CUDA. Memory

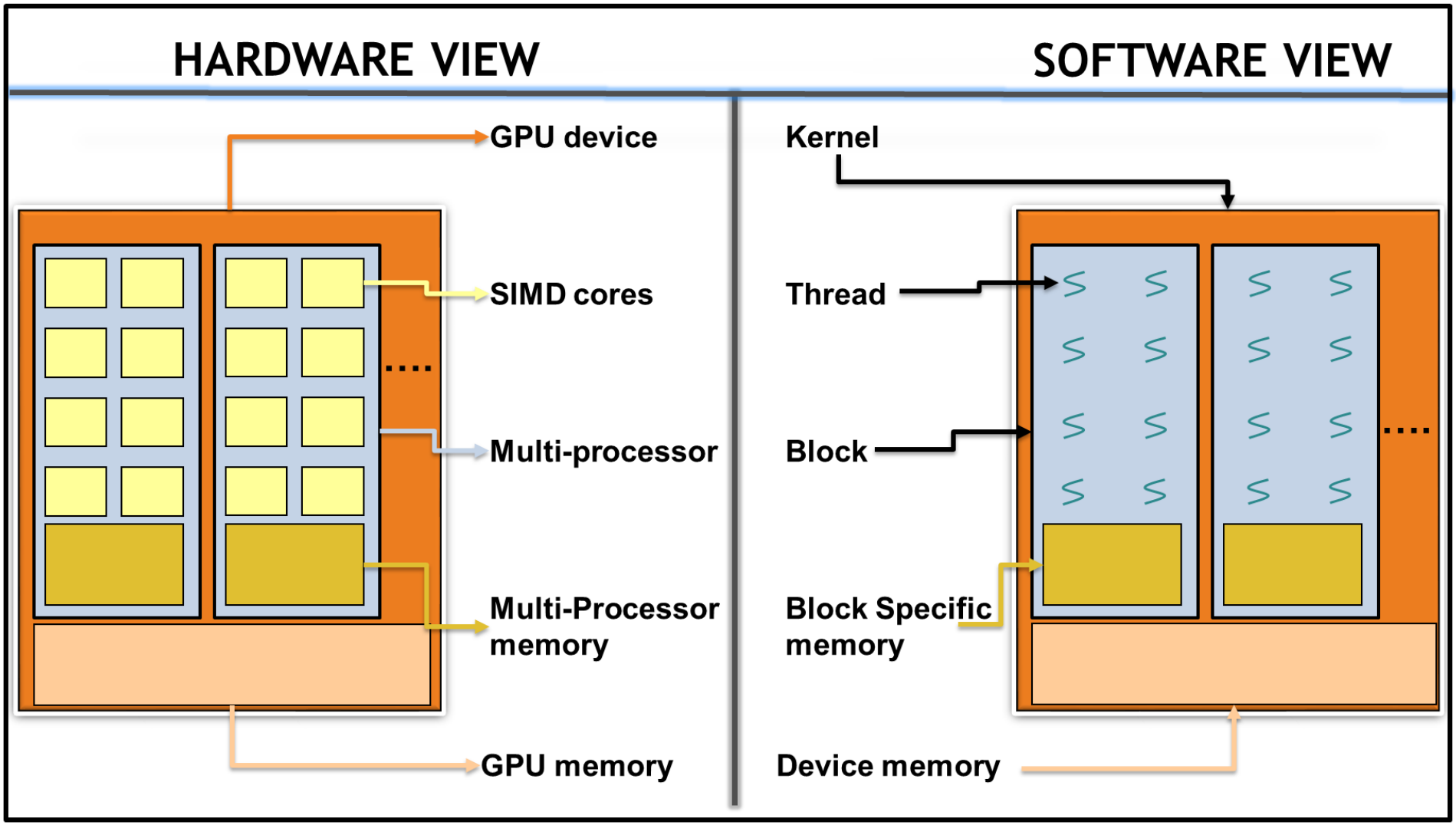
F.CS306 ПАРАЛЛЕЛ ПРОГРАММЧЛАЛ – Лекц 12

Г.ГАНБАТ
ganbatg@must.edu.mn

Хичээлийн агуулга

- CUDA программын дараалал
- CUDA Санах ойн шатлал
- Өгөгөдөл солилцоо
- Санах ой нөөцлөлт
- Өгөгөдөл зөөвөрлөлт
- Жишээ, optimization

