Основы технологии CUDA

Лекция 3: Новое в CUDA. Основы работы с глобальной памятью

Автор курса:

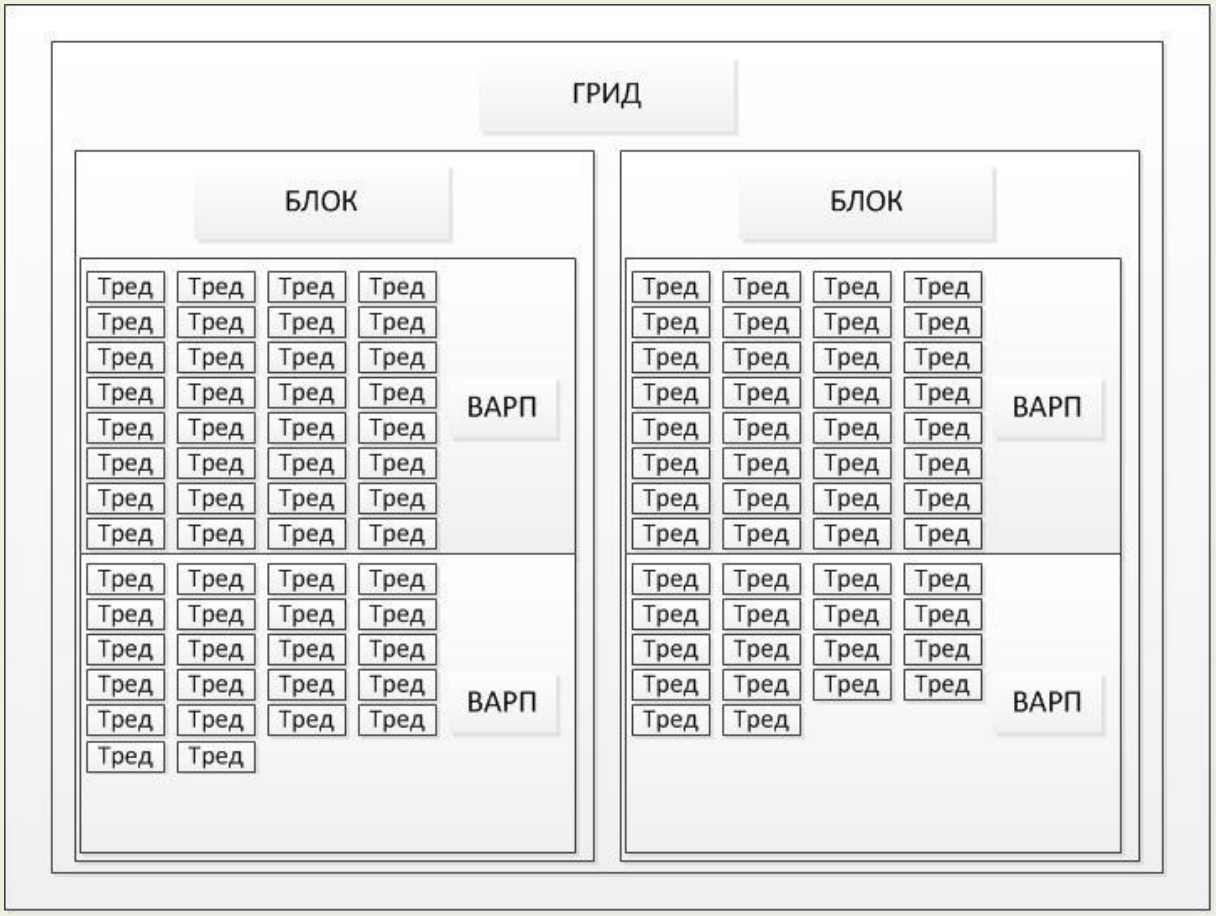
- Казённов Андрей

Повторим основные термины курса

- **Ядро (Kernel)** – Параллельная часть алгоритма, выполняется на гриде.
- **Грид (Grid)** – объединение блоков, которые выполняются на одном устройстве.
- **Блок (Block)** – объединение потоков, которое выполняется целиком на одном SM. Имеет свой уникальный идентификатор внутри грида.
- **Тред (Thread)** – единица выполнения программы. Имеет свой уникальный идентификатор внутри блока.
- **Варп (Warp)** – 32 последовательно идущих треда, выполняется физически одновременно.

