

GPU Programming in Computer Vision

Warps

Lecture Week

Lecture

- 10-14 (1h lunch pause) each day
- attendance mandatory to pass the course

Exercises

- 14-18 each day
- no need to be finished the same day

Deadline for exercises:

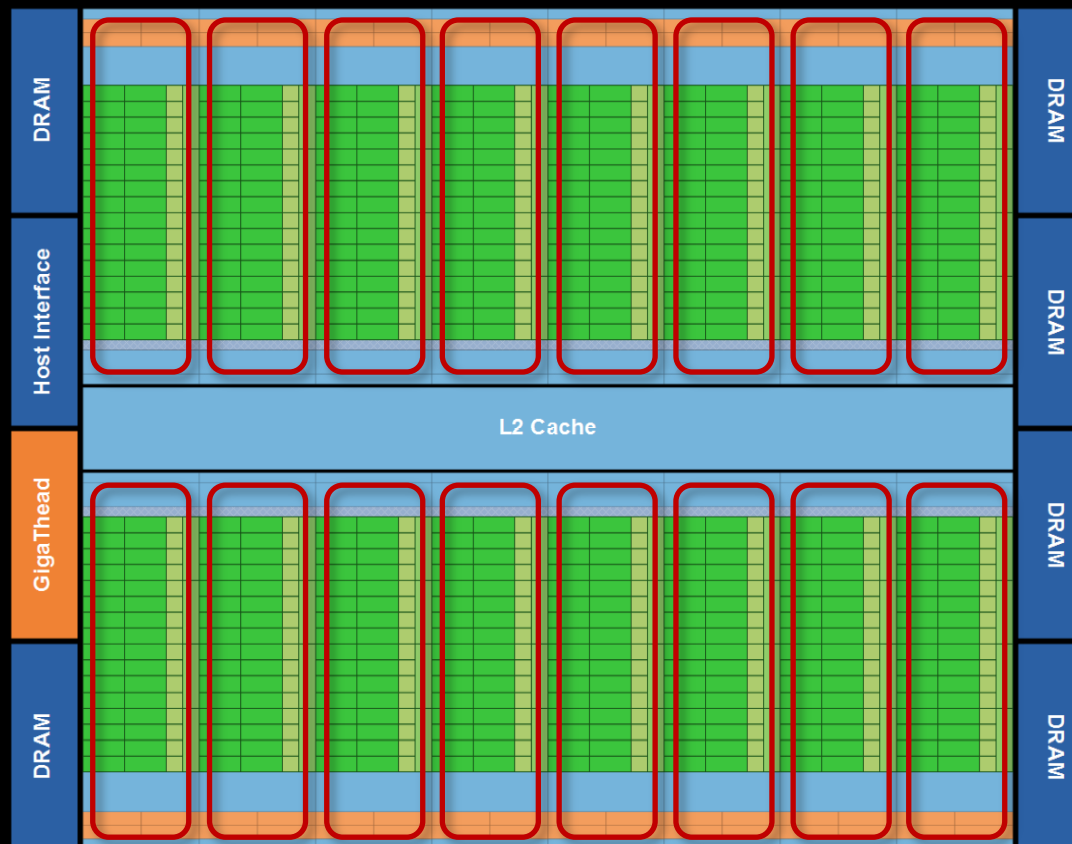
- 02.09.2013, 23:59
- Submit all solutions by email in a zip archive

September						
Mo.	Di.	Mi.	Do.	Fr.	Sa.	So.
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

