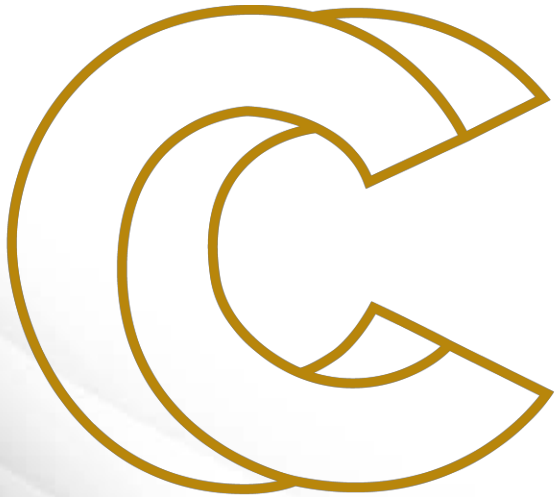
TÜBİTAK

Sabancı Üniversitesi



EURO^2

Performance Engineering on CPUs and GPUs:

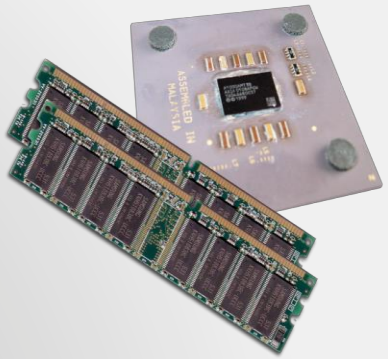
- GPUs: Things to be Careful for Performance -

Kamer Kaya, Sabancı University

ncc@ulakbim.gov.tr

Terminology

- *Host* The CPU and its memory (host memory)
- *Device* The GPU and its memory (device memory)



Host

Device

Hello World! with Device Code

```
__global__ void gpu_kernel() {…}

int main() {
        gpu_kernel<<<1,1>>>();
        printf("Hello to the GPU!\n");
        return 0;
}
```

main() runs on the CPU.

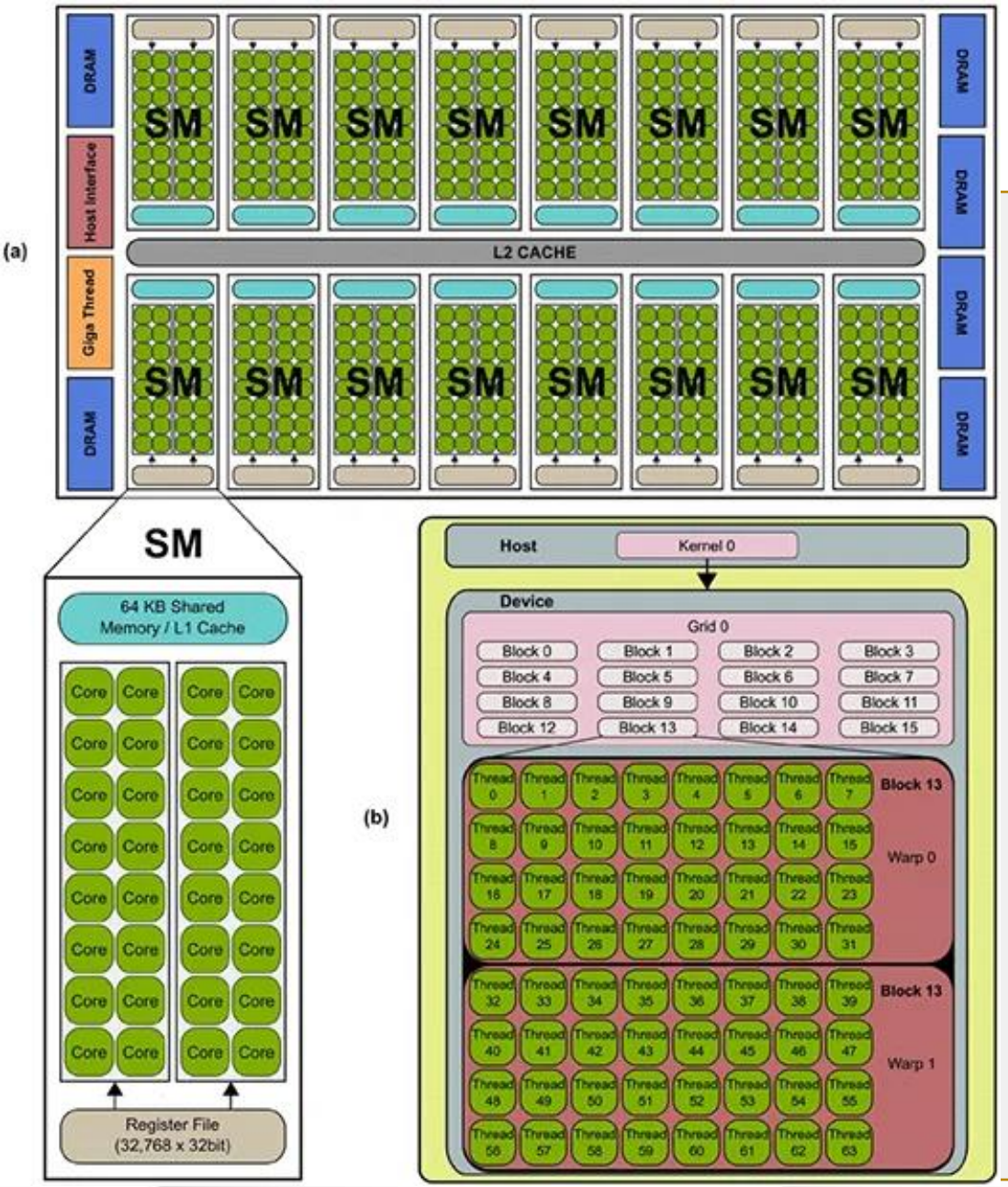
gpu_kernel() runs on the GPU.