Основы технологии CUDA

Схематическое изображение иерархии



Ограничения на размеры грида и блока

	Грид	**Блок**
X	65536	512
Y	65536	512
Z	1	64
Всего	4294967296	512

Место выполнения

- **Грид** выполняется на всей графической карте одновременно. Нет возможности указать, например, только 2 TPC из 10.

- Одновременно на графической карте может выполняться несколько ядер, а значит и гридов.

- **Блок** выполняется целиком на одном SM. Независимо от его размера.

- На одном SM одновременно может выполнятся до 8ми блоков. Количество блоков определяется ограничениями по регистрам и разделяемой памяти

Новые спецификаторы переменных

Спецификатор	Находится	Доступна	Вид доступа
__device__	device	device	R
__constant__	device	device / host	R / W
__shared__	device	block	RW / **__syncthreads()**

- **__device__** -- аналог **const** на CPU
- **__constant__** **--** заполняется с хоста, представляет из себя дополнительный способ передачи параметров на видеокарты
- **__shared__** **--** задает разделяемую память. Не может быть инициализирован при объявлении

Основы технологии CUDA

Новые спецификаторы функций

Спецификатор	Выполняется на	Может вызываться из
__device__	device	device
__global__	device	host
__host__	host	host

- **__device__** -- функция, выполняющаяся и вызывающаяся с устройства.

- **__global__** **--** задает ядро, выполняется на устройстве, вызывается с хоста. Всегда возвращает **void.** Применяется обособлено.

- **__host__** **--** обычная C/C++ функция, вызывается и выполняется на хосте. Может применяться совместно с __device__

Ограничения

В старых версиях CUDA < 3.0 функции не могут использовать

- static переменные
- Рекурсию
- Переменное число аргументов

От всех, кроме __global__ нельзя взять адрес.

Спецификаторы переменных не могут быть полями структур или union

Новые типы данных

- (u/) char, char2, char3, char4
- (u/) int, int2, int3, int4
- float, float2, float3, float4
- longlong, longlong2
- double, double2

- char2 a = make_char2('a','b');
  printf("%c %c", a.x, a.y);
- float4 b =make_float4(1.0, 2.0, 3.0, 4.0);
- printf("%f %f", b.x, b.y, b.z, b.w);

Новые типы данных

Специальный тип данных для задания параметров ядра dim3

- Обладает полноценным коструктором
- Основан на типе uint3
- Незаданные переменные заполняются 1
- dim3 block = dim3 (16,16, 2);
- dim3 grid = dim3 (1000)

Директива запуска ядра

<<<>>>

kernel <<<dim3 *grid*, dim3 *block*, size_t shared, stream_t *stream*>>> (data);

- dim3 *grid – конфигурация грида*
- dim3 *block – конфигурация блока*
- size_t shared,stream_t – количество динамически выделяемой разделяемой на блок
- stream_t *stream – поток, в котором надо запустить ядро*

Встроенные переменные

В любом ядре доступны следующие переменные:

- **dim3** gridDim; - размерность грида
- **uint3** blockIdx; - координата блока в гриде
- **dim3** blockDim; - размер блока
- **uint3** threadIdx; - координата треда в блоке
- **int** warpSize; - размер варпа

Пример задания и запуска ядра

```
#define BS 256 // Размер блока
#define N 1024 //Всего элементов для расчета
__global__ void kernel (int* data){
      int idx = blockIdx.x * BS + threadIdx.x;
      …some code…
}

int main (){
       int* data;
      dim3 block = dim3(BS);
      dim3 grid = dim3(N / BS);
      …some code…
      kernel <<<grid, block>>> (data);
…some code…
}
```

