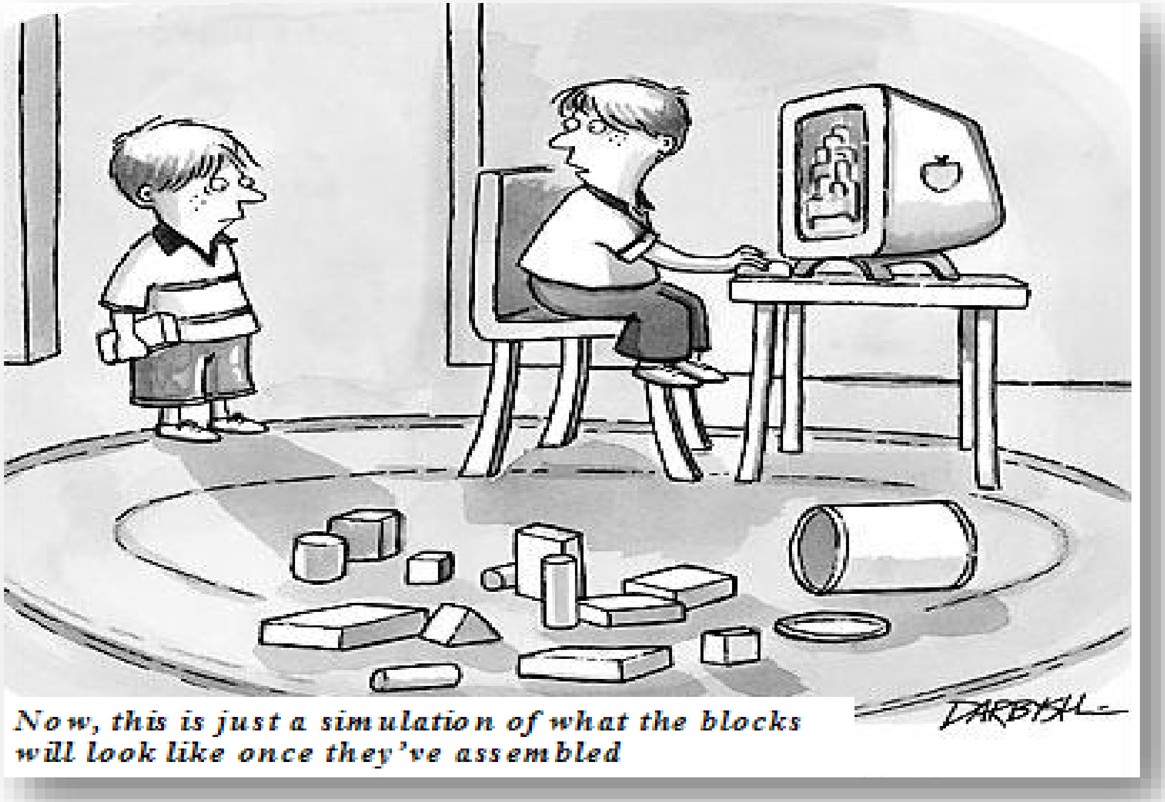
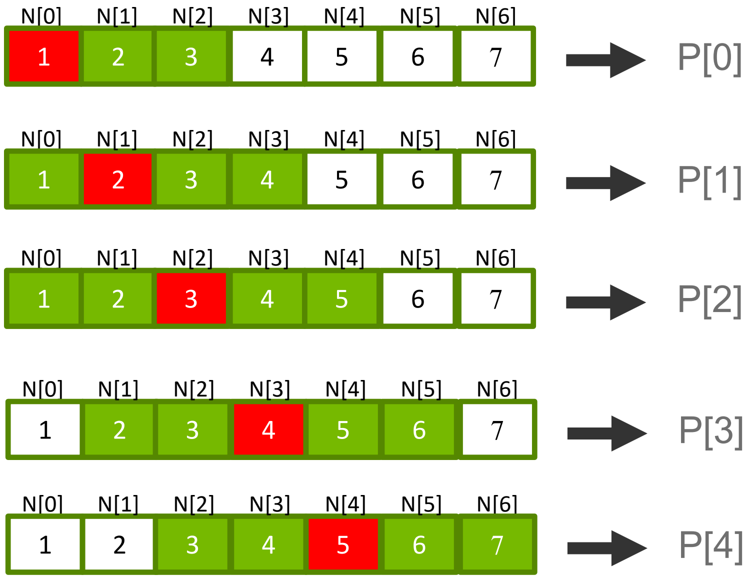
ECE569
Module 47


