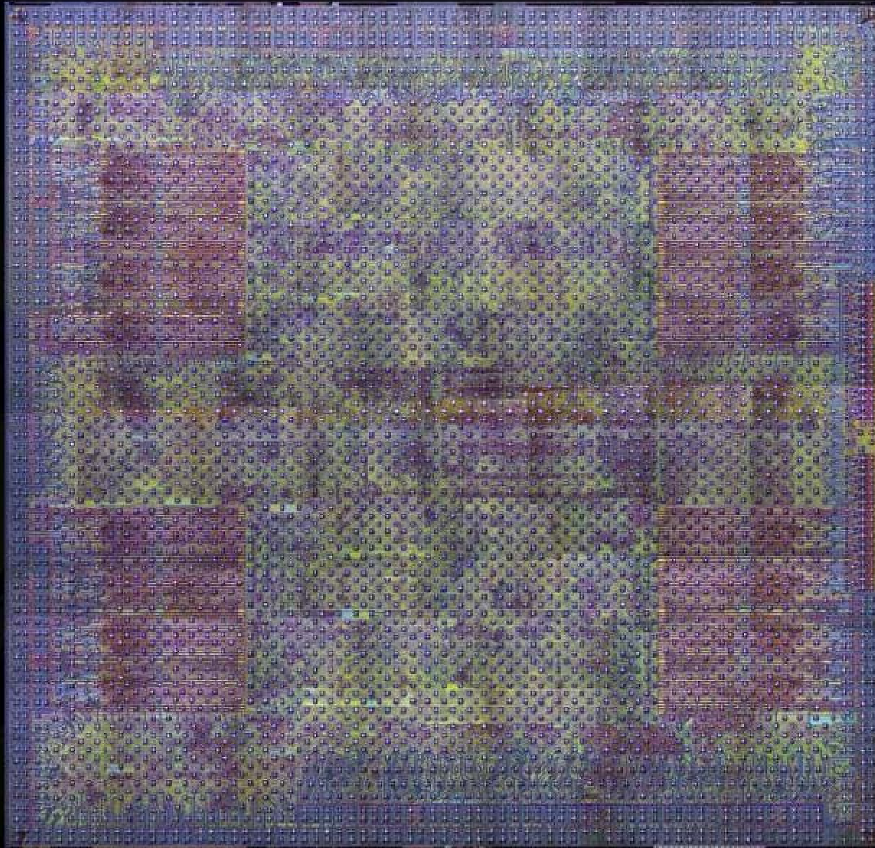
Архитектура и программирование потоковых многоядерных процессоров для научных расчётов

Лекция 4. Объединённая архитектура графических процессоров. Основные составные элементы аппаратной реализации GPU

