Parallel Computing with GPUs

Warp Level CUDA and Atomics Part 1 – Warp Scheduling and Divergence

The University Of Sheffield.

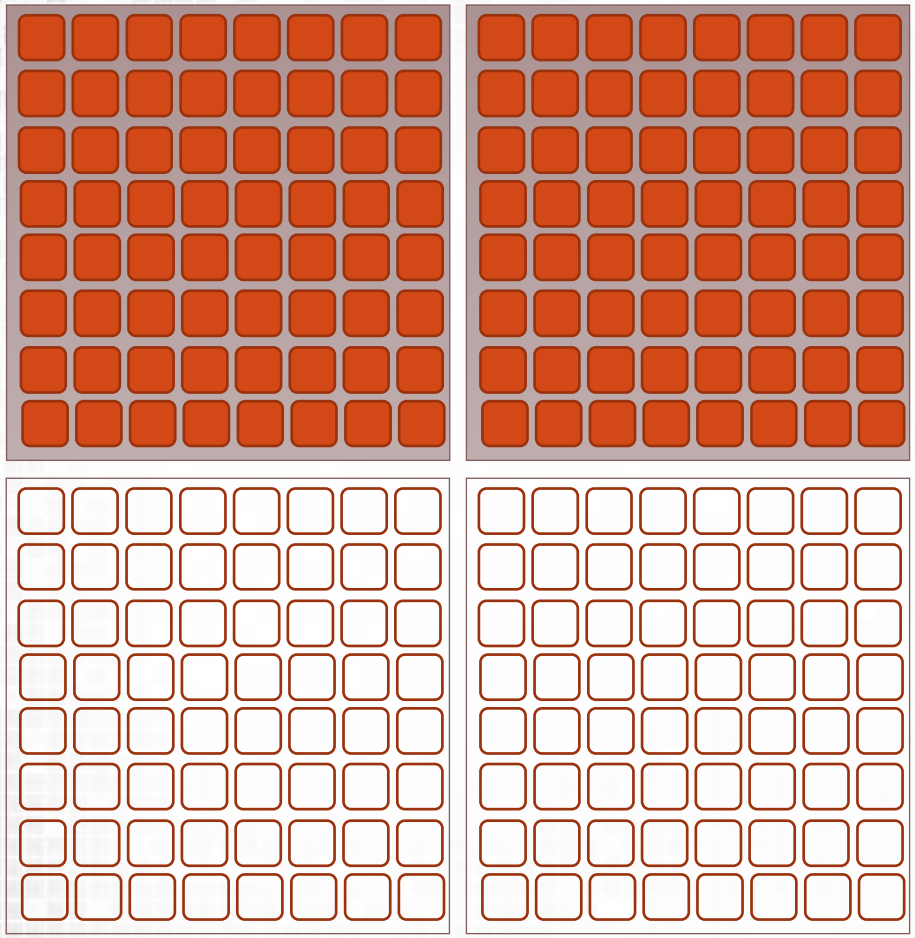
Dr Paul Richmond

http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- Warp Scheduling and Divergence
  - Summarise thread block scheduling
  - Examine warp scheduling
  - Define warp divergence and explore the impact via examples

Thread Block Scheduling



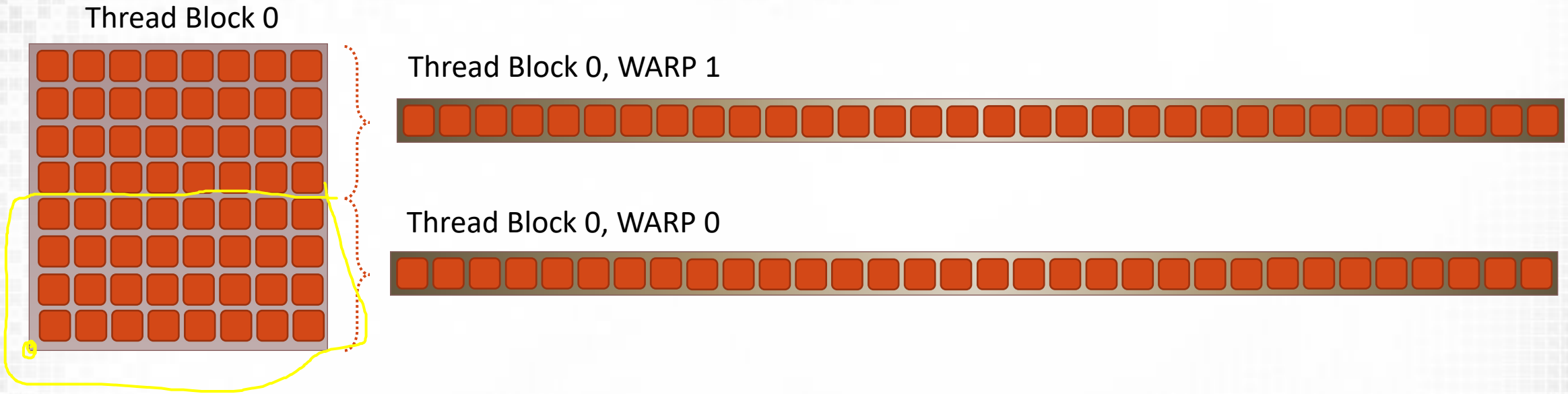
Multiple blocks scheduled to Streaming Multiprocessor

