CUDA
Introduction

Летняя студенческая школа 2008

Основные понятия

- **GPU**

  Graphical **P**rocessor **U**nit.

- **GPGPU**

  General **P**urpose computations on **GPU**s.

- **CUDA**

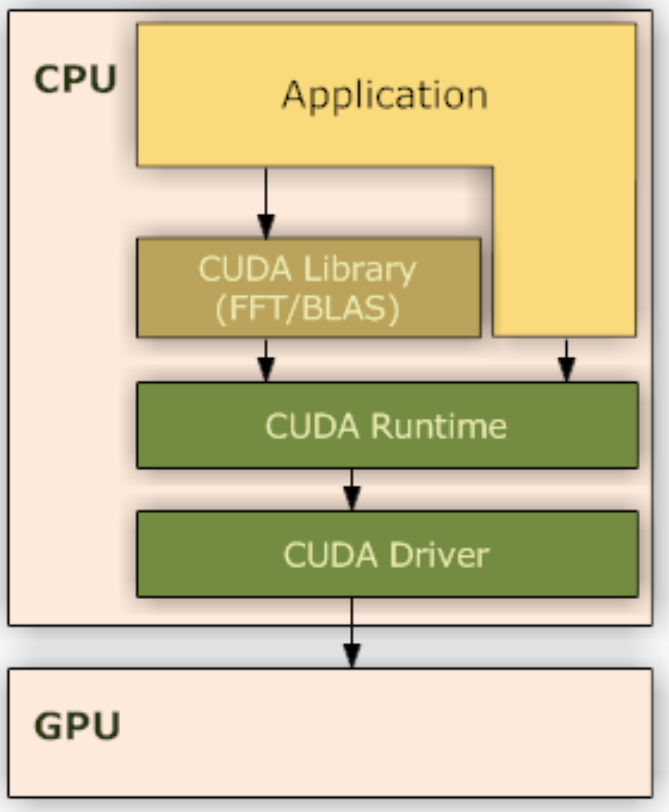
  Compute **U**nified **D**evice **A**rchitecture

  Программно-аппаратная архитектура для вычислений на GPU $\geqslant$ Nvidia GeForce 8400.