NVIDIA®
World Leader in Visual Computing Technologies

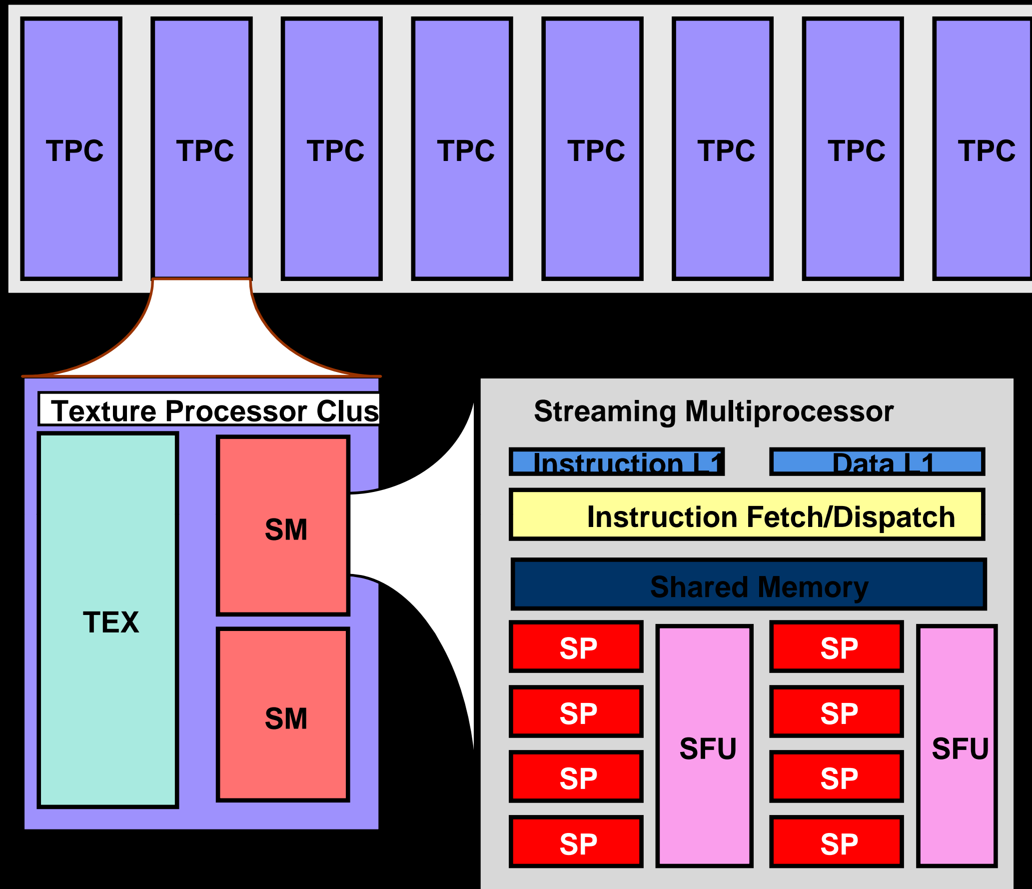
GPU процессоры



- Процессоры
  - G80 – 90nm (260/150W)
  - G92 - 65nm (110/~60W)
- Регулярная вычислительная структура – мало памяти
- Тактирован
  - NVClock (periphery logic)
  - Hot Clock (SM cores)
  - Mem Clock (DRAM access cirquits)
- Знание вычислительной архитектуры **необходимо** для успешного программирования

NVIDIA
World Leader in Visual Computing Technologies

Аппаратная архитектура GPU



- SPA - Streaming Processor Array (8x TPC)
- TPC - Texture Processor Cluster (2x SM + TEX)
- SM - Streaming Multiprocessor (8x SP)
  - Multi-threaded processor core
  - Fundamental processing unit for CUDA thread block
- SP - Streaming Processor
  - Scalar ALU for a single CUDA thread

NVIDIA®
World Leader in Visual Computing Technologies

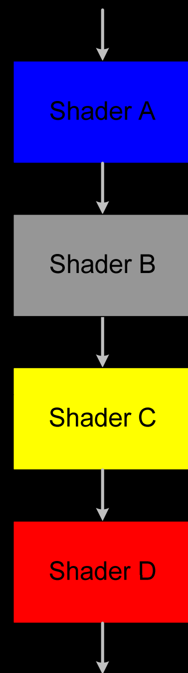
Цели проекта GeForce 8800

- **Традиционно GPU были кристаллами для наиболее эффективного исполнения одной программы**
- Разделение вычислений и операций с пмятью
  - Запуск произвольных (менее предсказуемых программ) – CUDA
    - Соотношение количества вычислений/переносов данных
    - Очерёдность вычислений/переносов данных
- Совмещение аппаратного обесп. для обсчёта векторных и растровых изображений
- Скаляризация ALU
- Облегчение работы компилятора

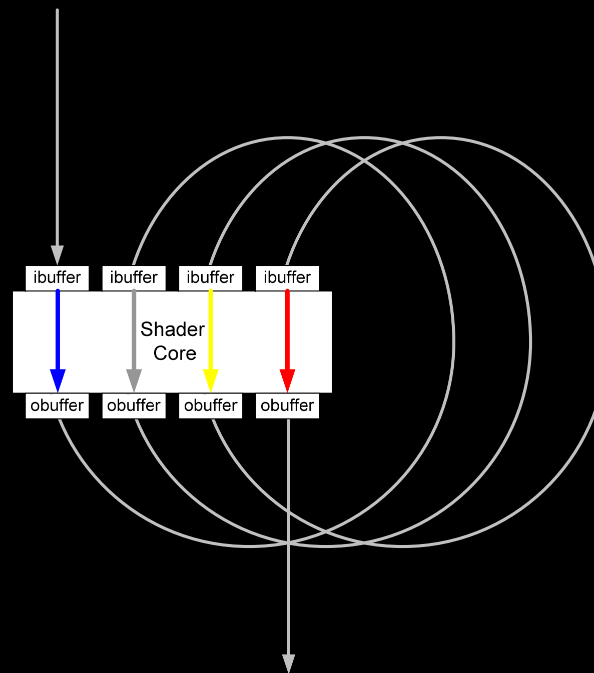
NVIDIA
World Leader in Visual Computing Technologies

