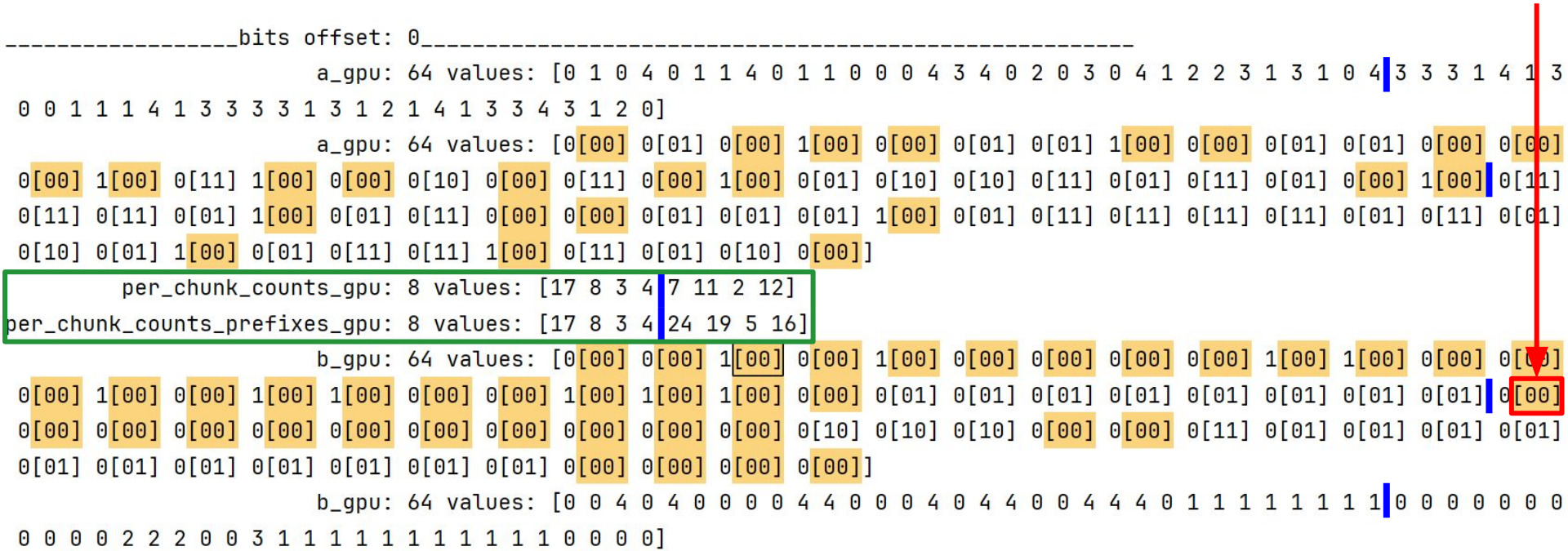
```
TEST239 output_index=49 value=1
TEST239 output_index=17 value=4
TEST239 output_index=50 value=1
TEST239 output_index=18 value=0
TEST239 output_index=19 value=0
TEST239 output_index=51 value=1
TEST239 output_index=52 value=1
TEST239 output_index=53 value=1
TEST239 output_index=20 value=4
TEST239 output_index=54 value=1
TEST239 output_index=55 value=1
TEST239 output_index=56 value=1
TEST239 output_index=57 value=1
TEST239 output_index=21 value=4
TEST239 output_index=58 value=1
TEST239 output_index=22 value=4
TEST239 output_index=59 value=1
```