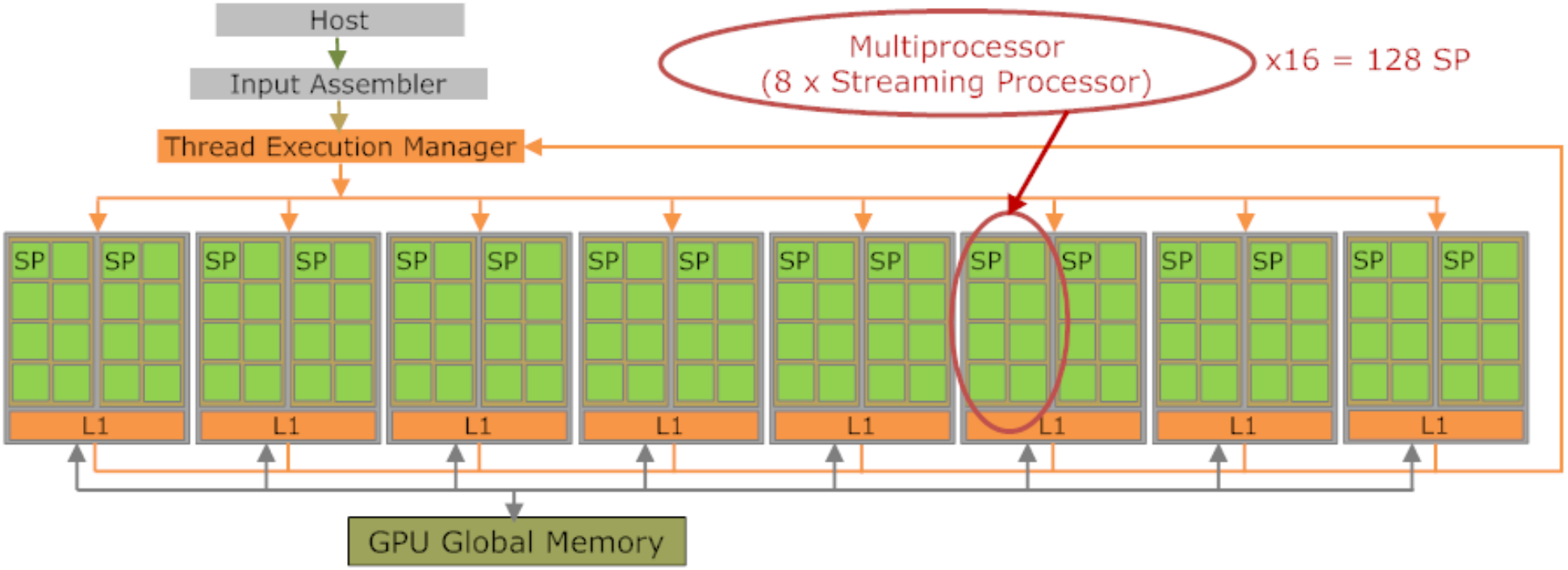
Compute Unified Device Architecture

Программно-аппаратная архитектура

- для вычислений общего назначения
- среда разработки является расширением языка C
- compiler & profiler & emulator
- unified API and GPU architecture
- Linux, Mac OS, Windows



GPU Architecture



GPU supported CUDA

GPU	**Multiprocessors**	**Memory**(Mb)	**Bus width** (bits)	**GHz**
GeForce 8800	16 (128)	368-768	256-384	1.35 - 1.6
Quadro FX 5600	16 (128)	1500	384	1.35 - 1.6
Tesla	30 (240)	4Gb-16Gb	512	1.3 - 1.5
GeForce GTX 260	24 (192)	1024	448	1.2
GeForce GTX 280	30 (240)	1024	512	1.3

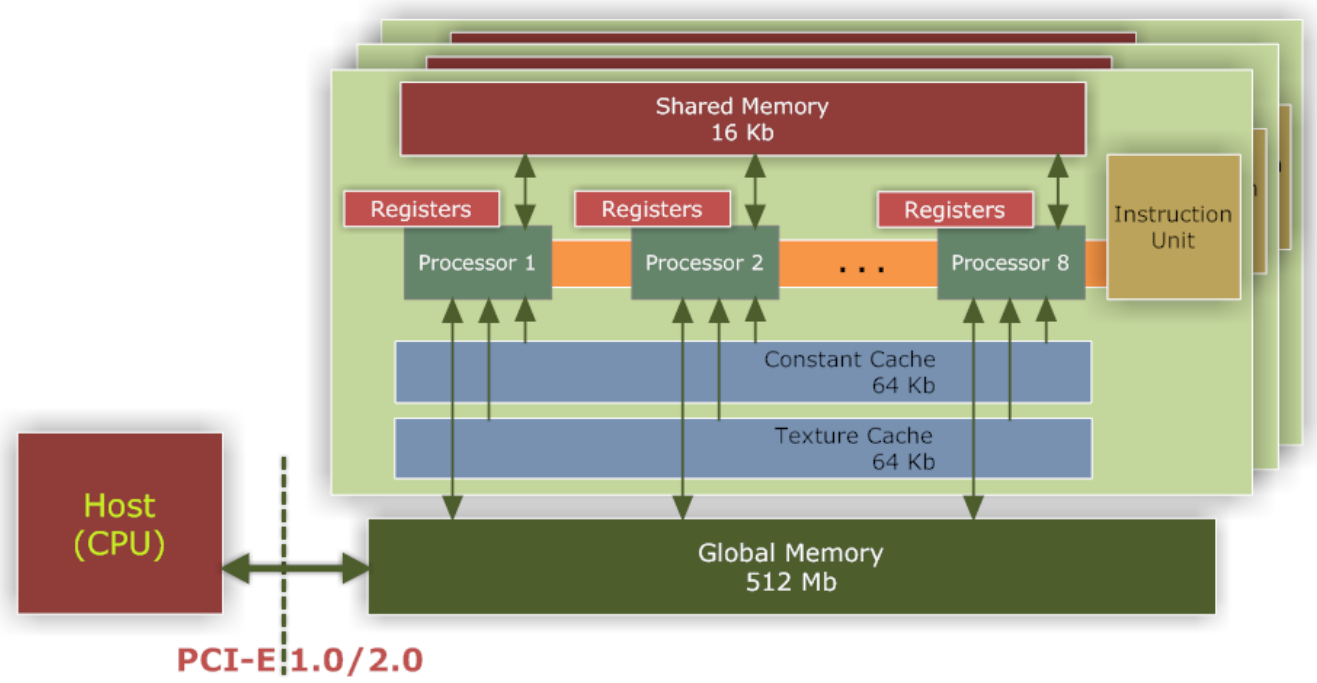
Bus width	**Memory bandwidth**
256	60 Gb/s
384	80 Gb/s
448	106 Gb/s
512	121 Gb/s

Processors	**GHz**	**Peak Performance**
128	1.3	332 Gflops
128	1.6	409 Gflops
192	1.2	460 Gflops
240	1.3	624 Gflops
240	1.5	720 Gflops