CUDA views

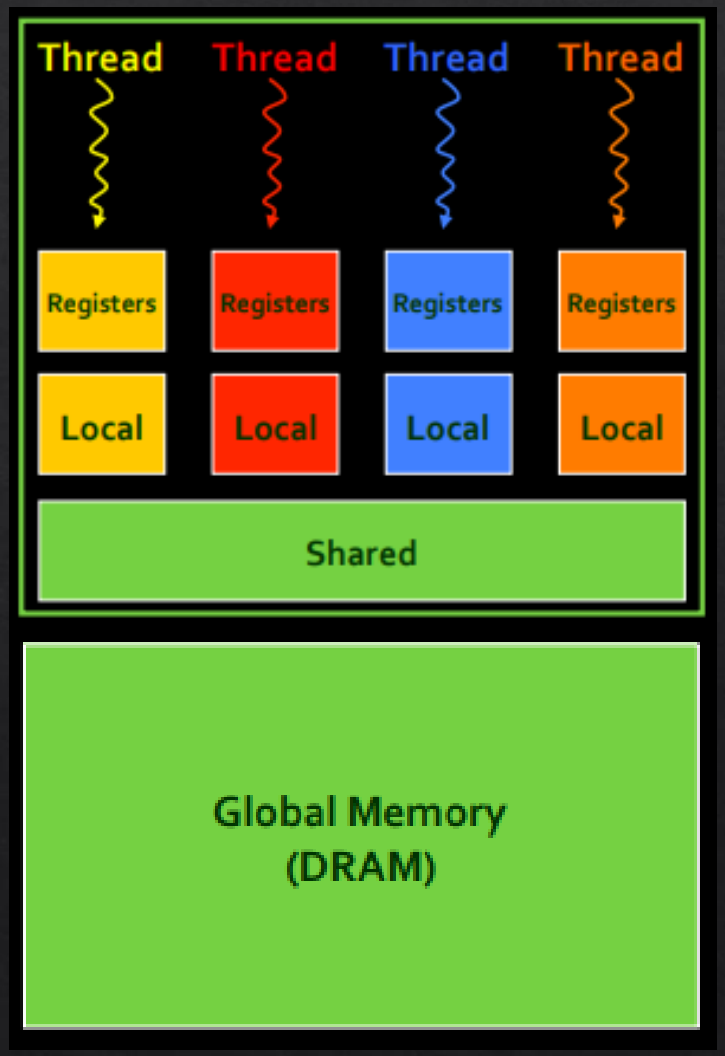

HARDWARE VIEW
SOFTWARE VIEW
GPU device
SIMD cores
Multi-processor
Multi-Processor memory
GPU memory
Kernel
Thread
Block
Block Specific memory
Device memory