Streaming Multiprocessor 1

Streaming Multiprocessor 2

- No guarantee of block ordering on SMPs
- Hardware will schedule blocks to a SMP as soon as necessary resources are available

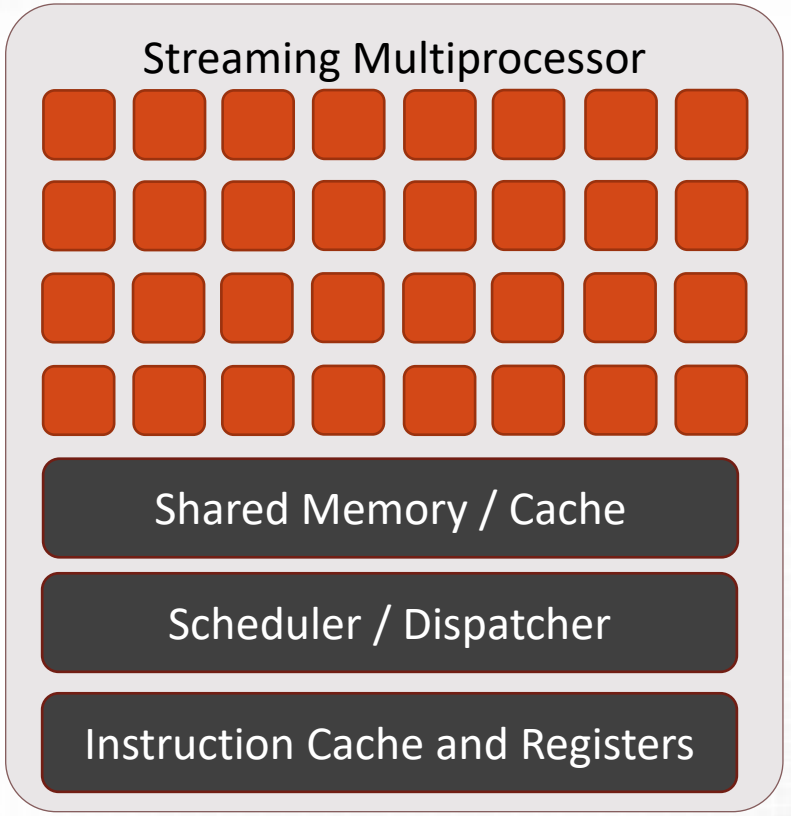
Thread Block Scheduling



- Each thread block is mapped to one or more warps
- 2D blocks are split into warps first by `x` index then `y` then `z`

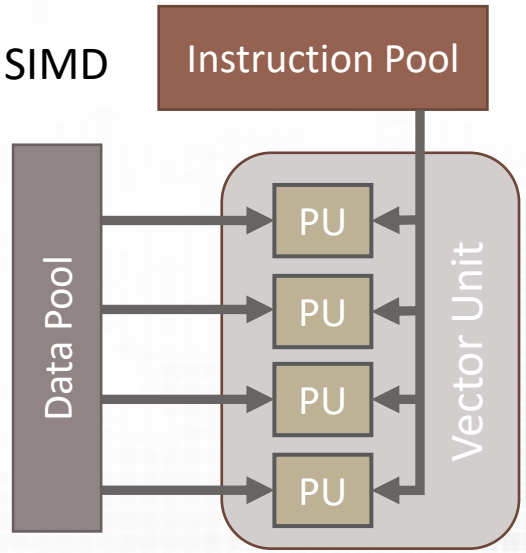
Warp Scheduling

- Zero overhead to swap warps (warp scheduling)
  - Warps contain only threads from a single thread block
  - Warps can be swapped with warps from different blocks assigned to the same streaming multi processor
  - At any one time only one warp has operations being executed
    - Memory movement happens in background
  - Occupancy impacts how many warps are available for scheduling