Processor	**Bandwidth**	**Peak Performance**
Core 2 Duo	20 Gb/s	24 Gflops
Cell	26 Gb/s	200 Gflops

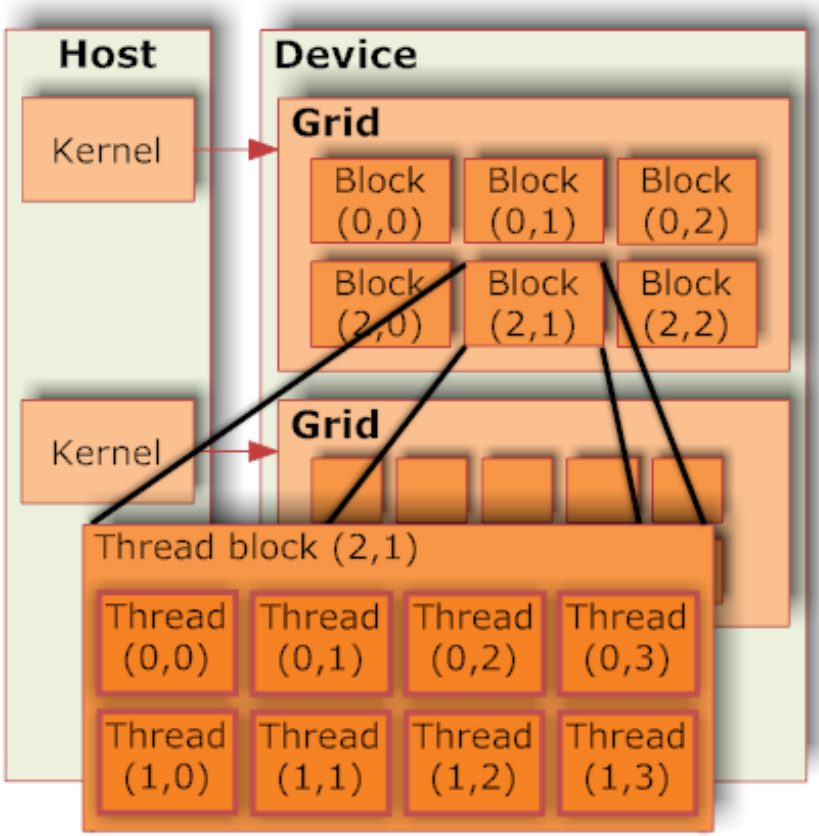
Multiprocessor's memory hierarchy

1. **registers** (r/w per-thread)
2. local memory (r/w, DRAM, uncached, per-thread)
3. **shared** memory (r/w, per-block)
4. **global** memory (r/w, DRAM, uncached, per-context)
5. **constant** memory (r-only, DRAM, cached, per-context)
6. **texture** memory (r-only, DRAM, cached, per-context)