- 50 раз вывелось
- Индекс 32 действительно отсутствует

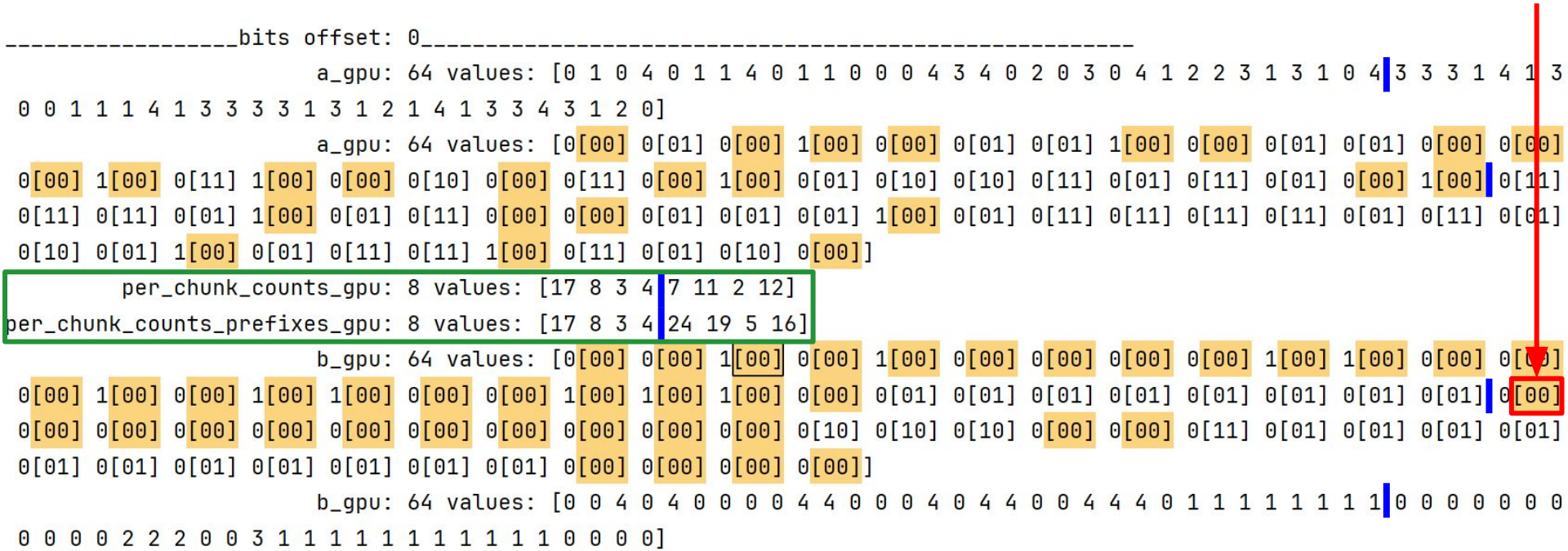
```
printf("TEST239 output_index=%d value=%d\n", output_index, value);
output_values[output_index] = value;
```

Выведем все случаи этой строки!