Основы технологии CUDA

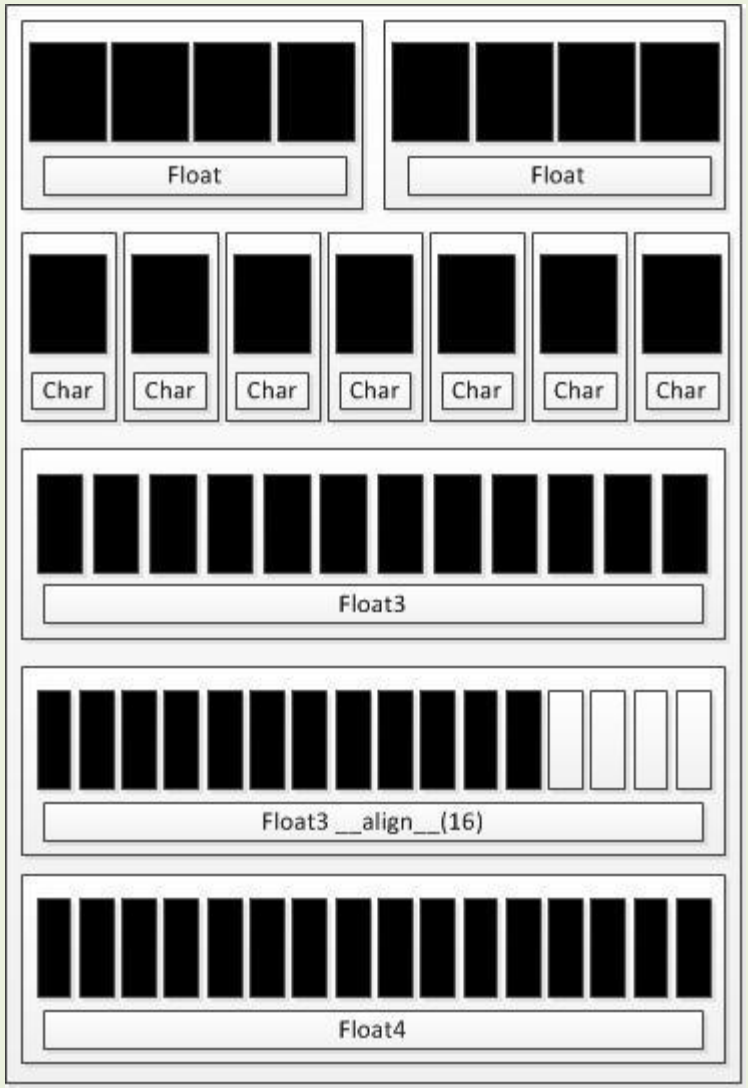
Правила чтения данных из памяти

- Обращение к памяти происходит 32/64/128-битовые слова

- При обращении к t[i]
  - sizeof(t [0]) равен 4/8/16 байтам
  - t [i] выровнен по sizeof (t [0])

- **Вся выделяемая память всегда выровнена по 256 байт**

- **При использовании типов данных не кратных** 4/8/16 желательно использовать **__align__(size)**



Типы памяти в CUDA

Тип памяти	Доступ	Уровень выделения	Скорость работы
Регистры	R/W	Per-thread	Высокая(on-chip)
Локальная	**R/W**	**Per-thread**	**Низкая (DRAM)**
Shared	R/W	Per-block	Высокая(on-chip)
Глобальная	**R/W**	**Per-grid**	**Низкая (DRAM)**
Constant	R/O	Per-grid	Высокая(L1 cache)
Texture	R/O	Per-grid	Высокая(L1 cache)

Особенности работы с глобальной памятью

- GPU умеет объединять ряд запросов к глобальной памяти в один блок (транзакцию)
- Независимо происходит для каждого half-warp’а
- Длина блока должна быть 32/64/128 байт
- Блок должен быть выровнен по своему размеру