WARPS

NVIDIA GPU Architecture

Fermi
GPU

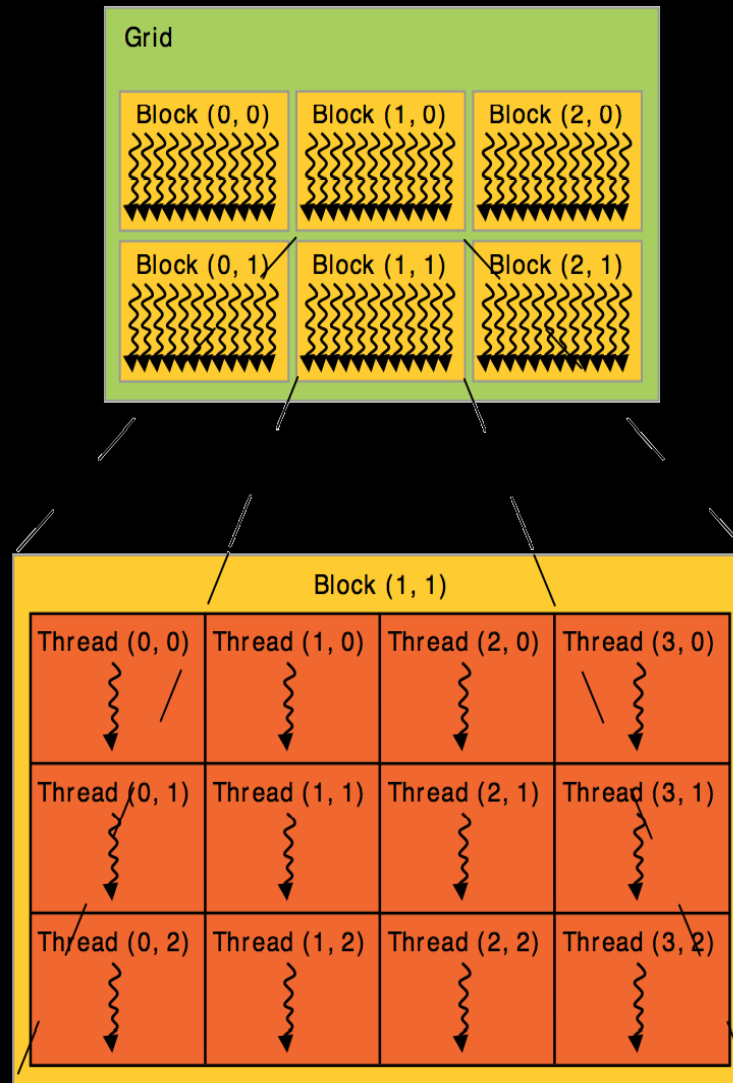


- **16 independent** multiprocessors (SMs)
- **No shared resources** except global memory
- **No synchronization**, always work **in parallel**

Warps

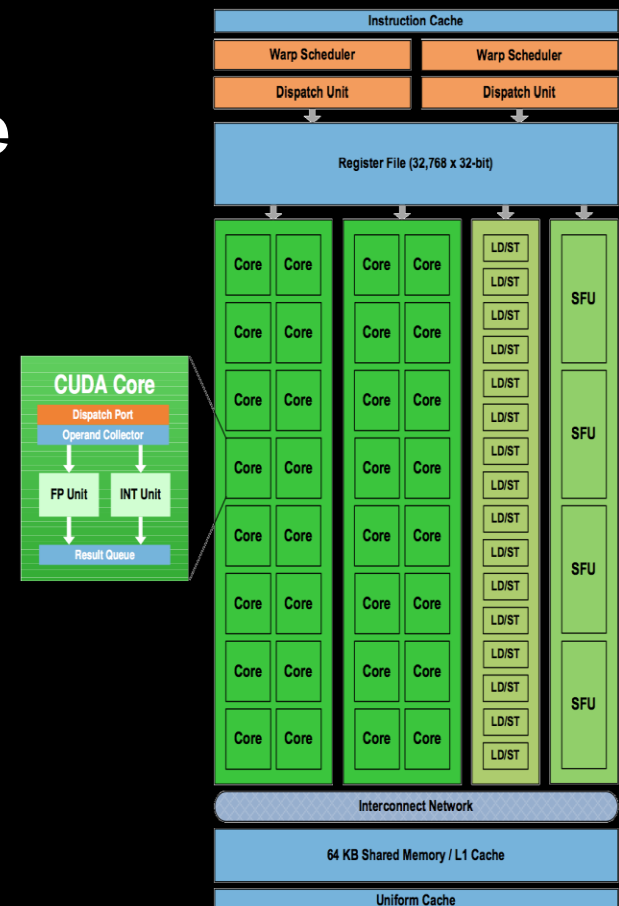
- **SIMT** (Single Instruction Multiple Thread) **execution**
 - threads run in groups of 32 called warps
- All 32 threads in a warp execute **the same** instruction
 - always, no matter what (even if threads diverge)
- Threads are executed **warp-wise** by the GPU
 - for each warp, the 32 threads are executed **in parallel**
 - warps are executed **one after another**
 - but several warps can run simultaneously
 - up to 2 for CC 2.x, up to 6 for CC 3.x