Усложнил случай - теперь **две** рабочие группы **по 32**

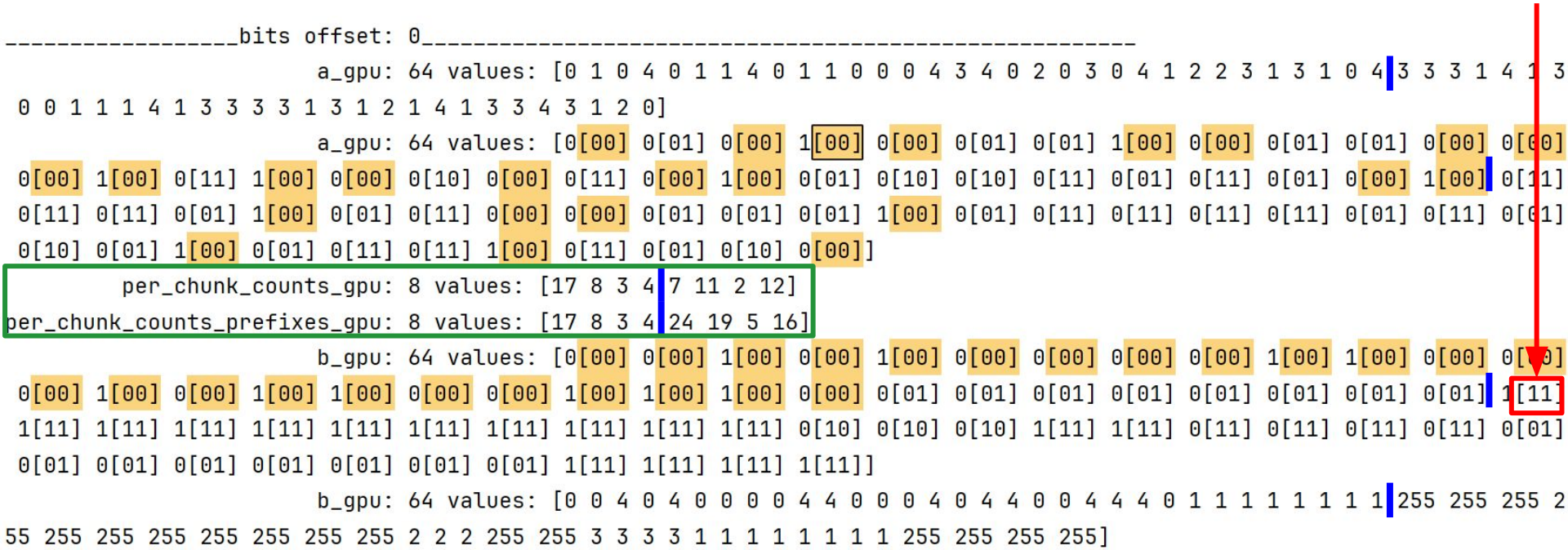


Усложнил случай - теперь **две** рабочие группы **по 32**



Обязательно инициализируйте буферы! Хотя бы нулями.
Лучше - уникальным числом-маркером, например **255**.

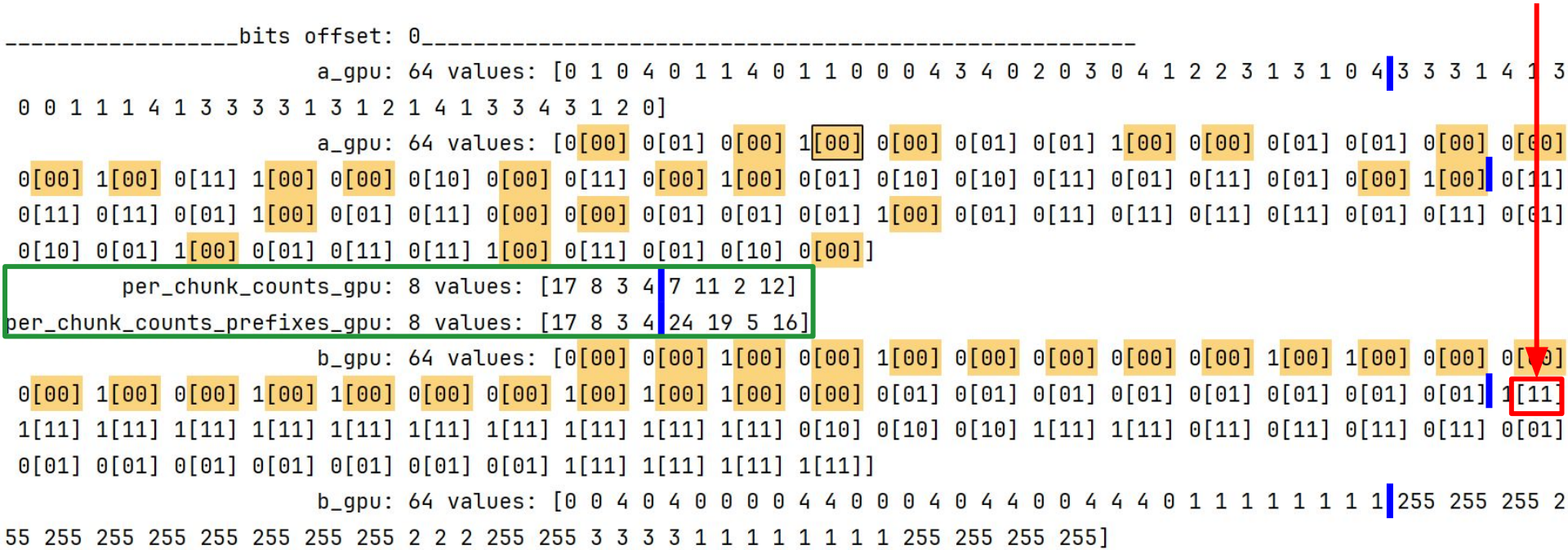
Усложнил случай - теперь **две** рабочие группы **по 32**



Обязательно инициализируйте буферы! Хотя бы нулями.
Лучше - уникальным числом-маркером, например **255**.