Now, this is just a simulation of what the blocks will look like once they've assembled
DARBYSHIRE

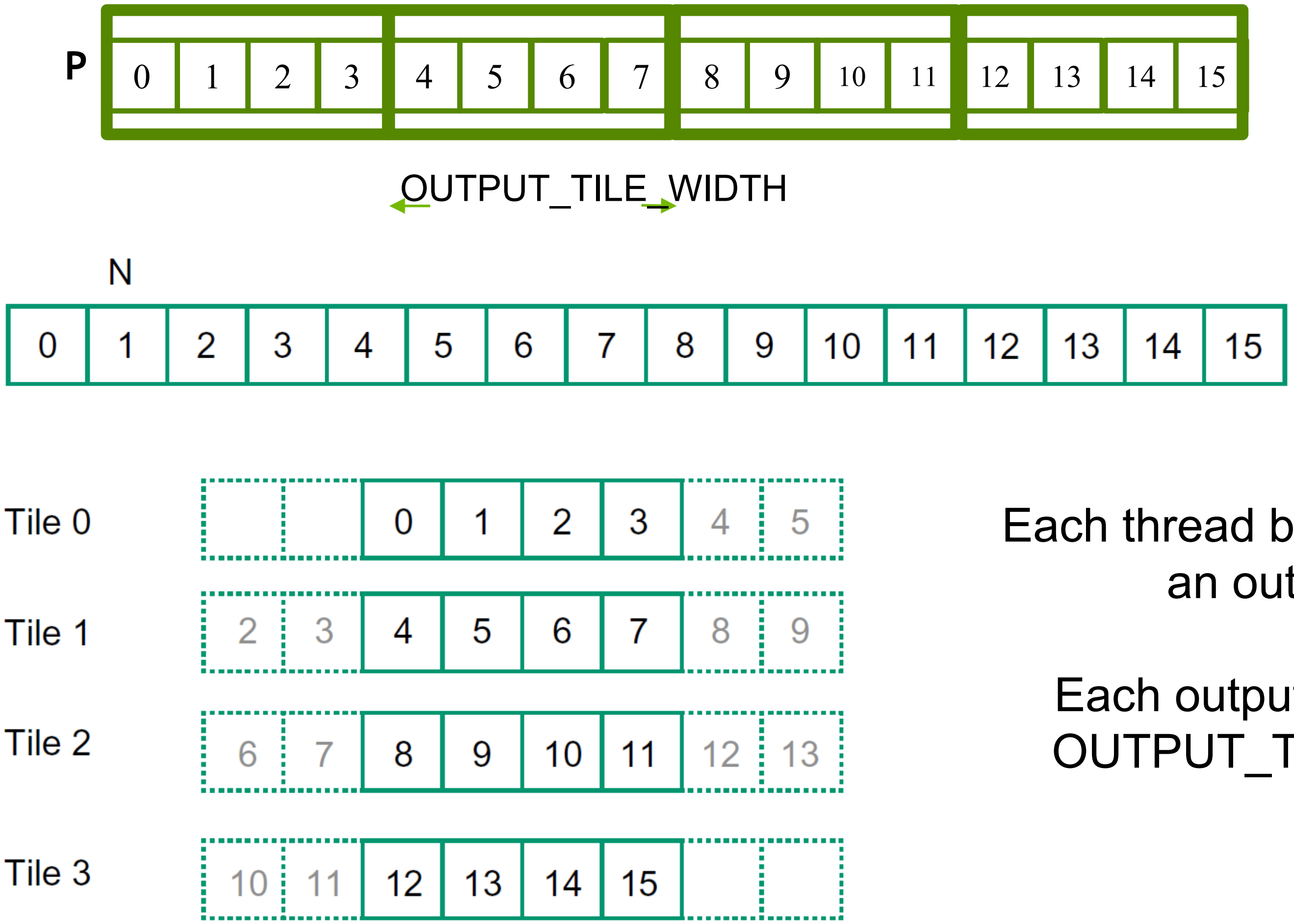
- Convolution – Tiling

Tiled 1D Convolution

- **Calculation of adjacent output elements involve shared input elements**
  - E.g., N[2] is used in calculation of P[0], P[1], P[2]. P[3] and P[4] for Mask_Width of width 5
  - load all input elements required by all threads in a block into the shared memory


N[0] N[1] N[2] N[3] N[4] N[5] N[6]
1 2 3 4 5 6 7
P[0]
N[0] N[1] N[2] N[3] N[4] N[5] N[6]
1 2 3 4 5 6 7
P[1]
N[0] N[1] N[2] N[3] N[4] N[5] N[6]
1 2 3 4 5 6 7
P[2]
N[0] N[1] N[2] N[3] N[4] N[5] N[6]
1 2 3 4 5 6 7
P[3]
N[0] N[1] N[2] N[3] N[4] N[5] N[6]
1 2 3 4 5 6 7
P[4]