CUDA программын дараалал

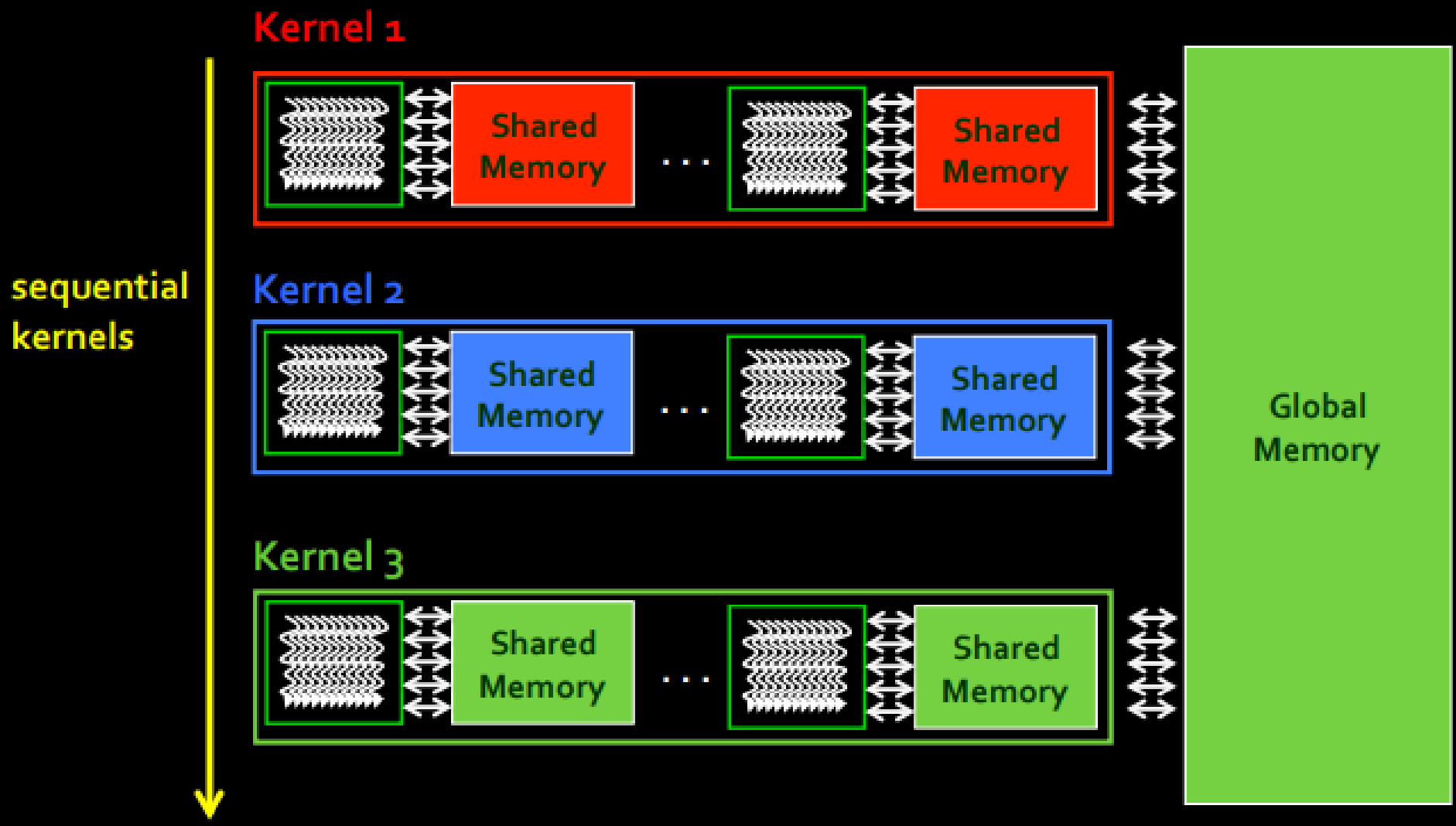
- CUDA нь Device-ын санах ой нөөцлөх ба Host болон Device-ын санах ой хооронд өгөгдөл дамжуулахад зориулагдсан API-ийг удирддаг.
  1. Host-д санах ой нөөцлөн өгөгдлийг ачаалах
  2. Device-д санах ой нөөцлөх
  3. Оролтын өгөгдлийг Host-ын санах ойгоос Device-руу зөөх
  4. Кернелийг ажиллуулах
  5. Гаралтын өгөгдлийг Device-ын санах ойгоос Host-руу зөөх

CUDA Санах ойн шатлал

- Thread
  - Registers
  - Local memory
- Thread Block
  - Shared memory
- All Thread Blocks
  - Global Memory



Өгөгдөл солилцоо



Санах ой нөөцлөлт

- Host нь device санах ойг удирдана
  - cudaMalloc (void ** pointer, size_t num_bytes)
  - cudaMemset (void* pointer, int value, size_t count)
  - cudaFree(void* pointer)
- CPU – cudaMallocHost, malloc, new
- GPU – cudaMalloc

```
cudaMalloc(void **devPtr, size_t count);
cudaFree(void *devPtr);
```

Өгөгдөл зөөвөрлөлт

```
cudaMemcpy (void *dst, void *src, size_t count, cudaMemcpyKind kind)
```

- Параметерууд
  1. Хуулан байрлуулах байршлийн хаяг
  2. Өгөгдлийн заагч
  3. Зөөвөрлөх байтуудын хэмжээ
  4. Хуулах чиглэл
     - cudaMemcpyHostToDevice
     - cudaMemcpyDeviceToHost
     - cudaMemcpyDeviceToDevice

Жишээ: Вектор нэмэх

```
void main(){
    float *a, *b, *out;
    float *d_a;

    a = (float*)malloc(sizeof(float) * N);

    // Allocate device memory for a
    cudaMalloc((void**)&d_a, sizeof(float) * N);

    // Transfer data from host to device memory
    cudaMemcpy(d_a, a, sizeof(float) * N, cudaMemcpyHostToDevice);
    …
    vector_add<<<1,1>>>(out, d_a, b, N);
    …
    // Cleanup after kernel execution
    cudaFree(d_a);
    free(a);
}
```