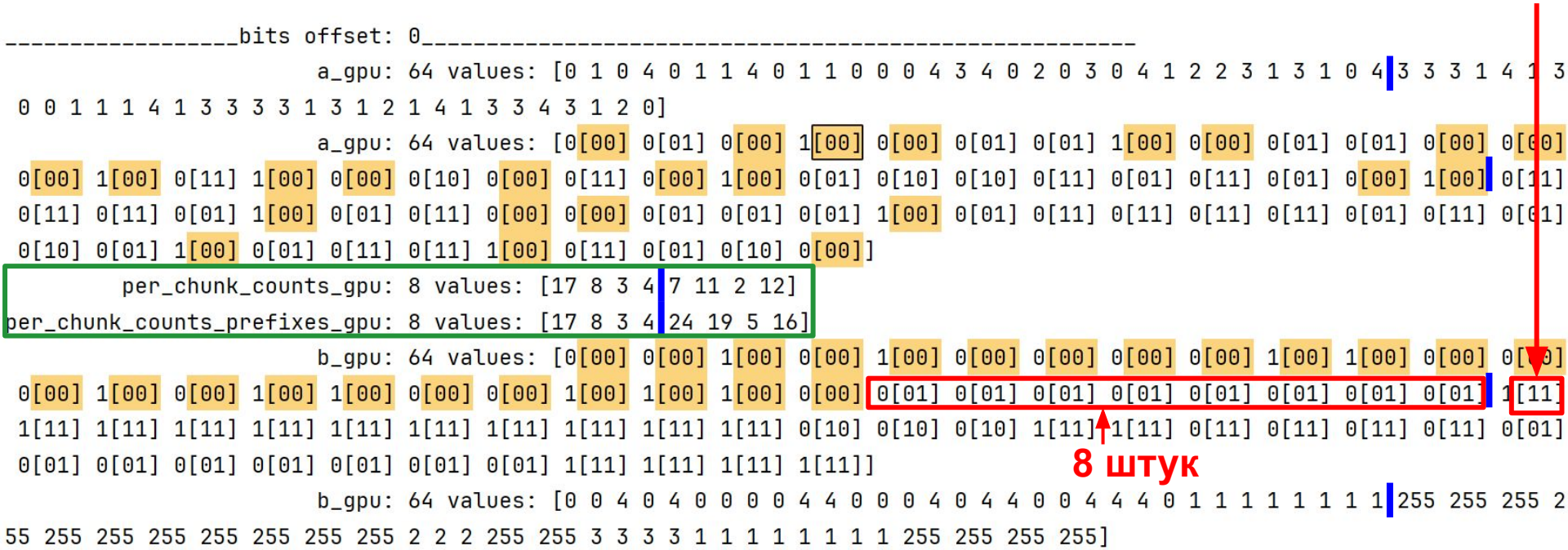
Усложнил случай - теперь **две** рабочие группы **по 32**

Почему сюда ничего не записалось? Кто должен был записать?



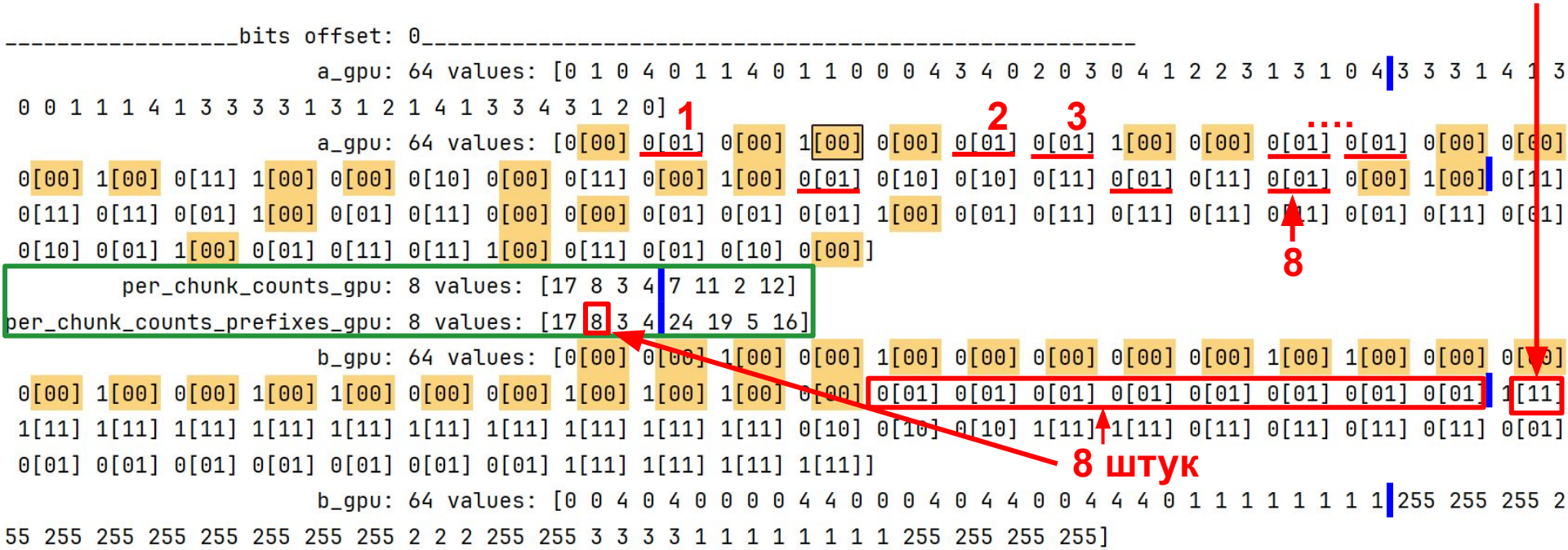
Усложнил случай - теперь **две** рабочие группы **по 32**