Output Tiling

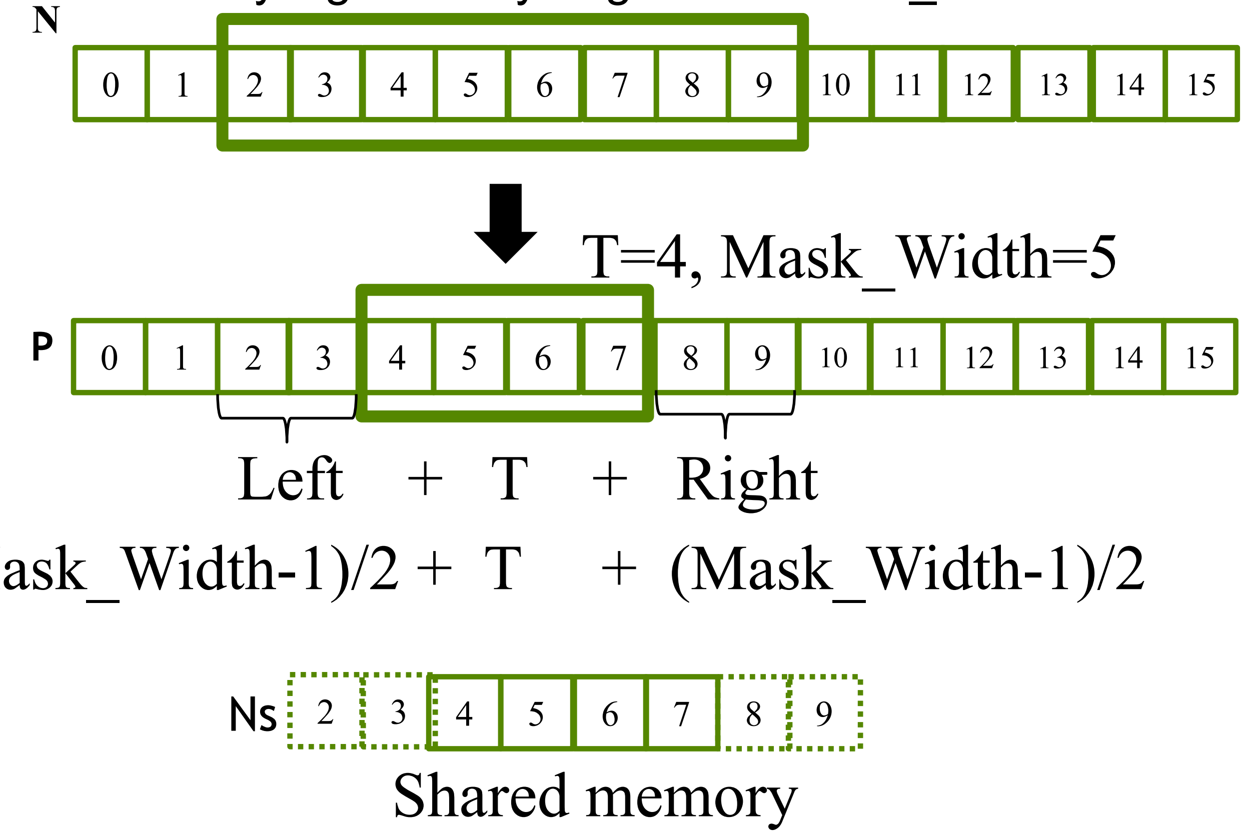


Each thread block calculates an output tile

Each output tile width is OUTPUT_TILE_WIDTH

Tiled 1D Convolution

- **Assume that we want to have each block to calculate TILE_SIZE (T) output elements**
  - T + Mask_Width -1 input elements are needed to calculate T output elements
    - T is usually significantly larger than Mask_Width