Thread Hierarchy



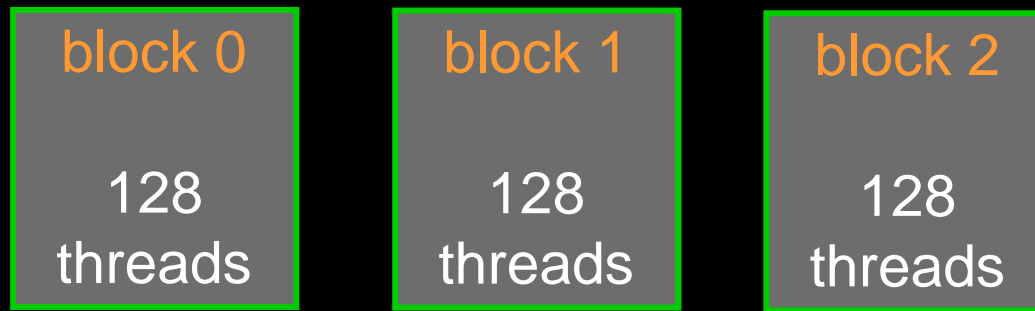
Blocks execute on Multiprocessors

- Each block is executed on one Multiprocessor (SM)
- Several blocks per SM possible



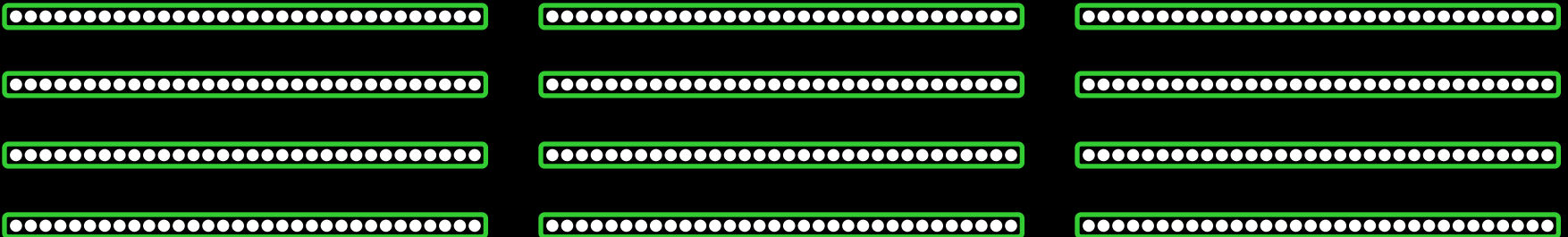
Execution on each Multiprocessor

- Assume there are three blocks on one SM, with 128 threads per block:



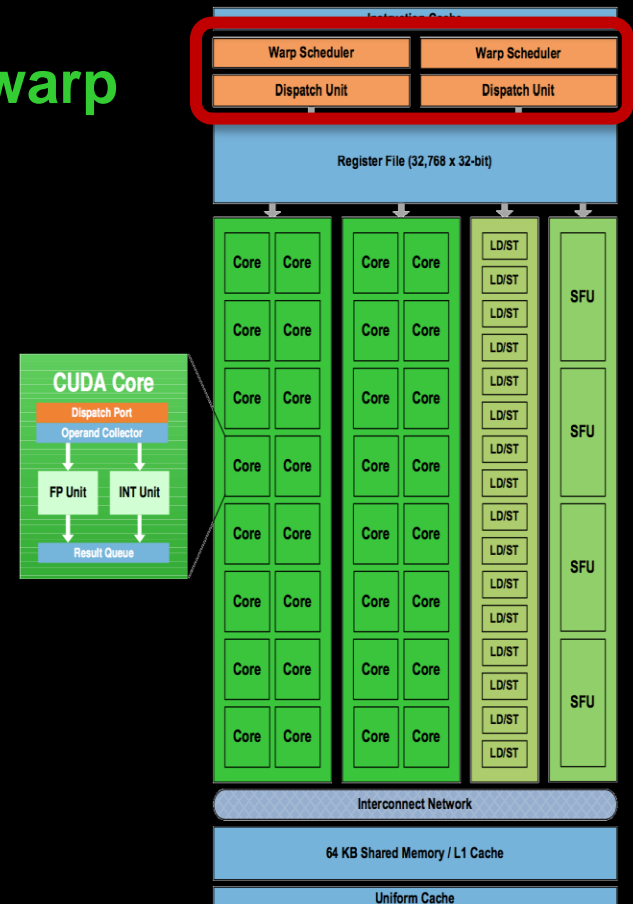
Execution on each Multiprocessor

- Threads **from all blocks** are divided into **warps**
- In our example:
 - 4 warps from every block (128 threads/32)
 - **12 warps overall** on SM (3 blocks * 4 warps/block)
 - $12 * 32 = 384$ threads