Почему сюда ничего не записалось? Кто должен был записать?



Усложнил случай - теперь **две** рабочие группы **по 32**

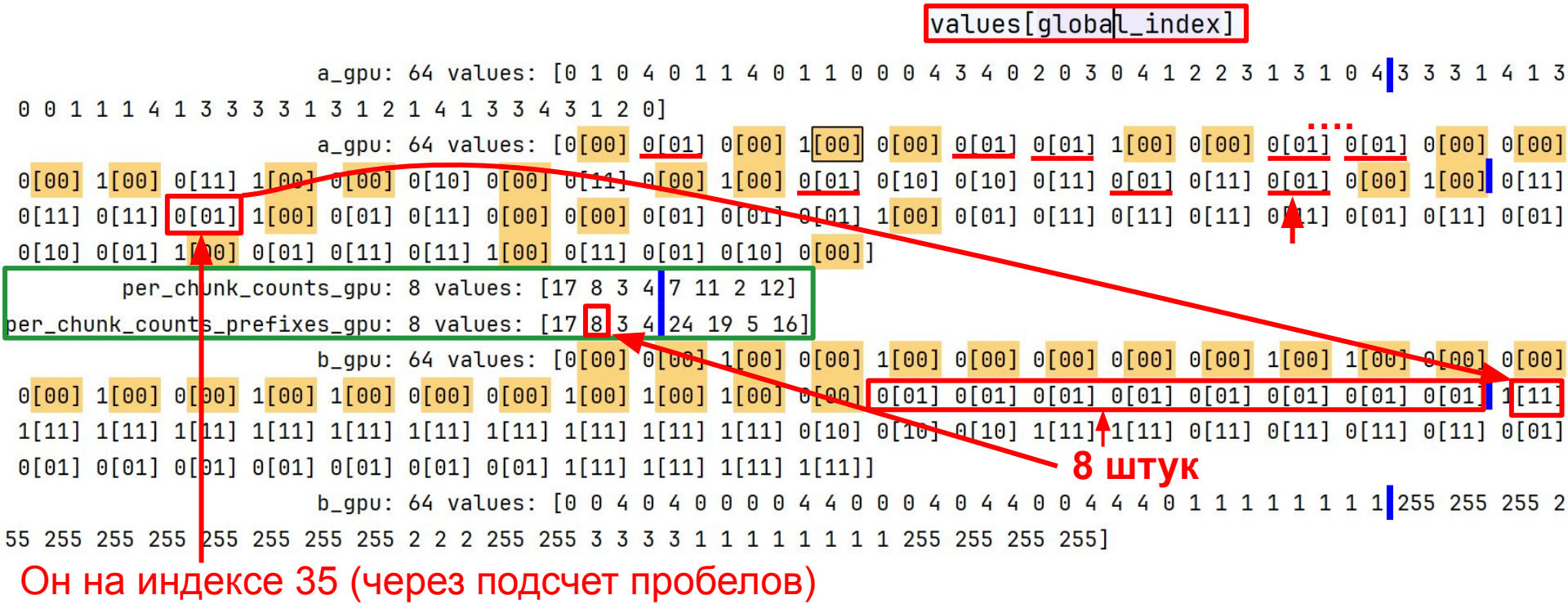
Почему сюда ничего не записалось? Кто должен был записать?