N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

T=4, Mask_Width=5

P: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

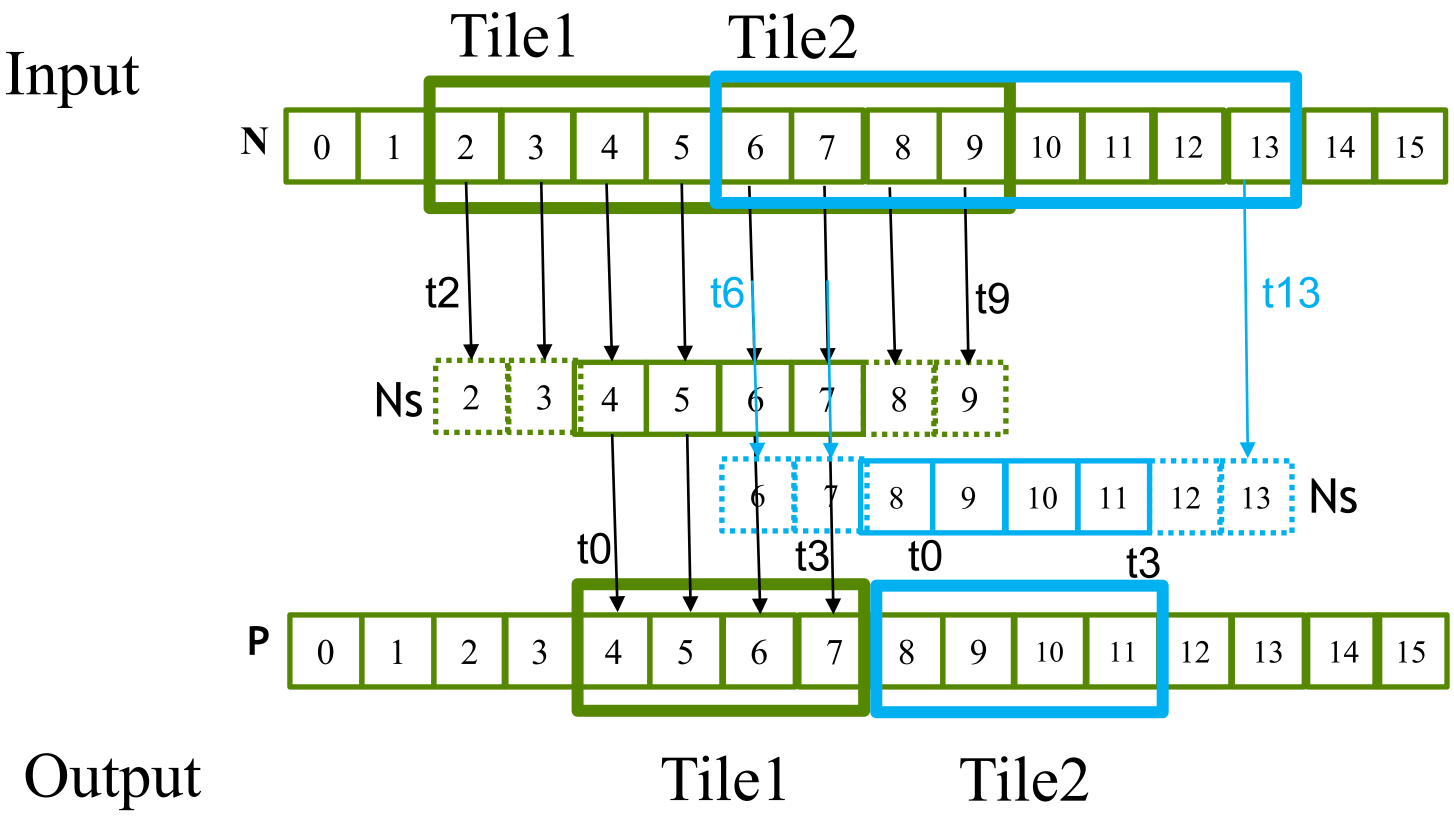
Left + T + Right

(Mask_Width-1)/2 + T + (Mask_Width-1)/2

Ns: 2 3 4 5 6 7 8 9

Shared memory

Definition - Input Tiles



Each input tile has all values needed to calculate the corresponding output tile.