Особенности работы с глобальной памятью: объединение

CC 1.0, 1.1

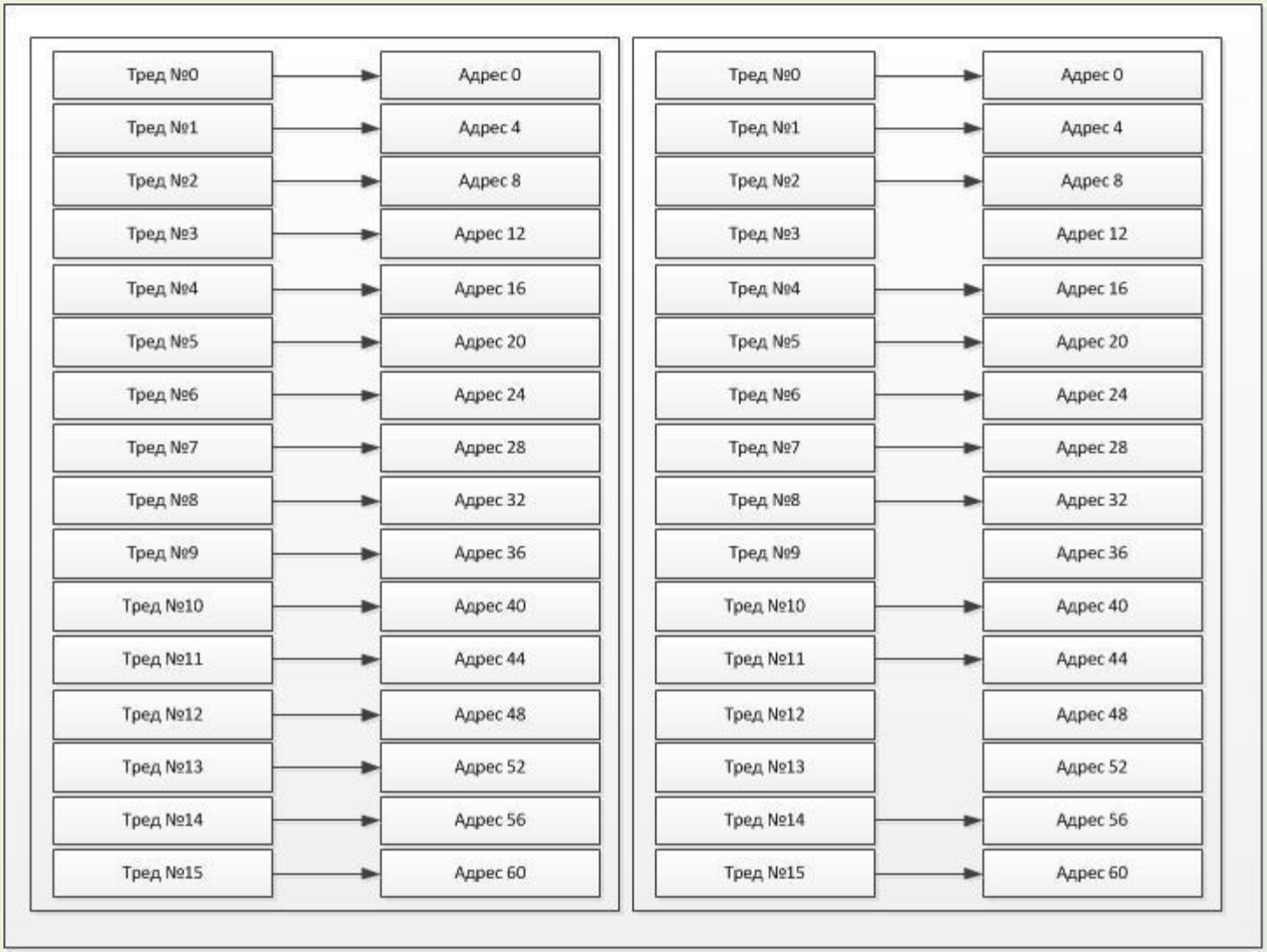
- Нити обращаются к
  - 32-битовым словам, давая 64-байтовый блок
  - 64-битовым словам, давая 128-байтовый блок
- Все 16 слов лежат в пределах блока
- k-ая нить half-warp'а обращается к k-му слову блока