Warps (pre Volta)

- Execution of GPU instructions is always in groups of threads called **warps**
- A warp has a single program counter (pre-volta)
- Within a warps threads always execute the same instruction (SIMD/SIMT)
- What happens if code within a warp has different control flow?
  - **Branch Divergence**



Divergent Threads

- All threads in warp execute the same instruction
  - Multiple code branch paths must be evaluated
  - Not all threads will be active during code execution
  - Coherence = all threads following the same path

- How to avoid divergence
  1. Avoid conditional code
  2. **Especially** avoid conditional code based on `threadIdx`

- Fully coherent code can still have branches
  - BUT all threads in the warp follow the same path

Divergent and Coherent Code

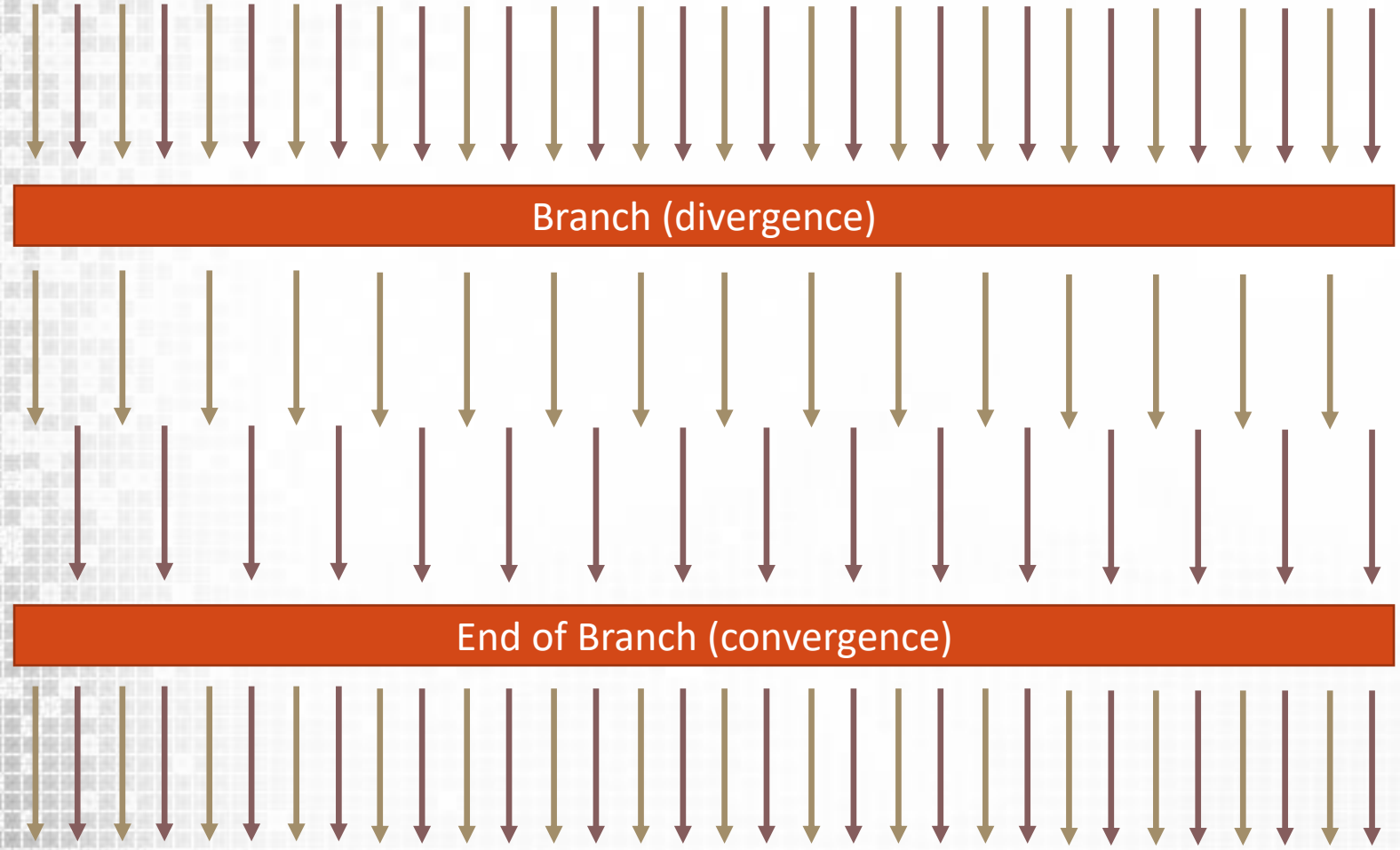
```
__global__ void a_kernel()
{
    if (blockIdx.x % 2)
        //something
    else
        //something else
}
```

```
__global__ void b_kernel()
{
    if (threadIdx.x % 2)
        //something
    else
        //something else
}
```

- Which is coherent?
- Which is divergent?