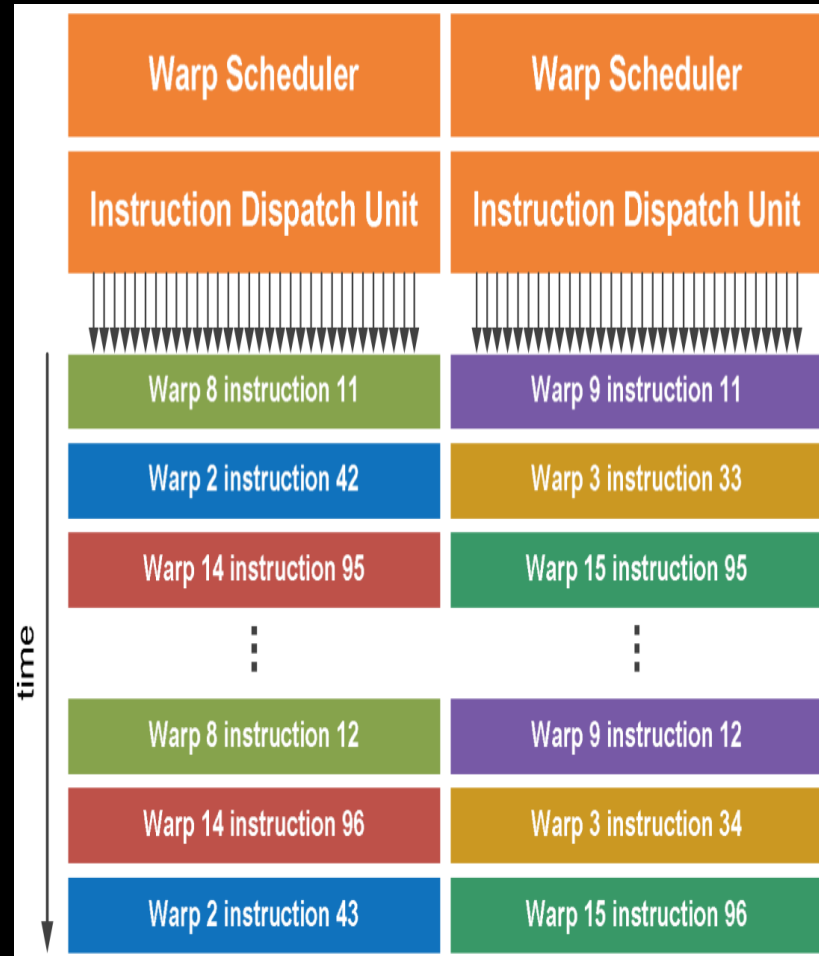
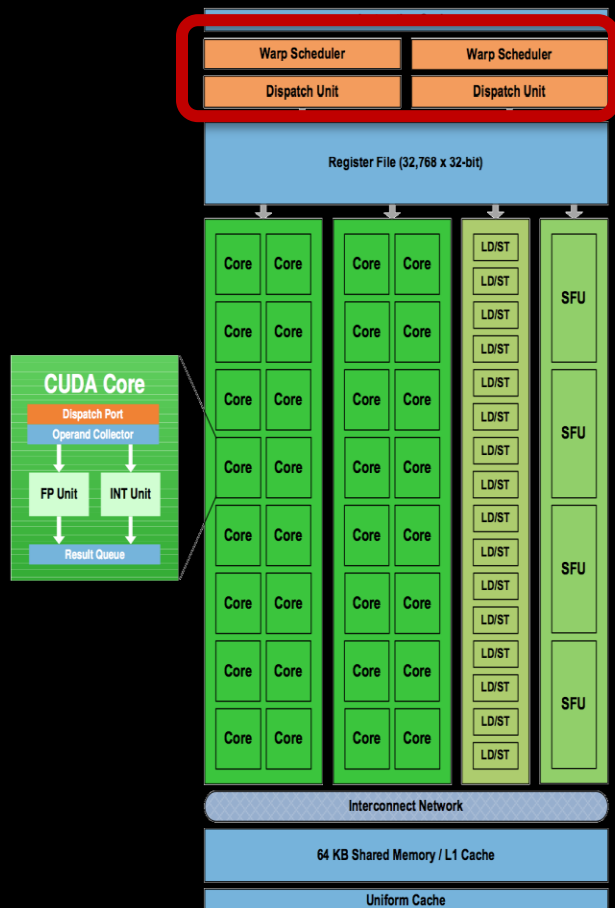


Execution on each Multiprocessor

- At each clock cycle
 - each **warp scheduler** chooses a **warp** which is ready to be executed
- For each chosen **warp**
 - the next instruction is executed for all 32 threads of the warp
 - issued for execution to
 - CUDA Cores
 - or load/store units
 - or special function units
 - or texture units



Execution on each Multiprocessor



Branch Divergence

- All 32 threads in a warp execute **the same** instruction
 - always, no matter what

```
__global__ void kernel (float *result, float *input)
{
    int i = threadIdx.x + blockDim.x*blockIdx.x;
    if (input[i]>0)
        result[i] = 1.f;
    else
        result[i] = 0.f;
}
```

What if different paths
are taken **within a warp**?

Branch Divergence: Serialization

```
if (input[i]>0) result[i] = 1.f; else result[i] = 0.f;
```

- If threads diverge **within a warp** execution is **serialized**
 - all 32 threads must execute the same instruction
- **Each path is taken** by each of the 32 threads
- Threads which do not correspond to this path are marked as **inactive** during execution
- Divergence in different warps: no serialization

Branch Divergence: Serialization

```
if (input[i]>0) result[i] = 1.f; else result[i] = 0.f;
```

threadIdx.x:	0	1	2	3	4	5	6	7	8	...	31
input[i]:	7	23	-2	5	-1	66	24	-41	-3	...	18
input[i]>0:	T	T	F	T	F	T	T	F	F	...	T

