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

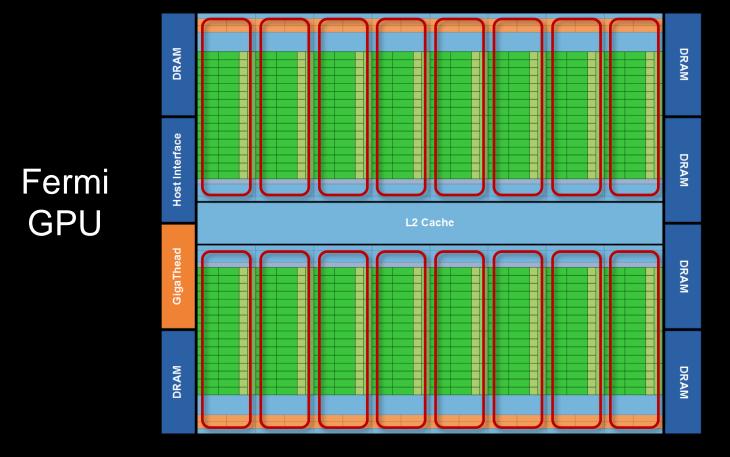
Deadline for exercises:

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

Sep	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



NVIDIA GPU Architecture

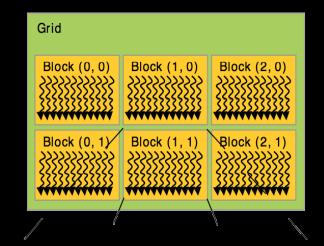


- 16 independent multiprocessors (SMs)
- No shared resources except global memory
- No synchronization, always work in parallel



- 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

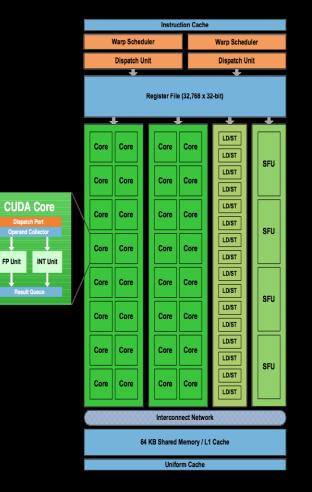
Thread Hierarchy



Block (1, 1)				
Thread (0, 0)	Thread (1, 0)	Thread (2, 0)	Thread (3, 0)	
/ ↓	~~~	~~~	\setminus	
Thread (0, 1)	Thread (1, 1)	Thread (2, 1)	Thread (3, 1)	
, *	>	~~~	₹,	
Thread (0, 2)	Thread (1, 2)	Thread (2, 2)	Thread (3, 2)	
, č	ž	Ę	Ę.	
✓	4	4	✓	

Blocks execute on Multiprocessors

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



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

block 0	block 1	block 2
128	128	128
threads	threads	threads

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

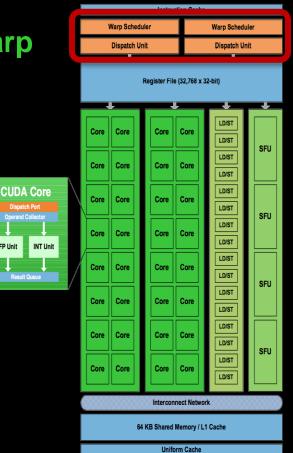


At each clock cycle

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



FP Unit

time

	Warp Scheduler	Warp Sc	Warp Scheduler	
	Dispatch Unit	Dispato	Dispatch Unit	
	Register File (32,768 x 32-bit)			
	Core Core Co	Core LD/ST		
	Core Core Co	ore Core LD/ST	SFU	
CUDA Core Dispatch Port Operand Collector	Core Core Co	LD/ST	SFU	
FP Unit INT Unit	Core Core Co	Core LD/ST		
Result Queue	Core Core Co	Core LD/ST	SFU	
	Core Core Co	Core LD/ST		
	Core Core Co	LD/ST	SFU	
	Core Core Co	Core LD/ST		
	Interconnect Network			
	64 KB Shared Memory / L1 Cache			
Uniform Cache				

Warp Scheduler	Warp Scheduler	
Instruction Dispatch Unit	Instruction Dispatch Unit	
Warp 8 instruction 11	Warp 9 instruction 11	
Warp 2 instruction 42	Warp 3 instruction 33	
Warp 14 instruction 95	Warp 15 instruction 95	
Warp 8 instruction 12	Warp 9 instruction 12	
Warp 14 instruction 96	Warp 3 instruction 34	
Warp 2 instruction 43	Warp 15 instruction 96	

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;