Divergence Example

All warps

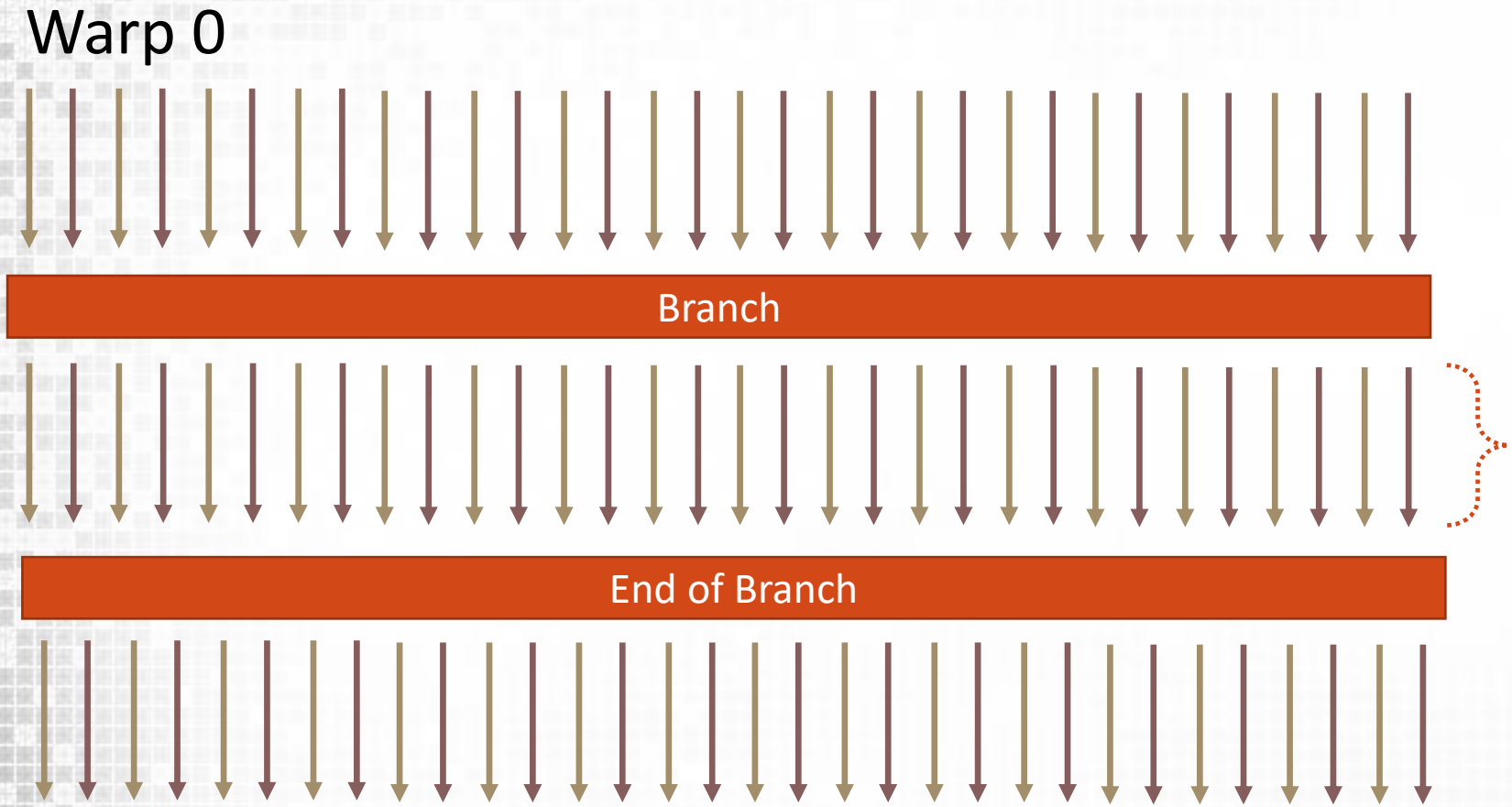



```
__global__ void a_kernel()
{
    if (threadIdx.x % 2)
        //something
    else
        //something else
}
```

```
if (threadIdx.x % 2)
```

```
else
```