Усложнил случай - теперь **две** рабочие группы **по 32**

Усложнил случай - теперь **две** рабочие группы **по 32**

Чисто технически через поиск находим где считывание из входного массива:



Усложнил случай - теперь **две** рабочие группы **по 32**

```
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;
```

```
                    a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                    a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
        per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                    b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
                    b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 255 255 255 2
55 255 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
```

8 штук

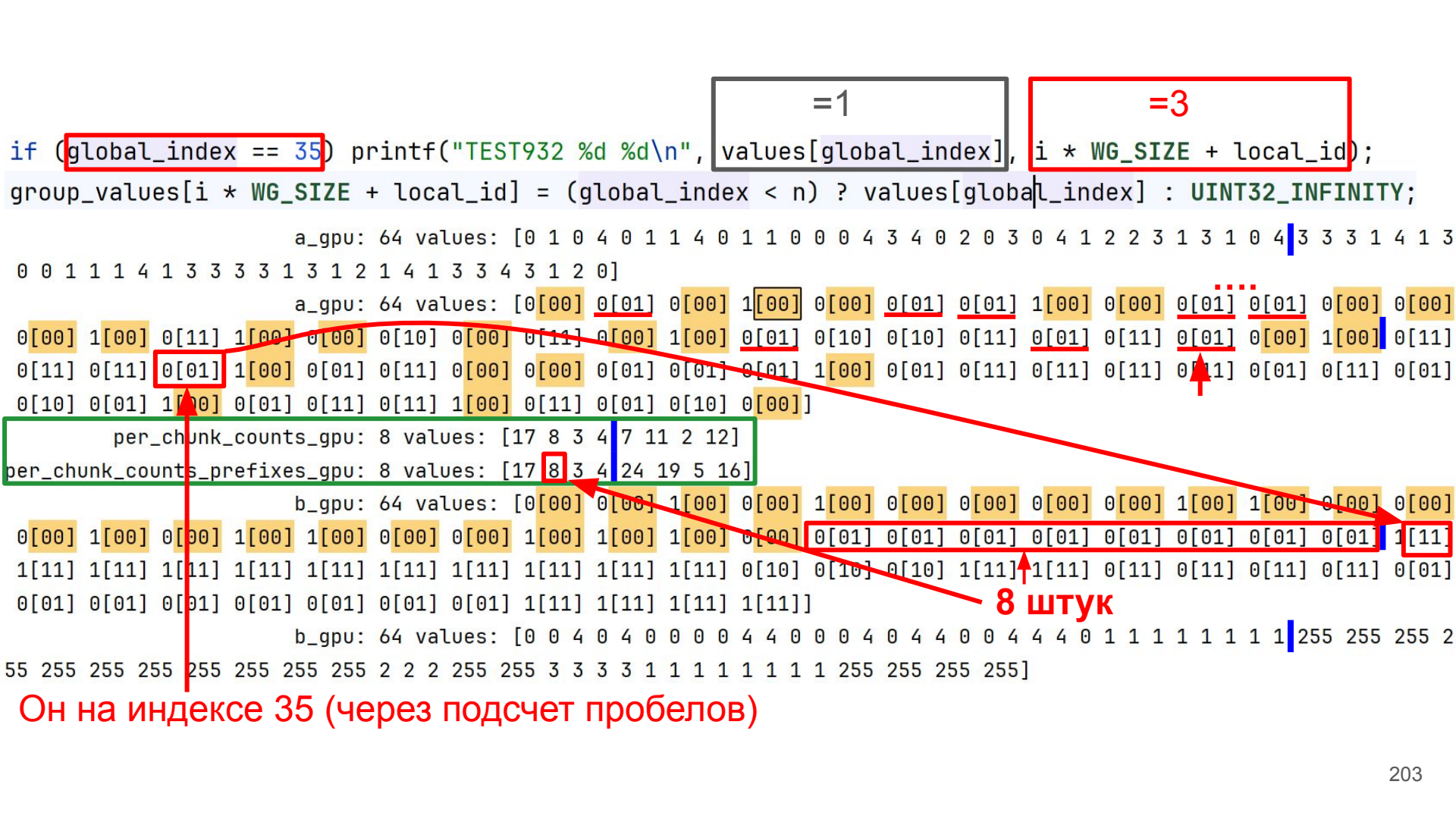
Он на индексе 35 (через подсчет пробелов)

```
if (global_index == 35) printf("TEST932 %d %d\n", values[global_index], i * WG_SIZE + local_id);
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;
```

```
                    a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
 0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
                    a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
 0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
 0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
 0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
         per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                    b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[11] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
                    b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 255 255 255 2
55 255 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
```