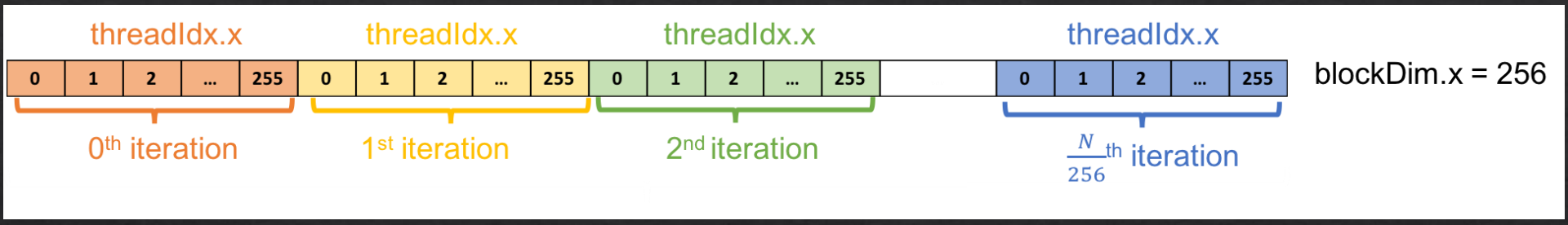
Параллелчлал

- CUDA нь ажиллахдаа GPU дээр хэдэн thread ачаалах ёстойг тодорхойлно.
- Бичиглэл: Кернщл нь M ширхэг thread блокын grid-тэй ачаалагдана. Блок бүр T ширхэг параллел thread агуулна.

```
<<< M , T >>>
```

- CUDA нь "***grid***“ бүтэц дотор зохион байгуулагдах "***thread block***“ –д thread-үүдийг бүлэглэн авч үздэг. Дараах хувьсагдуудыг ашиглан хандана:
  - blockIdx.x: Grid доторх блокын дугаар
  - threadIdx.x: Блок доторх thread-ийн дугаар
  - blockDim.x: Thread блокын хэмжээ / блок доторх thread-ийн тоо
  - gridDim.x: Grid-ын хэмжээ.

Жишээ: Вектор нэмэх



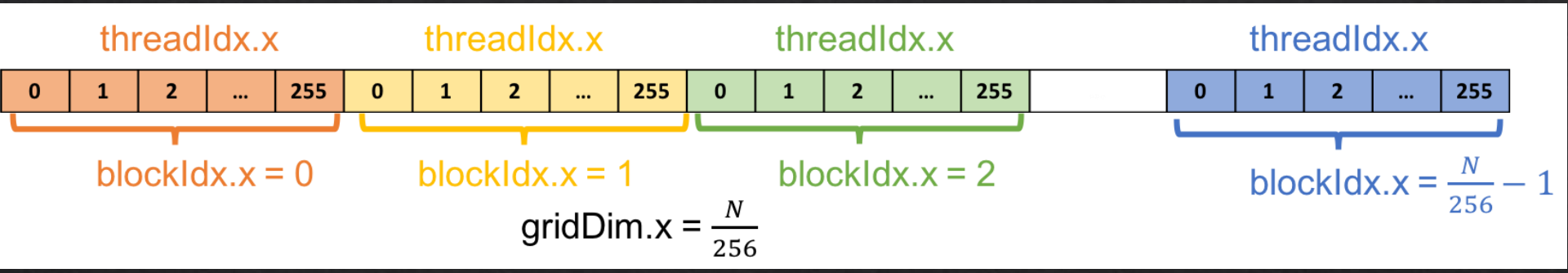

```
__global__ void vector_add(float *out, float *a, float *b, int n) {
    int index = 0;
    int stride = 1
    for(int i = index; i < n; i += stride){
        out[i] = a[i] + b[i];
    }
}
```

- Host-оос ажиллуулах

```
vector_add <<< 1 , 256 >>> (d_out, d_a, d_b, N);
```

Жишээ: Вектор нэмэх

- CUDA GPU нь SM-уудтай. Эдгээр нь олон зэрэгцээ thread блокуудыг ажиллуулах боломжтой.




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

Version	Execution Time (ms)	Speedup
1 thread	1425.29	1.00x
1 block	22.78	62.56x
Multiple blocks	1.13	1261.32x

Баярлалаа.