Divergence Example Alternative

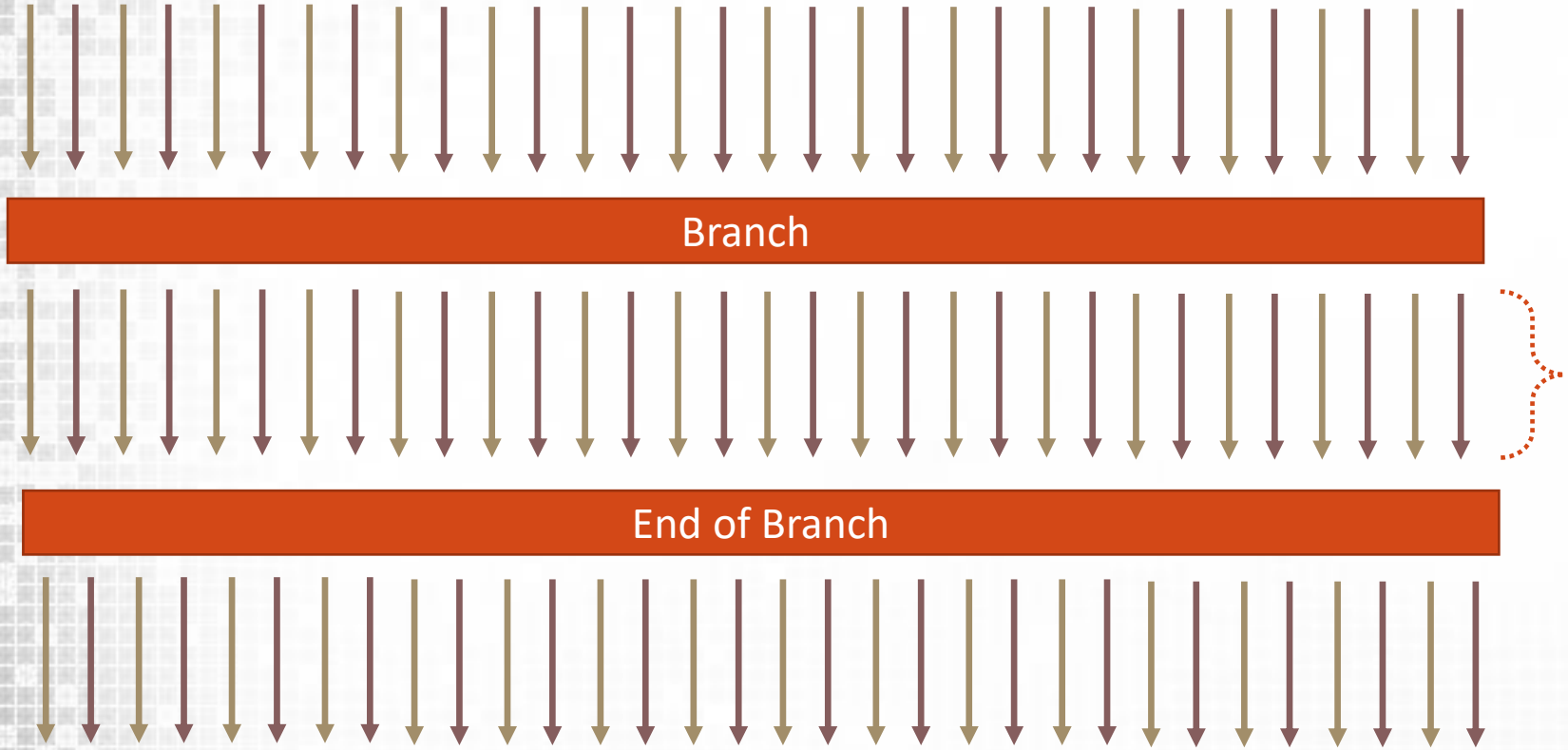



```
__global__ void a_kernel()
{
    if (blockIdx.x % 2)
        //something
    else
        //something else
}
```

```
if (blockIdx.x % 2)
```

Divergence Example Alternative

Warp 1 (assuming thread block size of 32)




```
__global__ void a_kernel()
{
    if (blockIdx.x % 2)
        //something
    else
        //something else
}
```