8 штук

Он на индексе 35 (через подсчет пробелов)


=1
=3
if (global_index == 35) printf("TEST932 %d %d\n", values[global_index], i * WG_SIZE + local_id);
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;
a_gpu: 64 values: [0 1 0 4 0 1 1 4 0 1 1 0 0 0 4 3 4 0 2 0 3 0 4 1 2 2 3 1 3 1 0 4 3 3 3 1 4 1 3
0 0 1 1 1 4 1 3 3 3 3 1 3 1 2 1 4 1 3 3 4 3 1 2 0]
a_gpu: 64 values: [0[00] 0[01] 0[00] 1[00] 0[00] 0[01] 0[01] 1[00] 0[00] 0[01] 0[01] 0[00] 0[00]
0[00] 1[00] 0[11] 1[00] 0[00] 0[10] 0[00] 0[11] 0[00] 1[00] 0[01] 0[10] 0[10] 0[11] 0[01] 0[11] 0[01] 0[00] 1[00] 0[11]
0[11] 0[11] 0[01] 1[00] 0[01] 0[11] 0[00] 0[00] 0[01] 0[01] 0[01] 1[00] 0[01] 0[11] 0[11] 0[11] 0[11] 0[01] 0[11] 0[01]
0[10] 0[01] 1[00] 0[01] 0[11] 0[11] 1[00] 0[11] 0[01] 0[10] 0[00]]
per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[11] 0[01]
0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 255 255 255 2
55 255 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
8 штук
Он на индексе 35 (через подсчет пробелов)

=3

```
if (global_index == 35) printf("TEST932 %d %d\n", values[global_index], i * WG_SIZE + local_id);
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;


unsigned int output_index = output_indices[i * WG_SIZE + local_id];
if (i * WG_SIZE + local_id == 3) printf("TEST777 output_index=%d\n", output_index);
output_values[output_index] = value;
```

```
        per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                      b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[11] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
                      b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 1255 255 255 2
55 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
```

```
if (global_index == 35) printf("TEST932 %d %d\n", values[global_index], i * WG_SIZE + local_id);
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;
```

=3

```
unsigned int output_index = output_indices[i * WG_SIZE + local_id];
if (i * WG_SIZE + local_id == 3) printf("TEST777 output_index=%d\n", output_index);
output_values[output_index] = value;
```

```
TEST777 output_index=2
TEST777 output_index=49
```

```
        per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                      b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[11] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
                      b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 1255 255 255 2
55 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
```

=3

```
if (global_index == 35) printf("TEST932 %d %d\n", values[global_index], i * WG_SIZE + local_id);
group_values[i * WG_SIZE + local_id] = (global_index < n) ? values[global_index] : UINT32_INFINITY;
```

```
unsigned int output_index = output_indices[i * WG_SIZE + local_id];
if (i * WG_SIZE + local_id == 3) printf("TEST777 output_index=%d\n", output_index);
output_values[output_index] = value;
```

TEST777 output_index=49

```
        per_chunk_counts_gpu: 8 values: [17 8 3 4 7 11 2 12]
per_chunk_counts_prefixes_gpu: 8 values: [17 8 3 4 24 19 5 16]
                      b_gpu: 64 values: [0[00] 0[00] 1[00] 0[00] 1[00] 0[00] 0[00] 0[00] 0[00] 1[00] 1[00] 0[00] 0[00]
 0[00] 1[00] 0[00] 1[00] 1[00] 0[00] 0[00] 1[00] 1[00] 1[00] 0[00] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11]
 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 1[11] 0[10] 0[10] 0[10] 1[11] 1[11] 0[11] 0[11] 0[11] 0[11] 0[01]
 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 0[01] 1[11] 1[11] 1[11] 1[11]]
                      b_gpu: 64 values: [0 0 4 0 4 0 0 0 0 4 4 0 0 0 4 0 4 4 0 0 4 4 4 0 1 1 1 1 1 1 1 1255 255 255 2
55 255 255 255 255 255 255 255 2 2 2 255 255 3 3 3 3 1 1 1 1 1 1 1 1 255 255 255 255]
```

CS Space
Клуб технологий и науки

Вопросы?

Vulkan

NVIDIA CUDA

OpenCL

@UnicornGlade
@PolarNick239
polarnick239@gmail.com
Николай Полярный