Унифицированная архитектура

Discrete Design



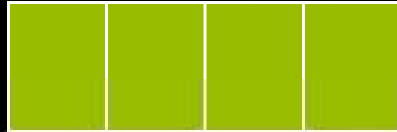
Unified Design



- Унификация исполняющих элементов
- Повышение программируемости
  - Последовательность операций жёстко задана архитектурой
  - Программируемая архитектура

Why unify? (1)

Vertex Shader

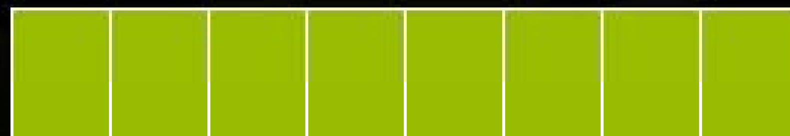


Pixel Shader

Heavy Geometry
Workload Perf = 4

Vertex Shader

Pixel Shader



Heavy Pixel
Workload Perf = 8

© NVIDIA Corporation 2006

Why unify? (2)

Unified Shader

Heavy Geometry
Workload Perf = 11

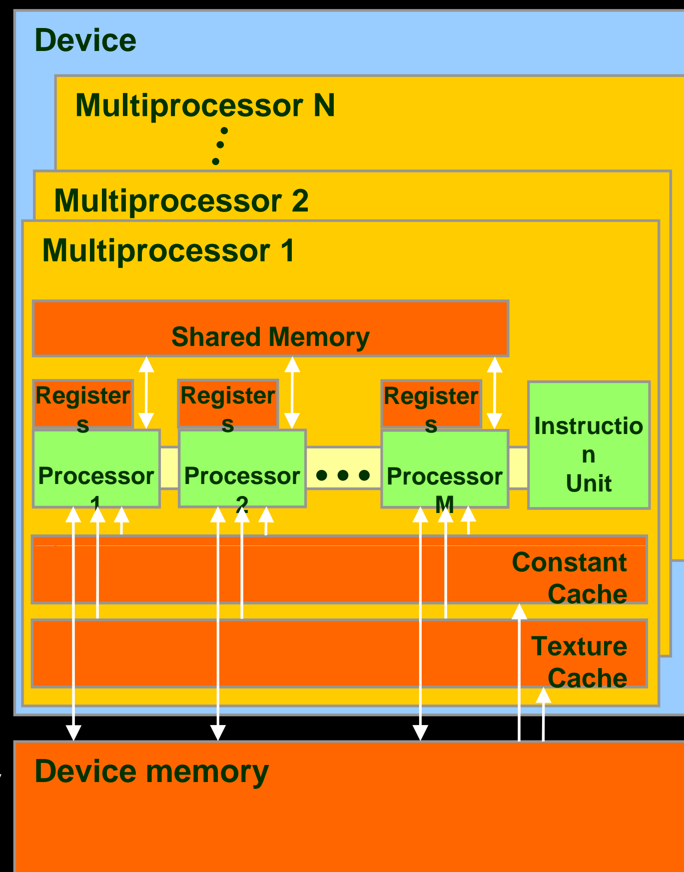
Unified Shader

Heavy Pixel
Workload Perf = 11