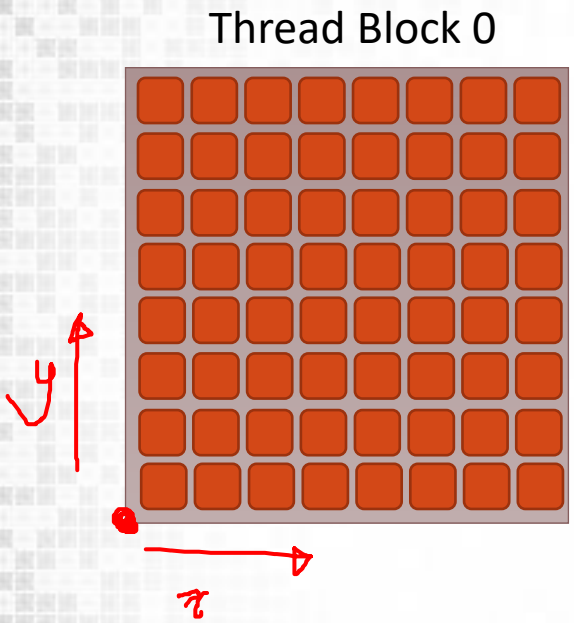
Levels of divergence

- Divergent code can be classified by how many "ways" it diverges.
  - E.g. the following examples are 4-way divergent (and functionally equivalent)
- If a warp has 32-way divergence this will have a **massive** impact on performance!

```
__global__ void a_kernel(int *a)
{
    int a = a[threadIdx.x + blockIdx.x*blockDim.x]
    if (a==0)
        //code for case 0
    else if (a==1)
        //code for case 1
    else if (a==2)
        //code for case 2
    else if (a==3)
        //code for case 3
}
```

```
__global__ void a_kernel(int *a)
{
    int a = a[threadIdx.x + blockIdx.x*blockDim.x]
    switch (a){
      case(0):
        //code for case 0 with break
      case(1):
        //code for case 1 with break
      case(2)
        //code for case 2 with break
      case(3)
        //code for case 3 with break
    }
}
```

2D blocks and divergence



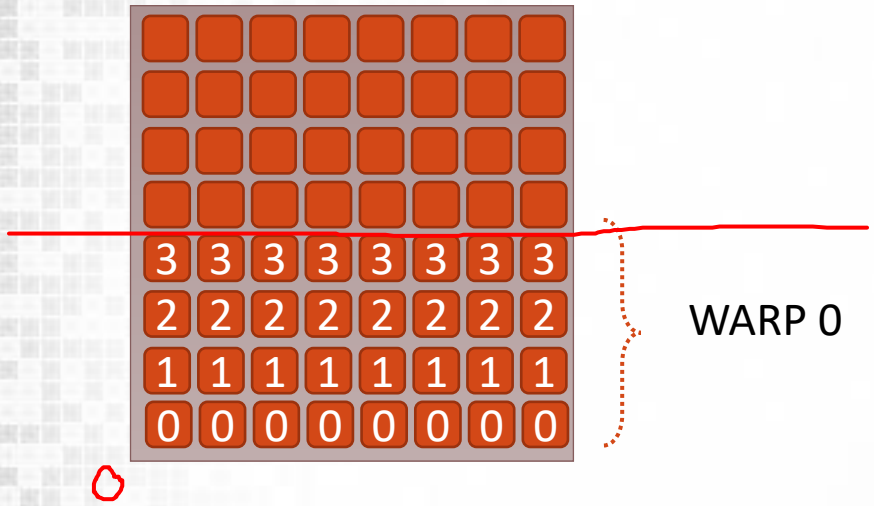

```
__global__ void a_kernel()
{
    if (threadIdx.y % 2)
        //something
    else
        //something else
}
```

```
__global__ void b_kernel()
{
    if (threadIdx.y / 4)
        //something
    else
        //something else
}
```

- How many ways of divergence?

2D blocks and divergence

Thread Block 0 – showing `threadIdx.y`




```
__global__ void a_kernel()
{
    if (threadIdx.y % 2)
        //something
    else
        //something else
}
```

2

```
__global__ void b_kernel()
{
    if (threadIdx.y / 4)
        //something
    else
        //something else
}
```

0

- How many ways of divergence?

Summary

- Warp Scheduling and Divergence
  - Summarise thread block scheduling
  - Examine warp scheduling
  - Define warp divergence and explore the impact via examples

- Next Lecture: Advanced Divergence (and Volta)