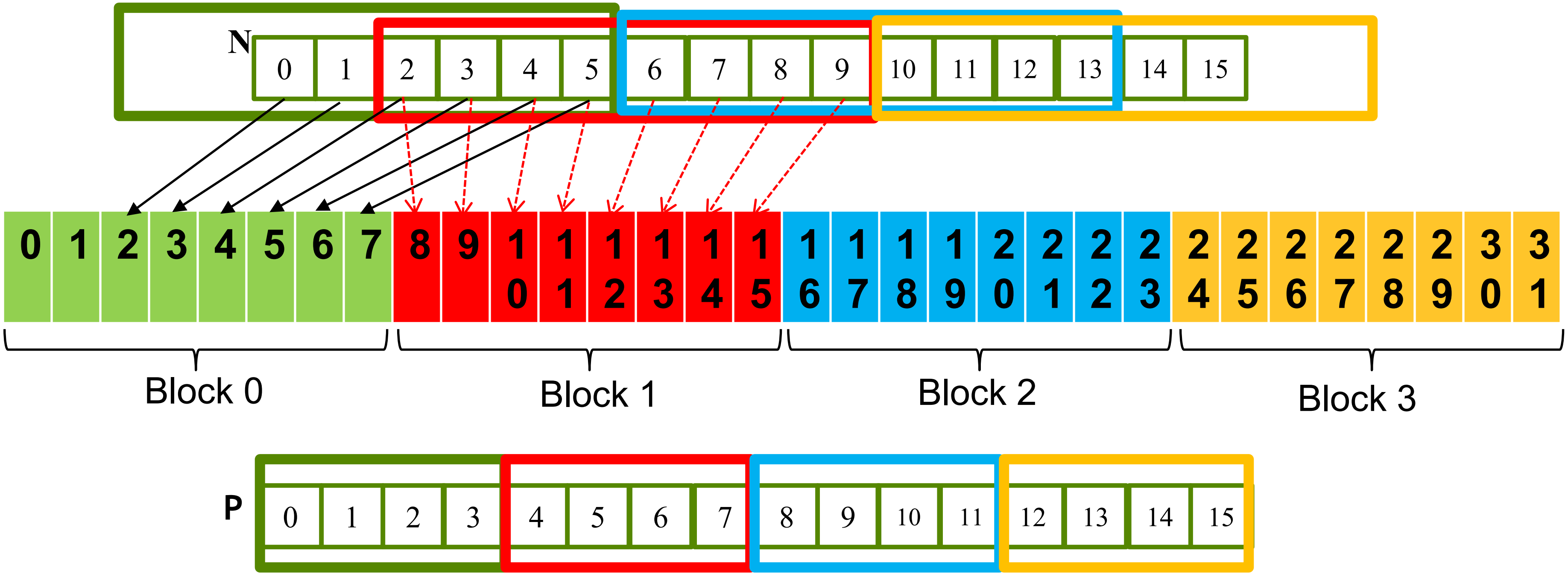
Implementation

- **Design 1: The size of each thread block matches the size of an output tile**

- **Design 2: The size of each thread block matches the size of an input tile**
  - Some threads will not participate in calculating output elements
    - blockDim.x would be 8 in our example
  - Each thread loads one input element into the shared memory

```
__shared__ float N_ds[TILE_SIZE + MAX_MASK_WIDTH - 1];
```

Reading from Global to Shared Memory



Output tile size (**To**) is 4, Mask Width (M) is 5

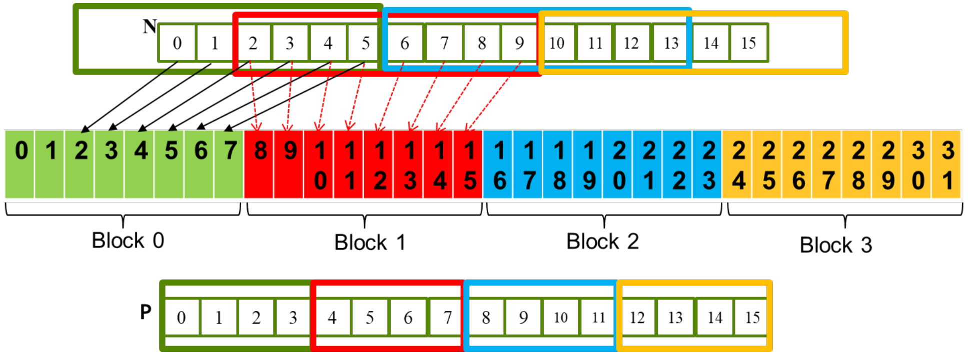