© NVIDIA Corporation 2006

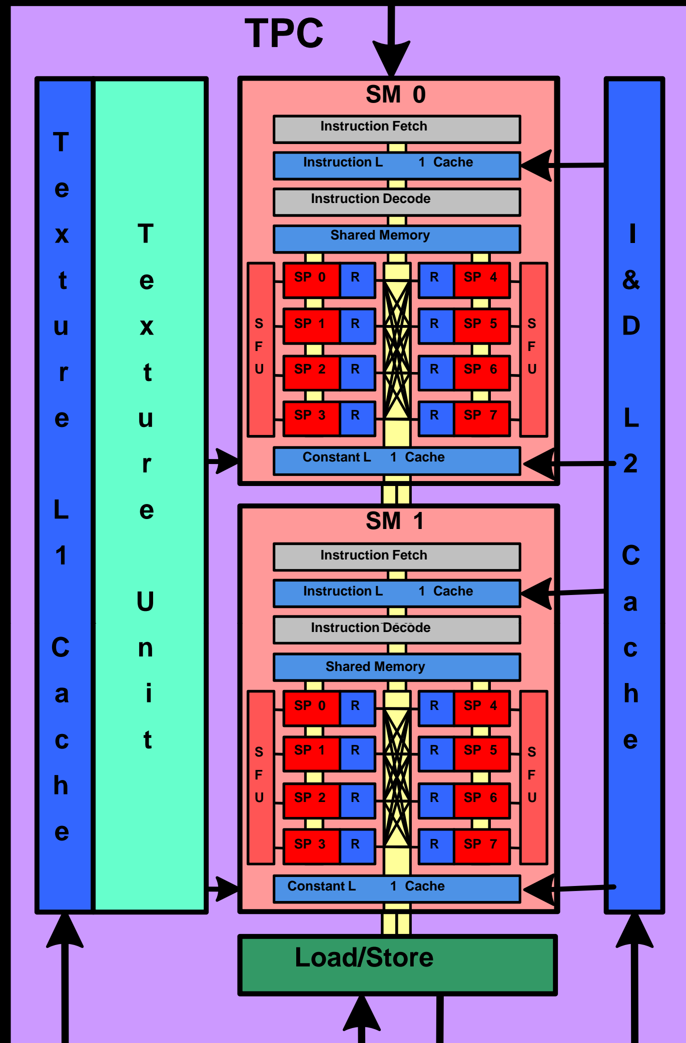
Что такое ВОРП (WARP)?

- Device делает 1 grid в любой момент
- SM обрабатывает 1 или более blocks
- Каждый Block разделён на SIMD группы, внутри которых одни и те же инструкции выполняются реально одновременно над различными данными (warps) warp size=16/32
- Связывание в ворпы детерминировано в порядке нарастания threadID
- threadID=TIDX.x+TIDX.y*Dx+TIDX.z*Dx*Dy
- **Важно! Полуворп** – первая или вторая половина ворпа



NVIDIA World Leader in Visual Computing Technologies

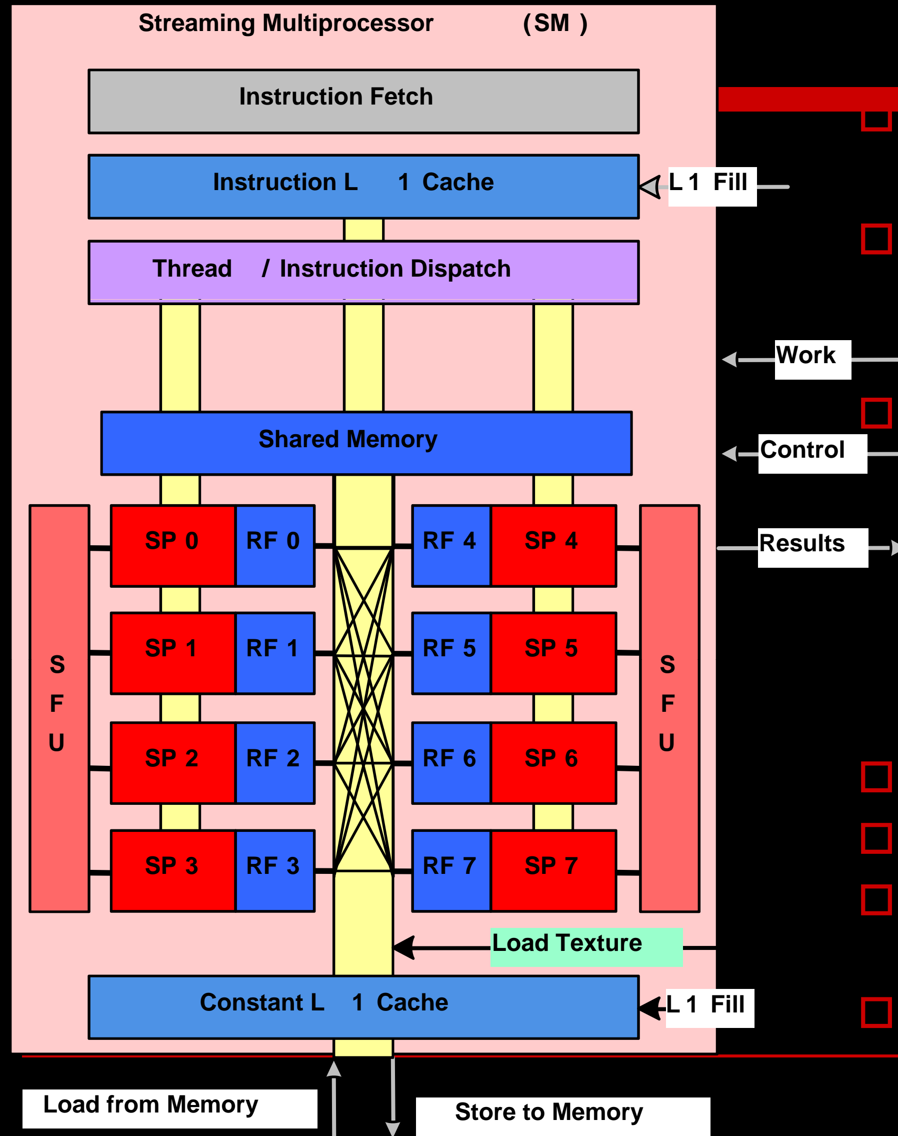
Texture Processor Cluster (TPC)



- TEX – Текстурный блок – логика адресации текстурных массивов в 1D, 2D, 3D
  - + L1 Кэш Текстур
- L2 Кэш инструкций и данных для обоих SM
- x2 Потоковых Мультипроцессора (Streaming Multiprocessor)
- x8 потоковых процессоров (streaming processors) = 8 MAD/clock cycle
- Регистры для хранения промежуточных результатов у выполняемых тредов => больше тредов лучше скрыты операции чтения, но межет не хватить регистров

NVIDIA®
World Leader in Visual Computing Technologies

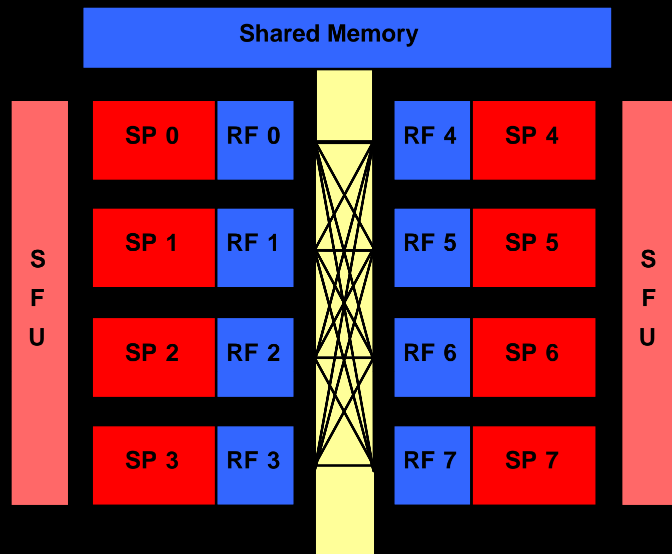
Streaming Multiprocessor (SM)



- 8 Streaming Processors (SP)
- 2 **Super Function Units** (SFU) – сложные функции (sin, cos, etc.)
- Много-поточная доставка инструкций
  - 1 - 512 потоков активны в кажд.момент
  - SIMD инструкции для ворпов 16/32 тредов **!!!! ВЕТВЛЕНИЯ**
- Hot clock = 1.35 GHz
- 20+ GFLOPS для каждого SP
- Локальный регистровый файл (RFn)
- 16 KB разделяемой памяти

NVIDIA
World Leader in Visual Computing Technologies

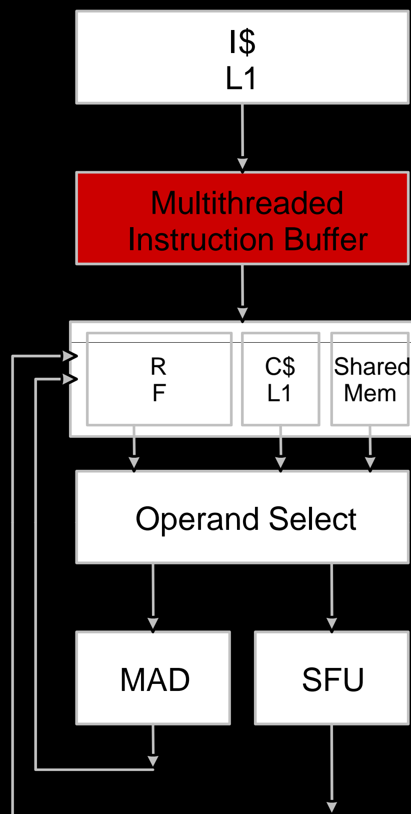
Streaming Processors (SP)



- Скалярный FP MAD ALU
- Место исполнения инструкции в 1 треде в 1 момент времени
- SFU разделены между SP
- Размер RF – 32KB Если переполнен – отправляется в локальную память
- Для оптимальной загрузки MAD/SFU конвейера ~ 8 варпов

NVIDIA
World Leader in Visual Computing Technologies

Буфер инструкций SM



- Буфер инструкций выбирает варп и инструкцию, которые будут исполнены в следующий момент времени
- Критерии выбора:
  - Готовность данных
  - Длительность исполнения ворпа (старые имеют приоритет)
- Активный ворп будет исполнять свои инструкции последовательно, пока не возникнет ситуация, мешающая
  - Нет промежуточных результатов для новой
  - Не прочитаны операнды из памяти
- Загрузка конвейера
  - Программно 32/16 тредов на варп
  - Аппаратно 8 SP в каждом SM

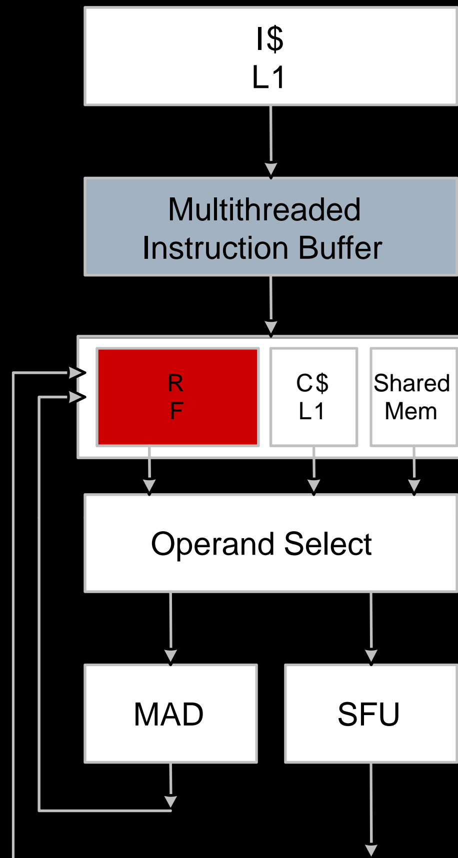
NVIDIA
World Leader in Visual Computing Technologies

Ветвления внутри ворпа

- **Ветвления разрушают SIMD структуру исполнения**
- Инструкции ложной ветви не исполняются для текущего треда
- Время теряется как если бы все треды ворпа прошли всеми возможными путями (последовательно)
- Компилятор может востанавливать точки ре-синхронизации, рассинхронизировавшихся тредов
- Программист может помогать используя __syncthreads();

NVIDIA
World Leader in Visual Computing Technologies

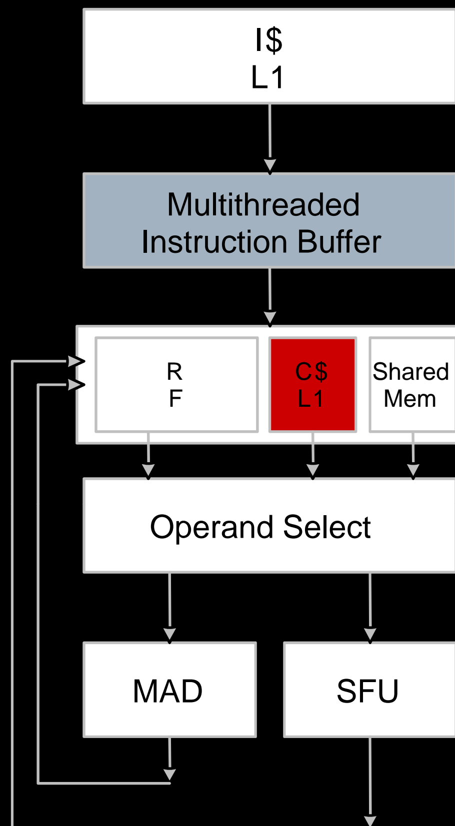
Регистровый файл SM



- В каждом SM регистровый файл
  - Размер = 32K
  - Распределён неравномерно между SP
- За один clock можно прочитать 4 операнда для каждого SP
- TEX и Load/Store могут читать и писать RF

NVIDIA®
World Leader in Visual Computing Technologies

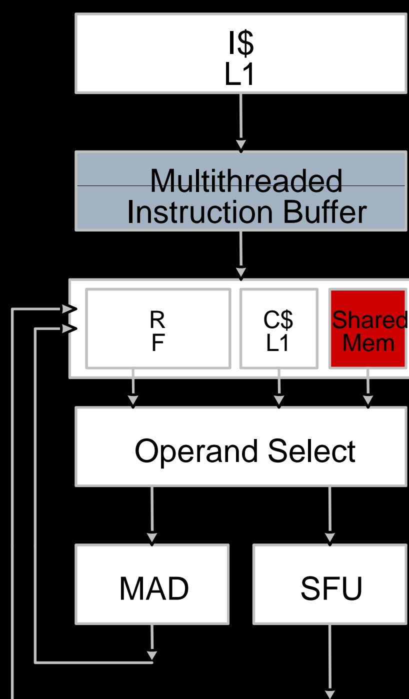
Память констант



- RO память констант находится в DRAM на плате
- Каждый из SM имеет L1 кэш (RO) в операциях с памятью констант
- Константы могут адресоваться
  - без индекса
  - Линейный индекс на основе threadID

NVIDIA
World Leader in Visual Computing Technologies

Разделяемая память

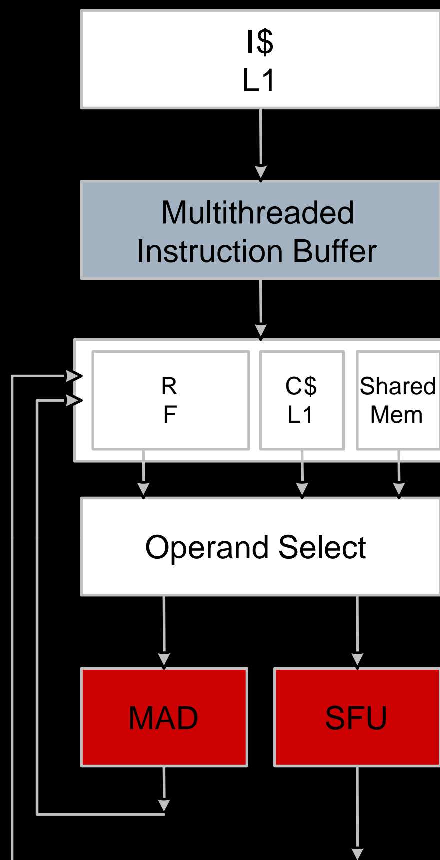


BANK0	BANK1	BANK2		BANK15
word 0	word 1	word 2		word 15
word 16	word 17	word 18		word 31
word 32	word 33	word 34		word 47
word 4080	word 4081	word 4082		word 4095

- В каждом SM – 16 k RW разделяемой памяти
- 16 банков 32-битных слов
- Последовательная адреса ячеек принадлежат последовательным банкам
- Обращения к разным банкам возможны одновременно
- Обращения с конфликтами реализуются как несколько полледовательных обращений
- Каждое чтение исполняется от тредов полуворпа

NVIDIA
World Leader in Visual Computing Technologies

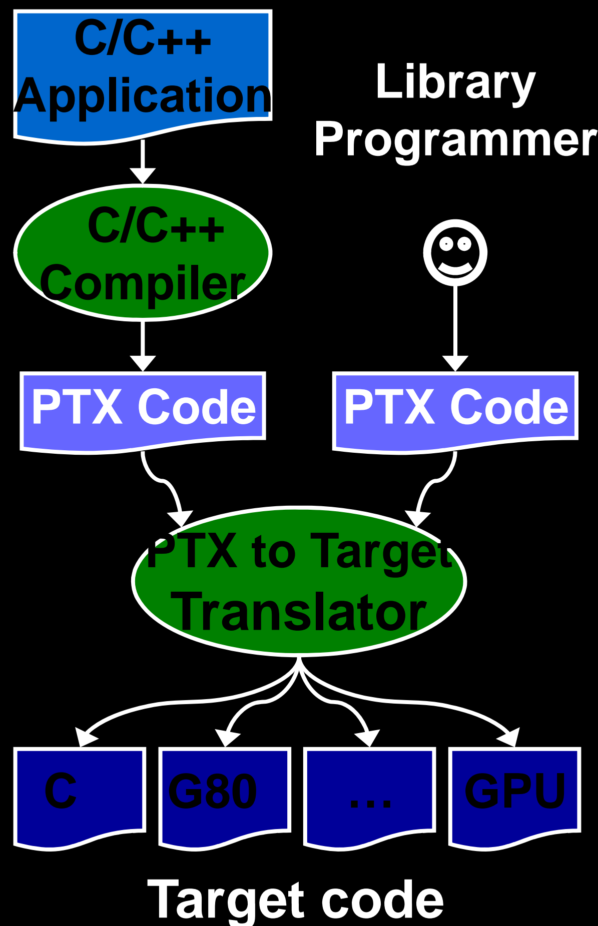
Конвейеры исполнения комманд



- Скалярный конвейер MAD (SPs)
  - FMUL,FADD,FMAD
  - Целочисленные операции, приведение типов
  - 1 инструкция за 1 clock
- Скалярный конвейер SFU
  - RCP,RSQ,LG2,EX2,SIN,COS
  - 1 инструкция за 4 clocks
  - можно также реализовывать FMUL, MOV
- TEX конвейер (RO Iдоступ к табулированным константам)
- LoaD / STore конвейер
  - Перенос содержимого регистров в/из локальную память
  - Доступ к локальной и глобальной памяти

NVIDIA
World Leader in Visual Computing Technologies

Parallel Thread eXecution Virtual Machine (PTX VM)



- Parallel Thread eXecution (PTX)
  - Virtual Machine, а также ISA
  - Программная модель
- ISA – Instruction Set Architecture
  - Variable declarations
  - Instructions and operands
- Транслятор – оптимизирующий компилятор
  - Трансляция PTX исполняемый код
- Драйвер видеокарты реализует VM runtime

NVIDIA
World Leader in Visual Computing Technologies

PTX код (пример1)

CUDA

```
float4 me = gx[gtid];
me.x += me.y * me.z;
```

PTX

```
ld.global.v4.f32  {$f1,$f3,$f5,$f7}, [$r9+0];
# 174        me.x += me.y * me.z;
mad.f32                $f1, $f5, $f3, $f1;
```

NVIDIA. World Leader in Visual Computing Technologies

PTX Код (пример 2)

- CUDA

```
__device__ void interaction(
            float4 b0, float4 b1, float3 *accel)
{
    r.x = b1.x - b0.x;
    r.y = b1.y - b0.y;
    r.z = b1.z - b0.z;
    float distSqr = r.x * r.x + r.y * r.y + r.z * r.z;
    float s = 1.0f/sqrt(distSqr);
    accel->x += r.x * s;
    accel->y += r.y * s;
    accel->z += r.z * s;
}
```

- PTX

```
sub.f32      $f18, $f1, $f15;
sub.f32      $f19, $f3, $f16;
sub.f32      $f20, $f5, $f17;
mul.f32      $f21, $f18, $f18;
mul.f32      $f22, $f19, $f19;
mul.f32      $f23, $f20, $f20;
add.f32      $f24, $f21, $f22;
add.f32      $f25, $f23, $f24;
rsqrt.f32    $f26, $f25;
mad.f32      $f13, $f18, $f26, $f13;
mov.f32      $f14, $f13;
mad.f32      $f11, $f19, $f26, $f11;
mov.f32      $f12, $f11;
mad.f32      $f9, $f20, $f26, $f9;
mov.f32      $f10, $f9;
```

NVIDIA
World Leader in Visual Computing Technologies

Итоги лекции

- В результате лекции студенты должны :
  - Понимать преимущества объединённой архитектуры нового поколения графических процессоров
  - Понимать принципиальные элементы архитектуры GPU процессоров
  - Иметь возможность применить практически знание архитектуры для оптимизации программ
  - Достаточные знания для начала самостоятельной работы