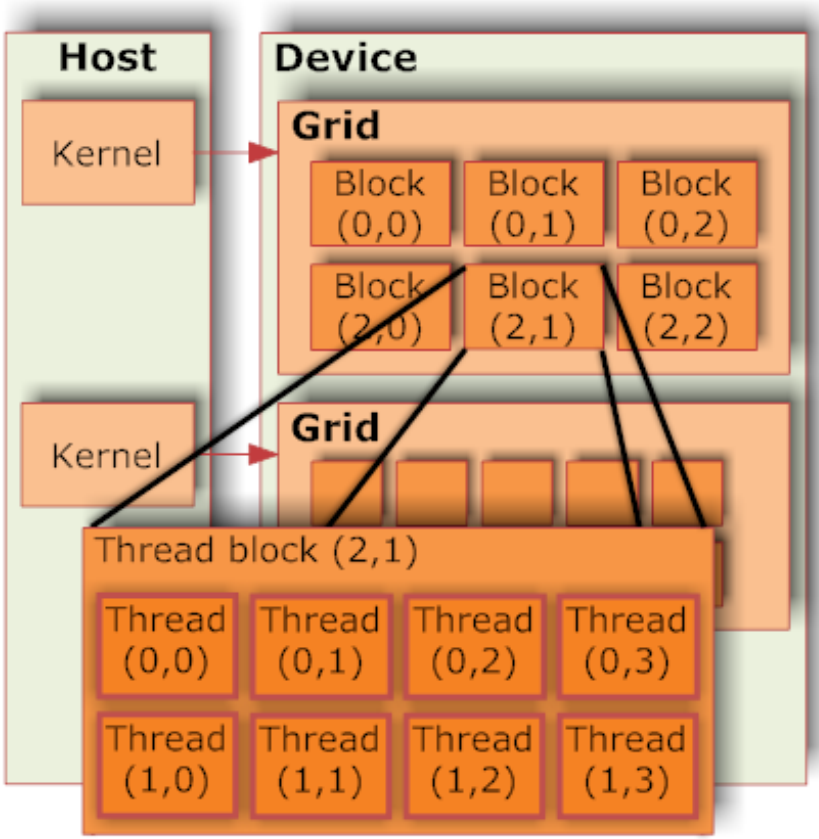
Programming model

- **Ядро** (kernel) — функция, исполняется над сеткой (grid) блоков потоков (threads block)
- **Блок потоков** (threads block) — набор потоков, способных общаться между собой посредством
  - **разделяемой памяти** (shared memory)
  - **точек синхронизации** (барьеры)
- Каждый поток и блок потоков имеет свои координаты



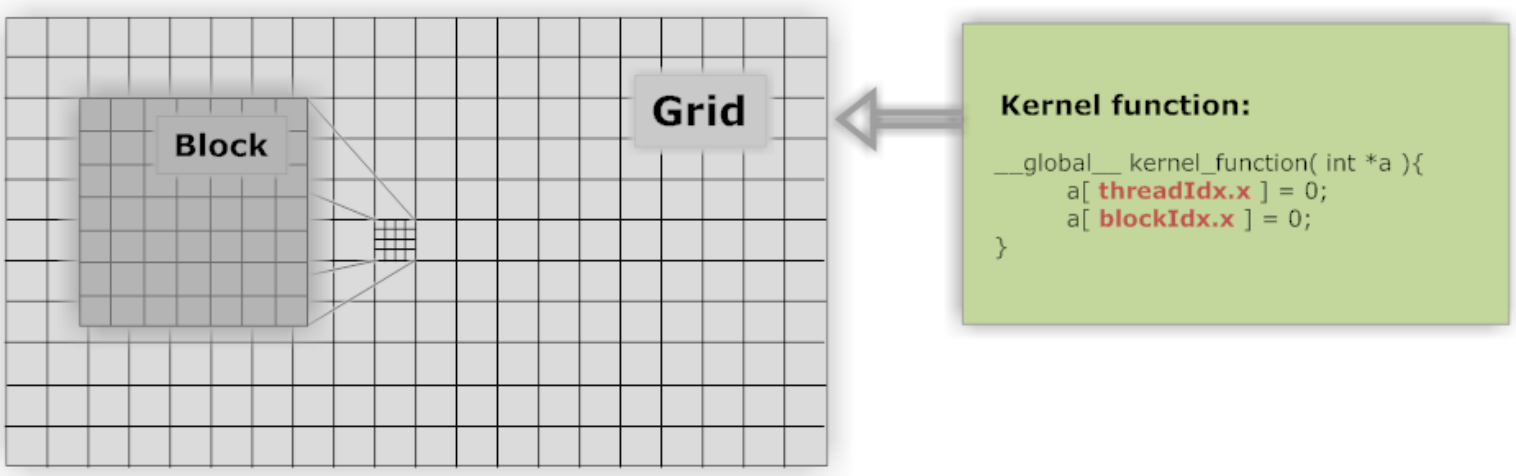
Execution model

- Блок потоков всегда исполняется только на одном мультипроцессоре
  - разделяемые переменные хранятся в shared memory
  - файл локальных регистров делится между обрабатываемыми мультипроцессором потоками
- На одном мультипроцессоре может обрабатываться несколько блоков потоков
  - разделяемая память и файл регистров делится между обрабатываемыми блоками



Итого

- **GPU** = array of multiprocessors
- **Multiprocessor** = **Streeming Processors** + **Shared Memory**
- **Kernel** = GPU-procedure
- **Grid** = array of thread blocks
- **Thread block** = array of SIMD-threads + shared memory communications



Why SIMD-threads?

- На каждом мультрипроцессоре может обрабатываться несколько блоков потоков
- На каждом такте мультипроцессор исполняет одну и ту же инструкцию над группой потоков, называемой warp
- Количество потоков в warp - warp size
- В случае условного ветвления, сначала исполняется одна ветвь над одной частью warp, затем другая над оставшейся.

global memory: should be aligned

shared memory: bank conflicts

CUDA Development

- Runtime API
  - memory management (allocate, free, copy, asynchronous)
  - execution management (execute, synchronize)
  - stream management (create, destroy, synchronize)
  - event management (create, destroy, record, get-elapsed-time)
- C language extensions
  - functions definition (device, kernel, host)
  - kernel call (grid and block size)
  - variable definition (device, constant, shared, register, align)
  - new types ([u]int[1..4], [u]float[1..4], [u]char[1..4])