Threads/block= Input tile size (Ti) = To+Mask Width-1 = 8 threads per block

Number of thread blocks = size of N / To= 16/4 = 4

```
Ns[threadidx.x] = N[index_i]
```

Write **index_i** as a function of i, T, and any other thread identifier in the grid

Reading from Global to Shared Memory



P [range]	i [range]	N [range]
Tile0:	Block 0:	
Tile1:	Block 1:	
Tile2:	Block 2:	
Tile3:	Block 3:	

i= threadIdx.x + blockIdx.x*blockDim.x

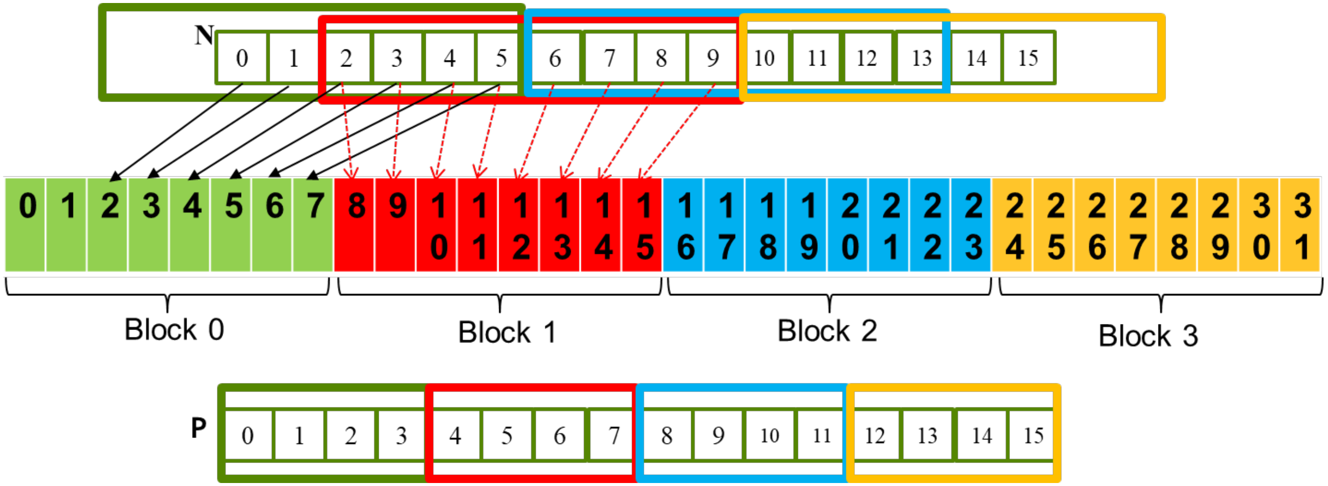
Output tile size (**To**) is 4, Input tile size (Ti) is 8, Mask Width (M) is 5

Grid is organized as 4 thread blocks and 8 threads/block

```
Ns[threadidx.x] = N[index_i]
```

Write the **index_i** expression

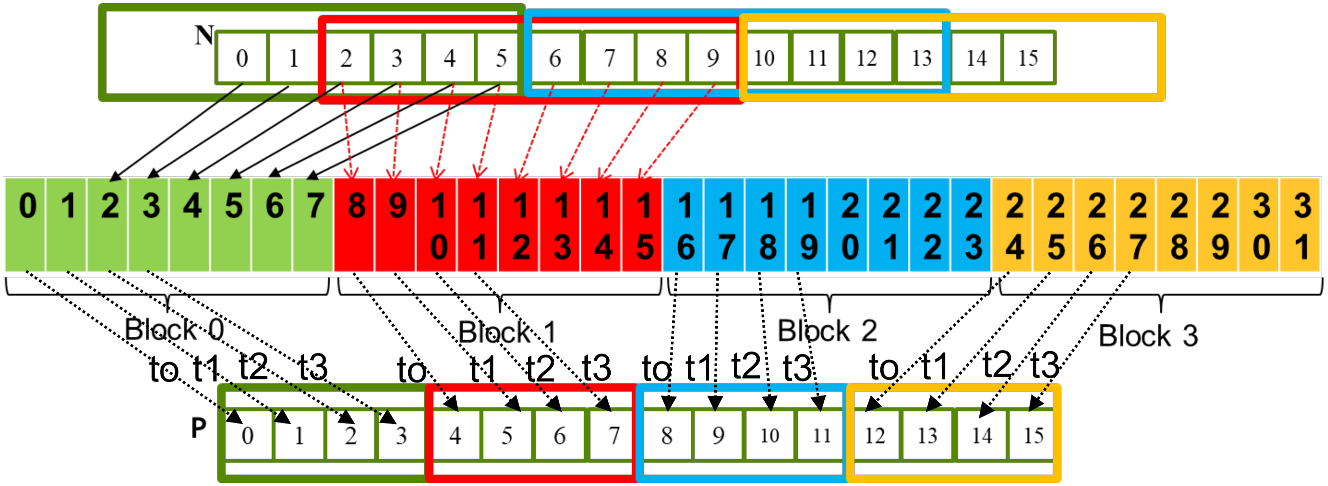
Loading from Global to Shared Memory



```
float output = 0.0f;

  if(_______________________________________) {
    Ns[tx] = N[index_i];
  }
  else{
    Ns[tx] = 0.0f;
  }
```

Some threads do not participate in calculating output



```
if (_______________________________________){
      output = 0.0f;
      for(j=______________________________)  {

          output += M[______]* Ns[_____________];
      }
      P[output_i] = output;
  }
```

Memory Accesses

- For a tiled 1D convolution, if the output tile width is 250 elements and mask width is 7 elements, what is the input tile width?
  - 250
  - 254
  - 256
  - 7

Memory Accesses

- For a tiled 1D convolution, if the output tile width is 250 elements and mask width is 7 elements, what would be the ratio of global memory reduction for generating the output tile by loading the input tile into the shared memory?
  - 250*7/256
  - 256*7/250
  - 7
  - 250

Next

- Project Requirements