`__global__` indicates a function:
runs on the device that
is called from host code (main)

When compiled with **nvcc:**

- Host and device codes are separated..
  - Device functions (e.g., **gpu_kernel()**) processed by Nvidia compiler.
  - Host functions (e.g. **main()**) processed by standard host compiler (e.g., **gcc)**

GPU Architecture




```
gpu_kernel<<<grid_dims, block_dims>>>();
```

- A **kernel** is executed by a **grid** (a set of blocks ordered in 3d)
- A **block** (a set of threads organized in 3d) is assigned to an SM.
- A **warp** is a set of threads controlled by a single controller.

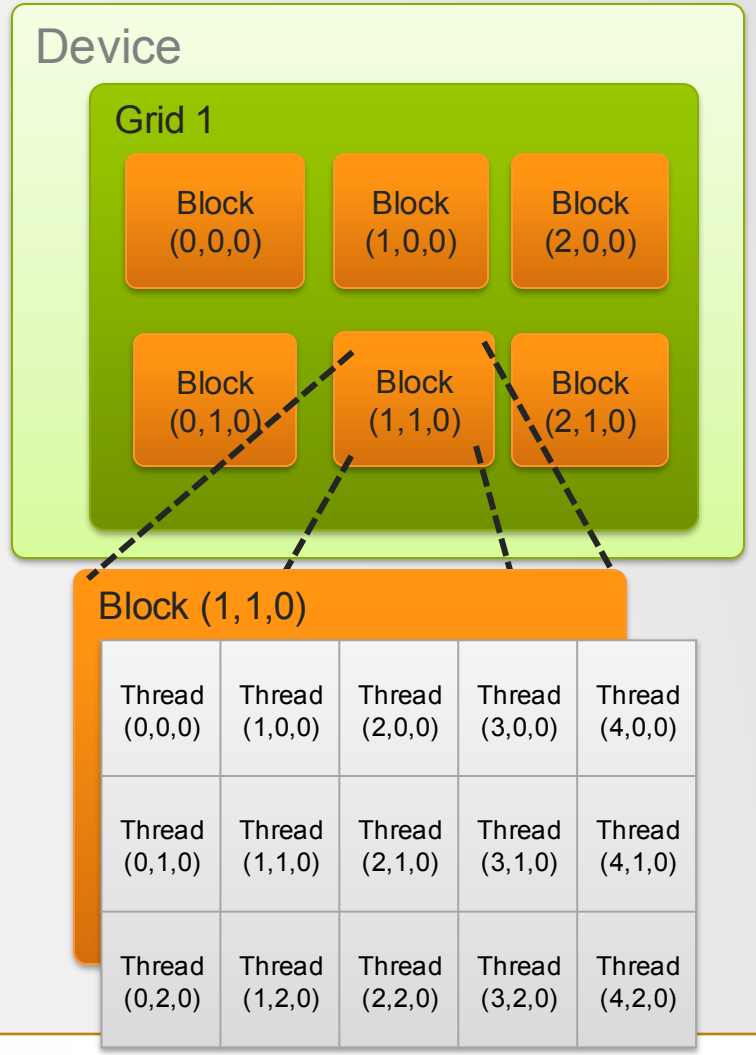
To fully use the device, we need both blocks and threads.