CC >= 1.2

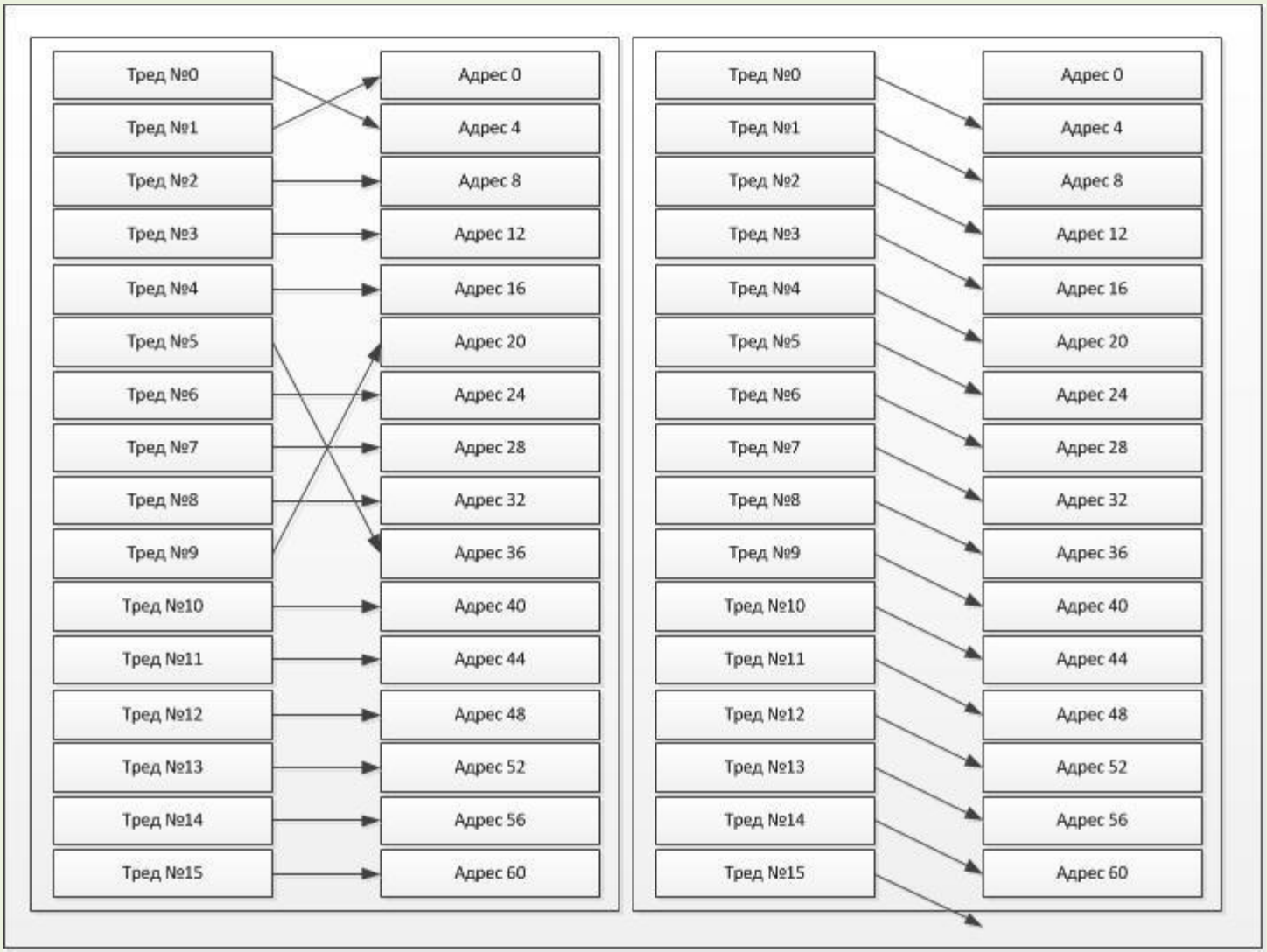
- Нити обращаются к
  - 8-битовым словам, дающим один 32-байтовый сегмент
  - 16-битовым словам, дающим один 64-байтовый сегмент
  - 32-битовым словам, дающим один 128-байтовый сегмент
- Получающийся сегмент выровнен по своему размеру

Основы технологии CUDA

Объединение есть

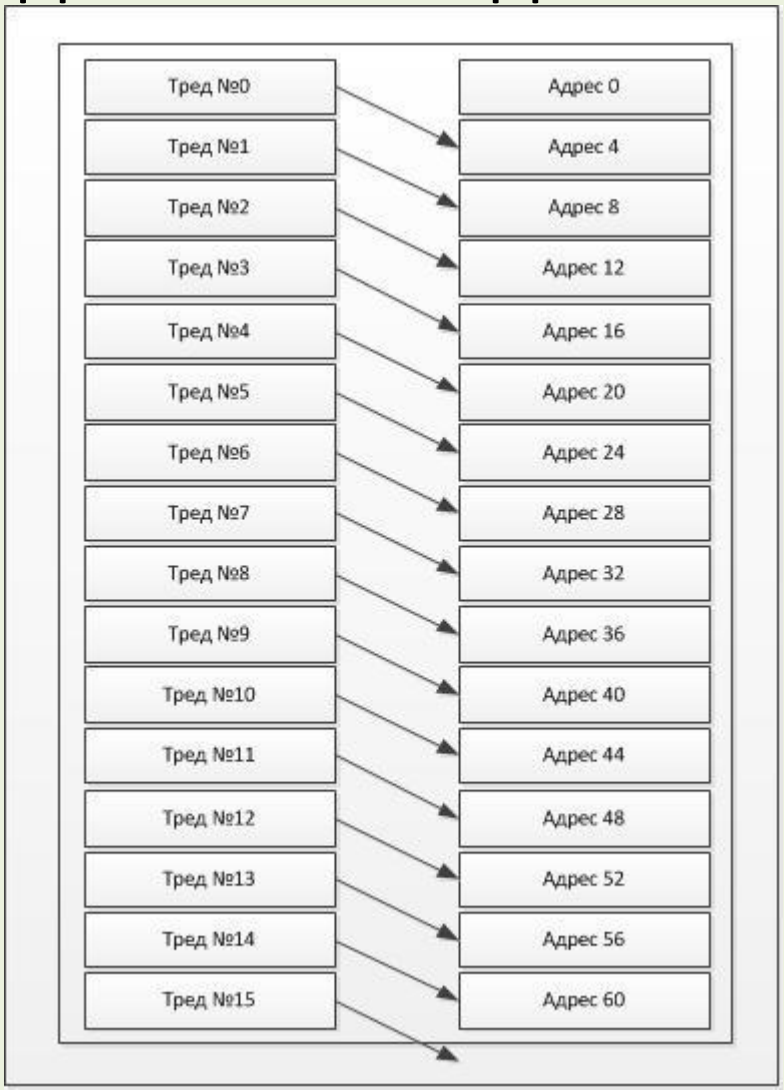


Объединения нет для CC 1.0, 1.1



Основы технологии CUDA

Объединения нет для CC >= 1.2



Ресурсы курса

Сайт: HPC.MIPT.RU

Раздел образование

Автор курса:

Казённов Андрей

- E-mail: kazennov@gmail.com
- ICQ: 622774