GPU: IDs and Dimensions

EURO²

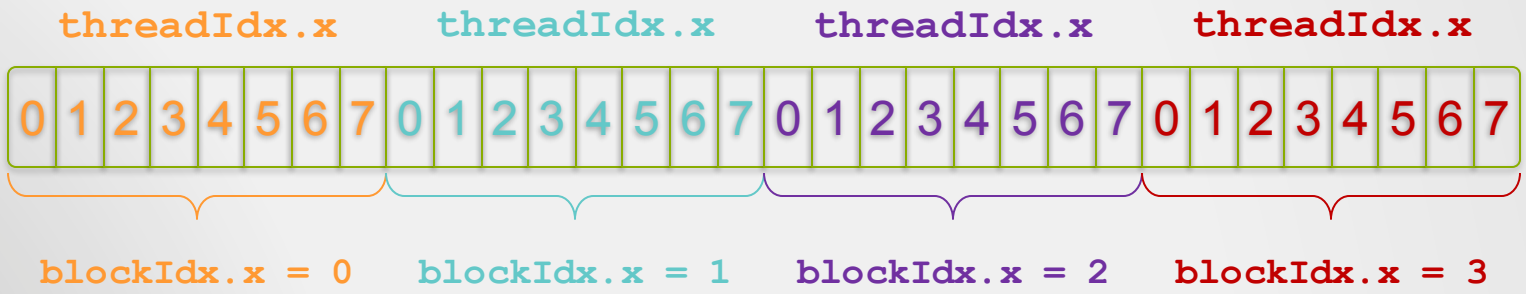
- A kernel is launched as a grid of blocks of threads
  - `blockIdx` and `threadIdx` are 3D
  - We showed only one dimension (`x`)

- Built-in variables:
  - `threadIdx`
  - `blockIdx`
  - `blockDim`
  - `gridDim`



CUDA: Parallelism – Blocks/Threads

- Using `blockIdx.x` and `threadIdx.x`
  - Consider indexing an array with one element per thread (8 threads/block)

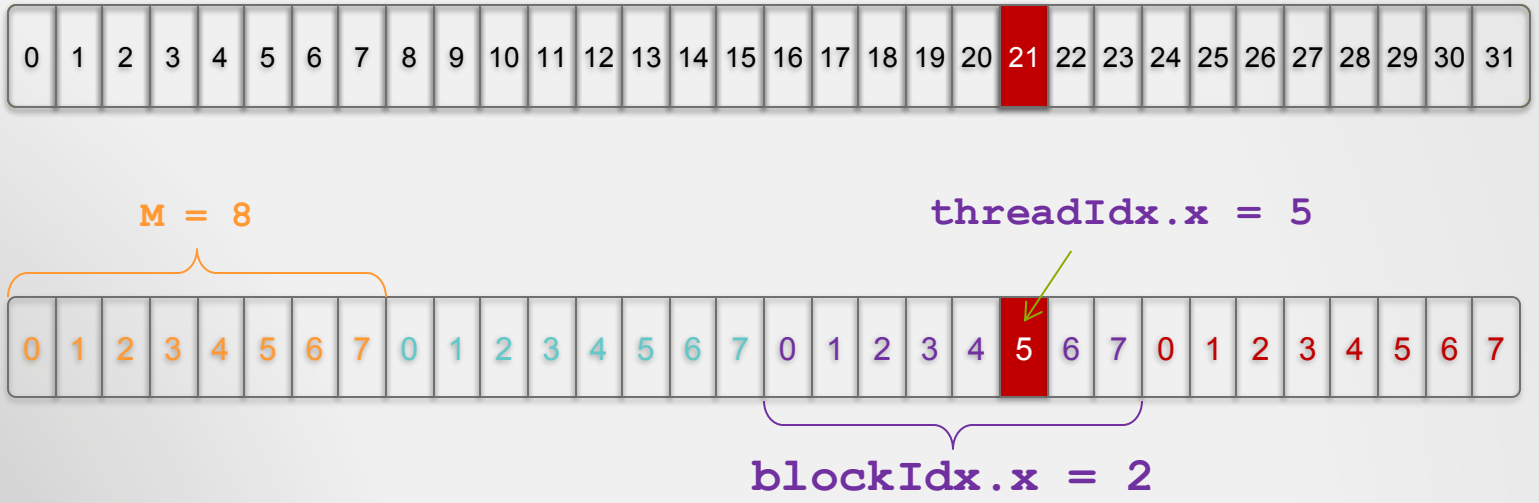


- With M threads/block a unique index for each thread is given by:

```
int index = threadIdx.x + blockIdx.x * M;
```

CUDA: Parallelism – Blocks/Threads

- Which thread will operate on the red element?




```
int index = threadIdx.x + blockIdx.x * M;
          =      5      +     2      * 8;
          = 21;
```

CUDA: Parallelism – Blocks/Threads

EURO²

- Use the built-in variable `blockDim.x` for threads per block

```
int index = threadIdx.x + blockIdx.x * blockDim.x;
```

- Combined version of `add()` to use parallel threads *and* parallel blocks

```
__global__ void add(int *a, int *b, int *c) {
    int index = threadIdx.x + blockIdx.x * blockDim.x;
    c[index] = a[index] + b[index];
}
```

CUDA: Parallelism – Blocks/Threads

EURO²

```
#define N (2048*2048)
#define THREADS_PER_BLOCK 512

                                    ....

                                    ....

add<<<N/THREADS_PER_BLOCK,,THREADS_PER_BLOCK>>>(d_a, d_b, d_c);
```

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia