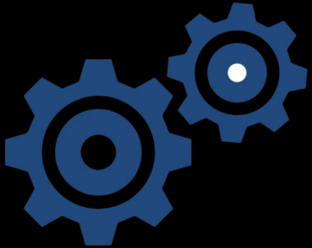
GPUs

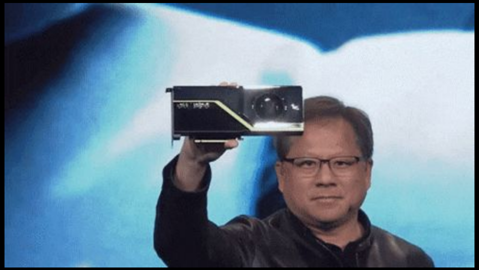
More than gaming

Agenda

Introduction



Architecture



Cuda

Practice

Intro

Why do these things even exist

Welcome to the 70’s

Not the 70’s we’re interested in

Welcome to the 70's

Not the 70's we're interested in

Welcome to the 70's

Now we talking

In a nutshell? Gaming.

You may have heard that gaming is what drove the industry forward to create more powerful dedicated hardware

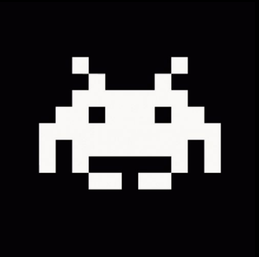
In a nutshell? Gaming.

You may have heard that gaming is what drove the industry forward to create more powerful dedicated hardware

And this is precisely right

The pre-GPU era

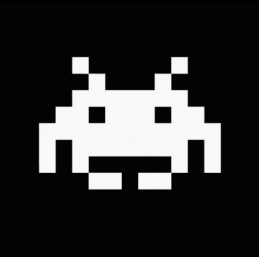
Early gpus were nothing more than super-specialized custom hardware to draw pixels on screen



The pre-GPU era

Early gpus were nothing more than super-specialized custom hardware to draw pixels on screen

This would free up the more slow cpu so that a not-so-powerful overall cpu was needed

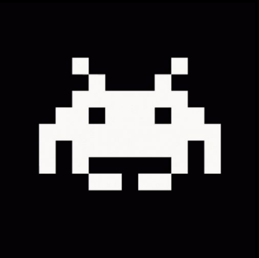


The pre-GPU era

Early gpus were nothing more than super-specialized custom hardware to draw pixels on screen

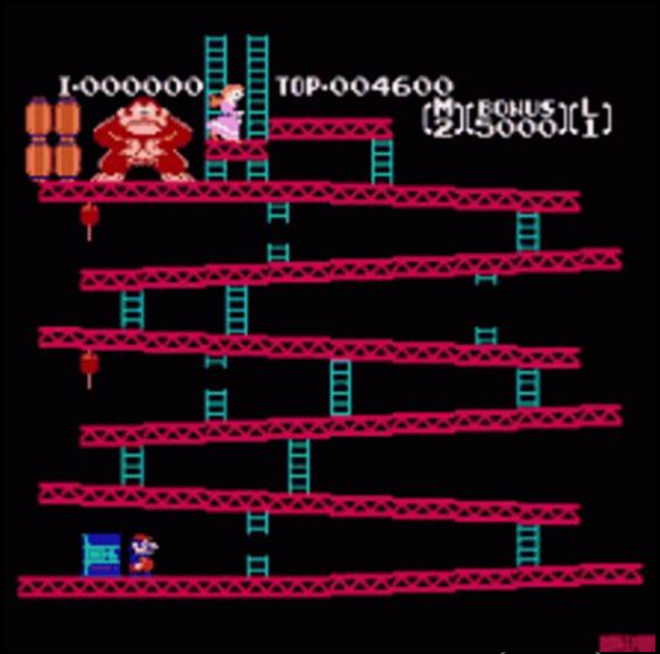
This would free up the more slow cpu so that a not-so-powerful overall cpu was needed

Making the hardware for the cabinet way less expensive



Things escalated quickly

In the 80’s Gpus could perform way more tasks meeting the demand to display color



Things escalated quickly

In the 80's Gpus could perform way more tasks meeting the demand to display color

In the 90's 3d rendering was already possible by such devices

Moar than just Moar performance

I would like to briefly break down what a specialized chip would do to make something like crash bandicoot possible.

Triangles

I bet you already know that each 3d model is composed of triangles.

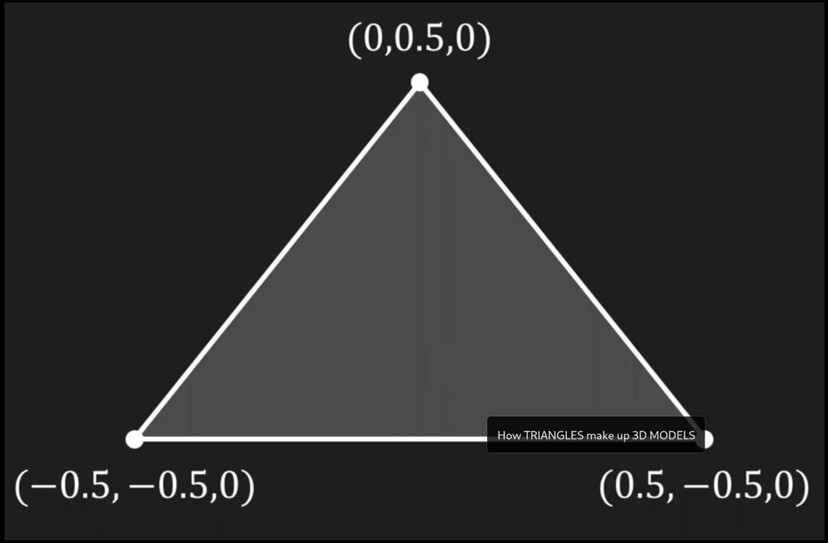
Triangles

I bet you already know that each 3d model is composed of triangles.

Such shapes hold (obviously) 3 vertices

Coordinates and operations

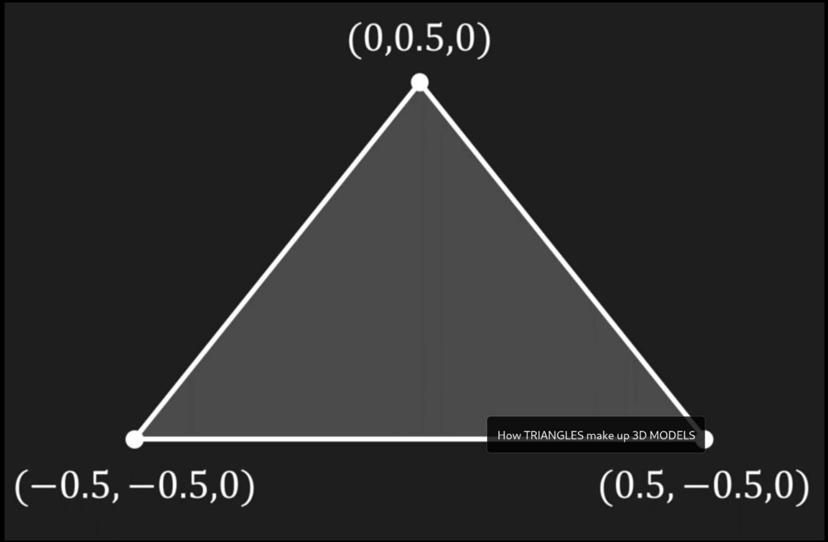
Each vertice holds 3 coordinates in a 3d environment



Coordinates and operations

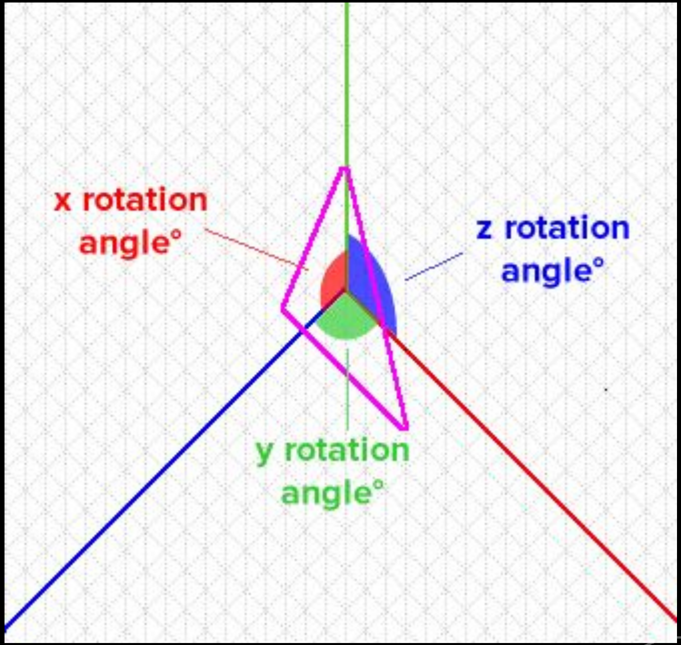
Each vertice holds 3 coordinates in a 3d environment

On each frame such triangles move, in particular



Coordinates and operations

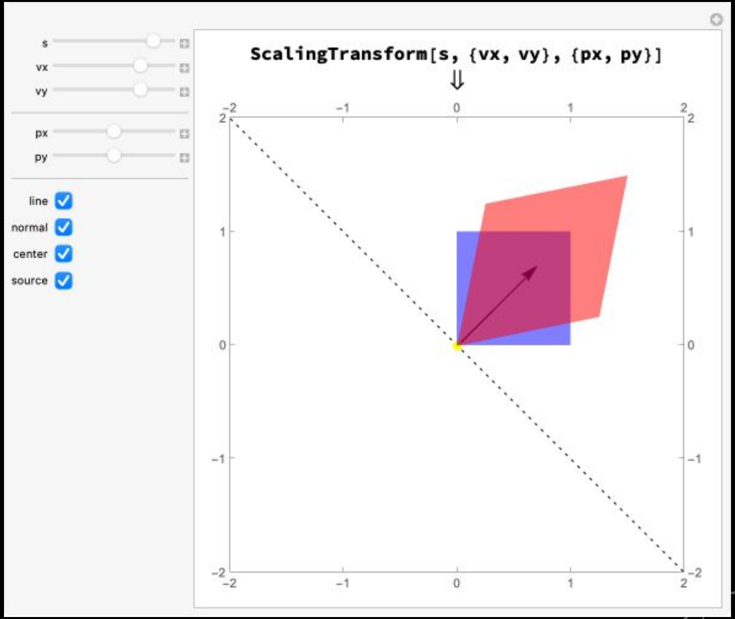
They rotate.



Coordinates and operations

They rotate

They shrink/enlarge (scale)

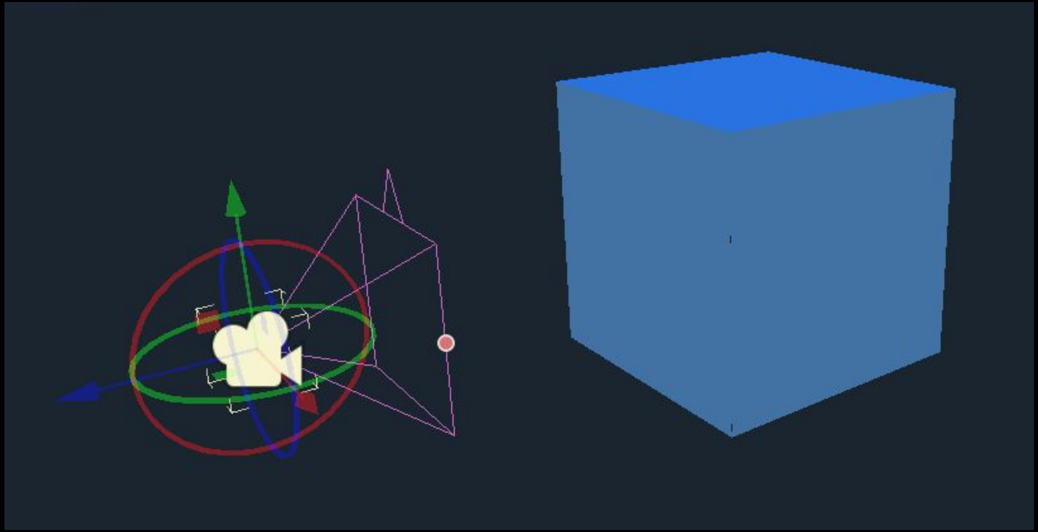


Coordinates and operations

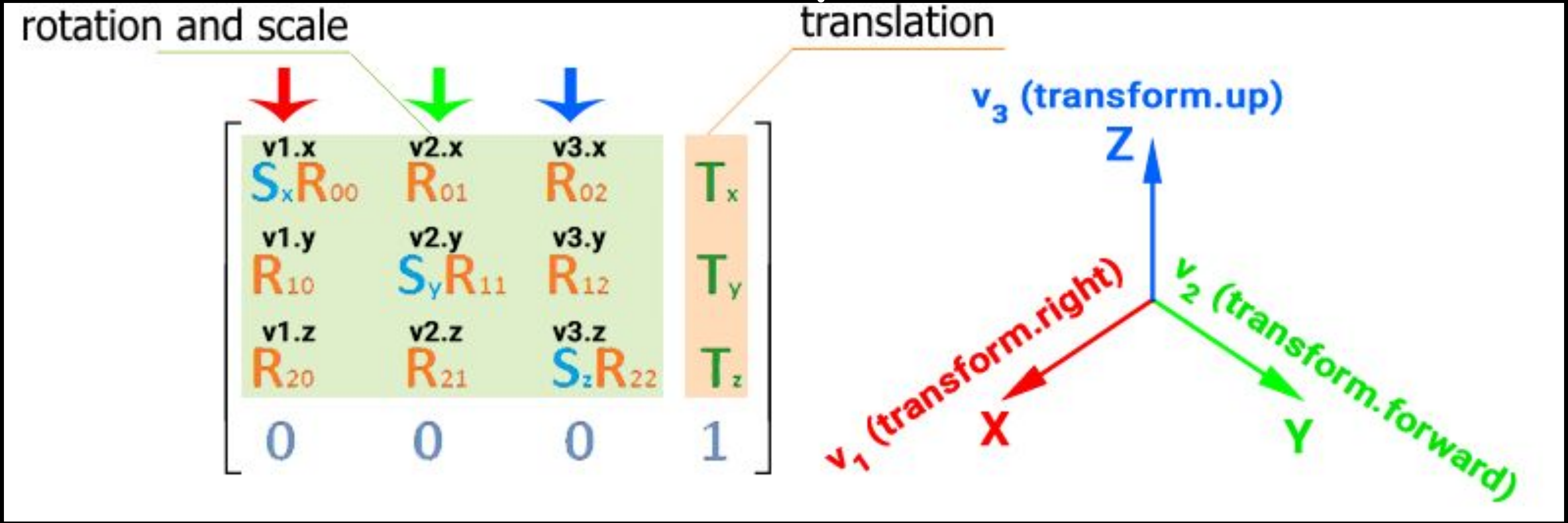
They rotate

They shrink/enlarge (scale)

They get projected onto the screen



Coordinates and operations



All of this with the help of simple first year linear algebra maths: matrices

The need for speed™

Scenes in the 90’s typically contained about 40K+ triangles

The need for speed™

Scenes in the 90’s typically contained about 40K+ triangles

Each frame needed to be computed in about 33.33 ms

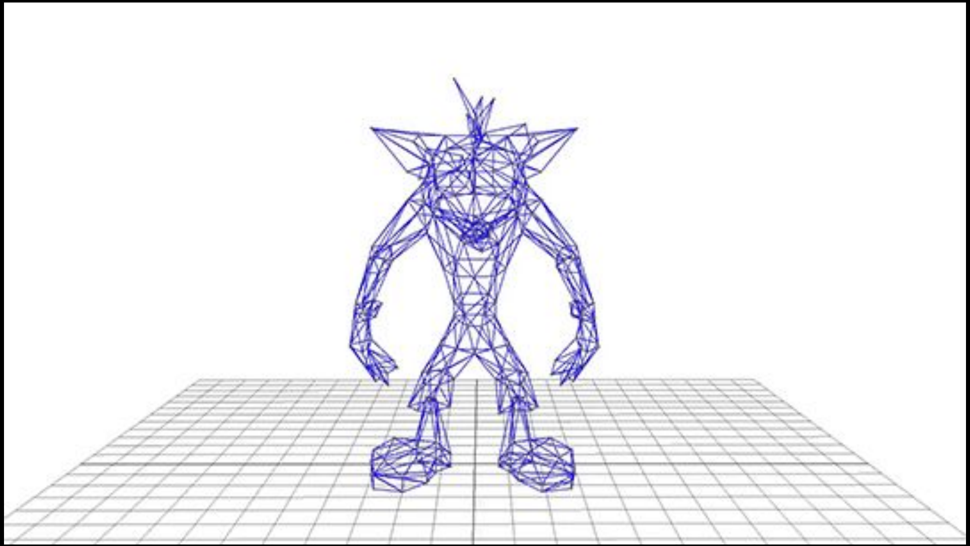
The keypoint

All of these operations can be done simultaneously

A chip to perform such task does not need to do everything a cpu can, just some multiplication/division for a huge number of objects in parallel

Modern Gpus

Thus “modern” gpus were born



In the meantime...

STVDIVM VRBIS

CPU History

How to get faster programs?

CPU History

How to get faster programs?

Just make the CPU faster.

CPU History

Does it actually work?

CPU History

Does it actually work?

Kinda

CPU History

Does it actually work?

Kinda

What are the limitations?

CPU History

Does it actually work?

Kinda

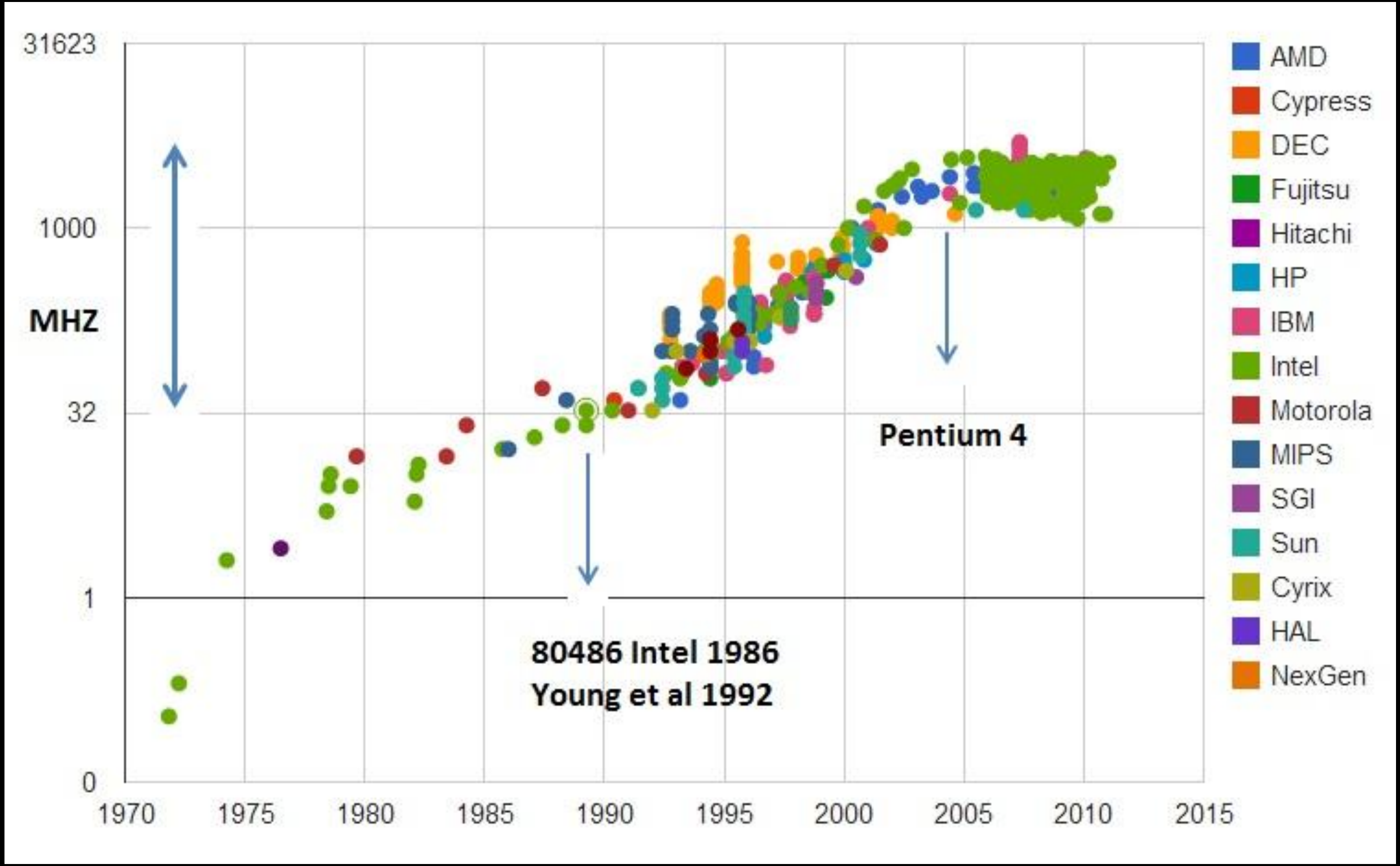
What are the limitations?

Physics

CPU History

More Frequency =

CPU History



31623
1000
32
1
0
MHZ
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
80486 Intel 1986
Young et al 1992
Pentium 4
AMD
Cypress
DEC
Fujitsu
Hitachi
HP
IBM
Intel
Motorola
MIPS
SGI
Sun
Cyrix
HAL
NexGen

CPU History

Maybe we need to find another way?

CPU History

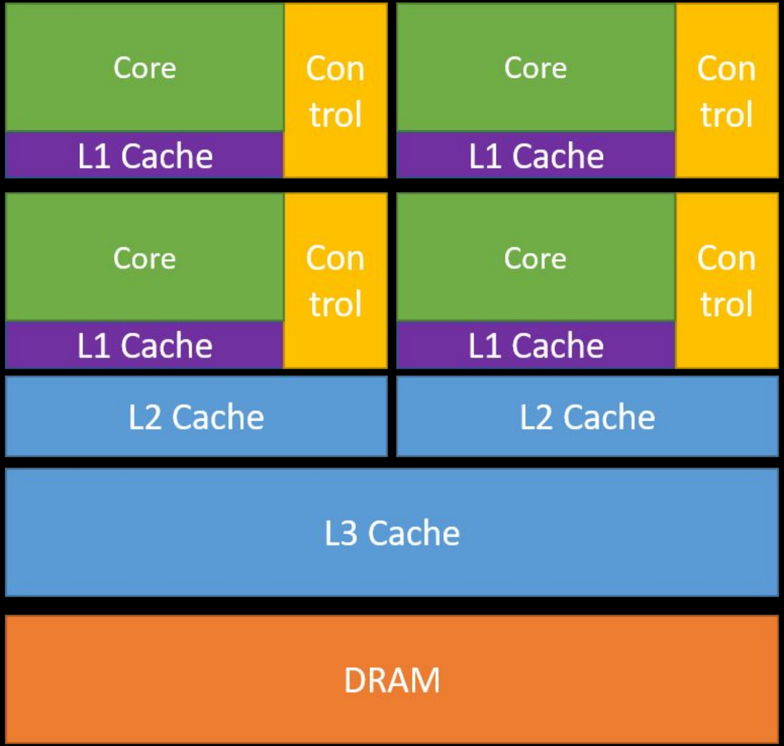
- Maybe we need to find another way?
  - We could go parallel

It’s Multicore Time

Average Multicore Experience:

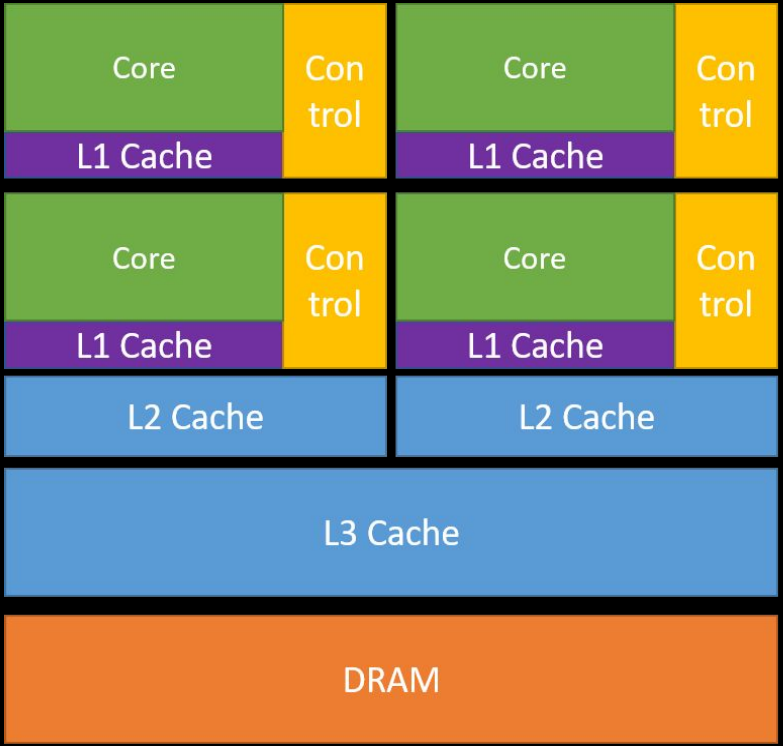
Multicore

Multicore CPU Architecture ->



Multicore

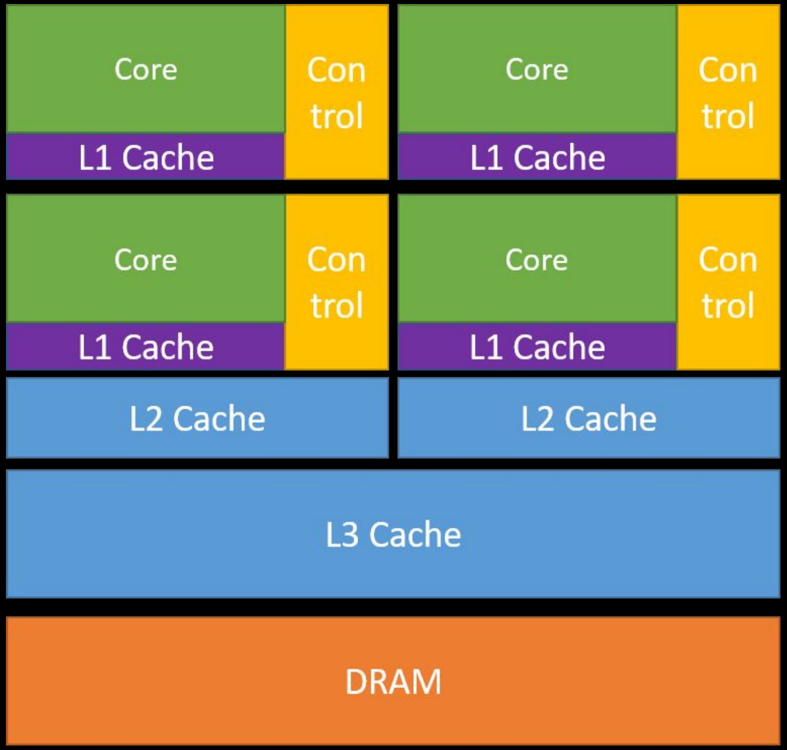
Not enough?



Multicore

Not enough?

We need more hardware

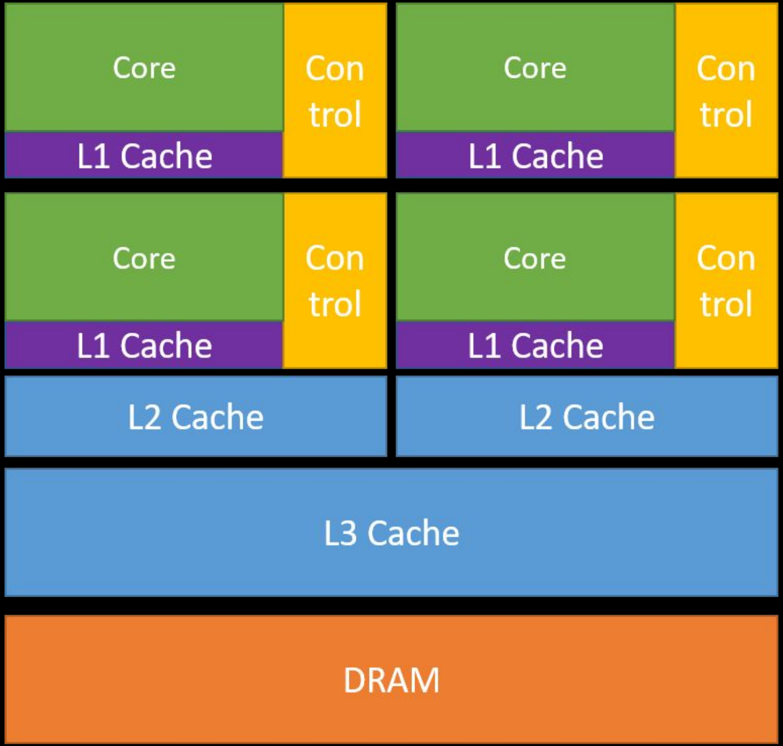


Multicore

Not enough?

We need more hardware

Maybe a dedicated one…



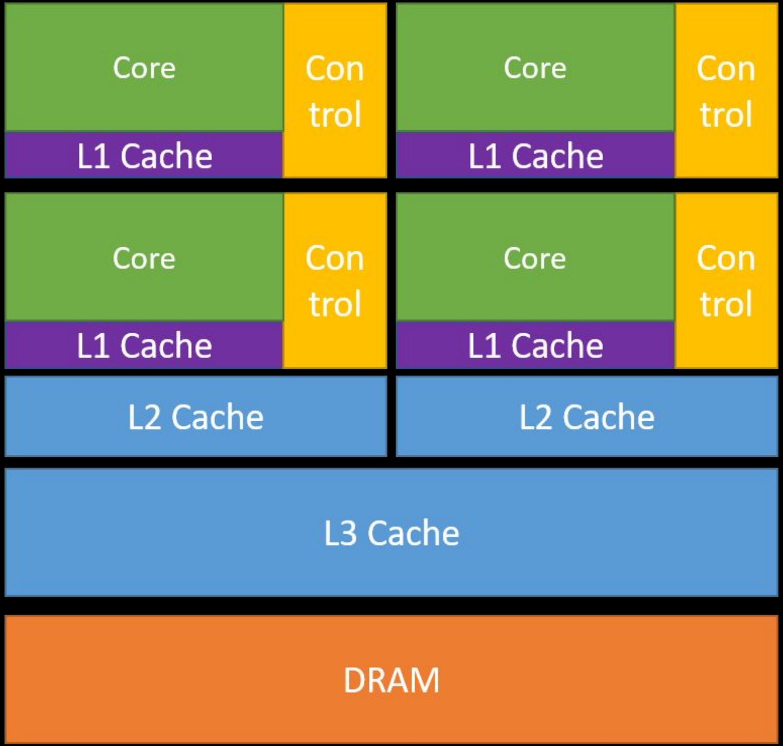
Multicore

Not enough?

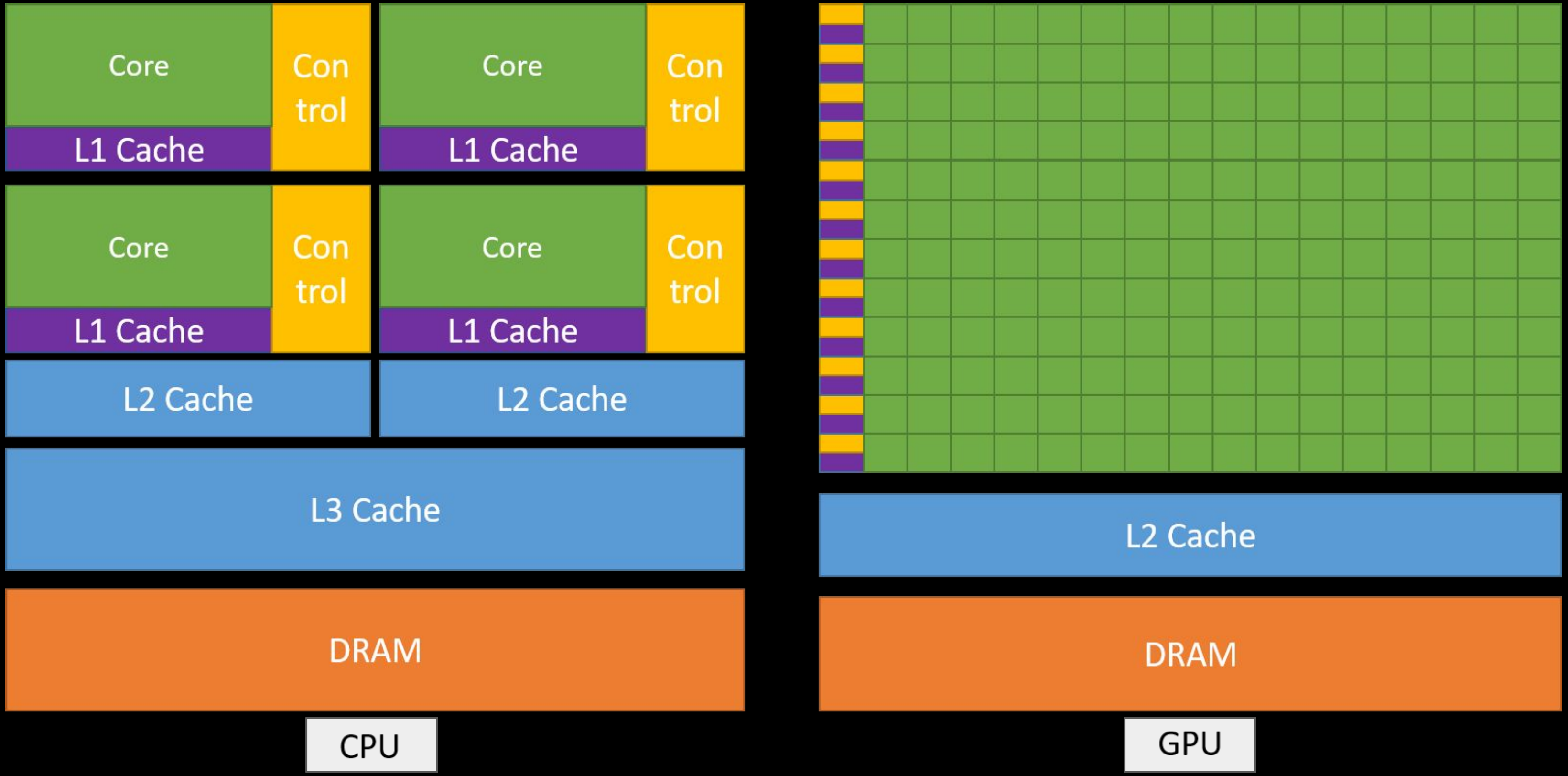
We need more hardware

Maybe a dedicated one...

WAIT!! WE HAVE GPUS!!

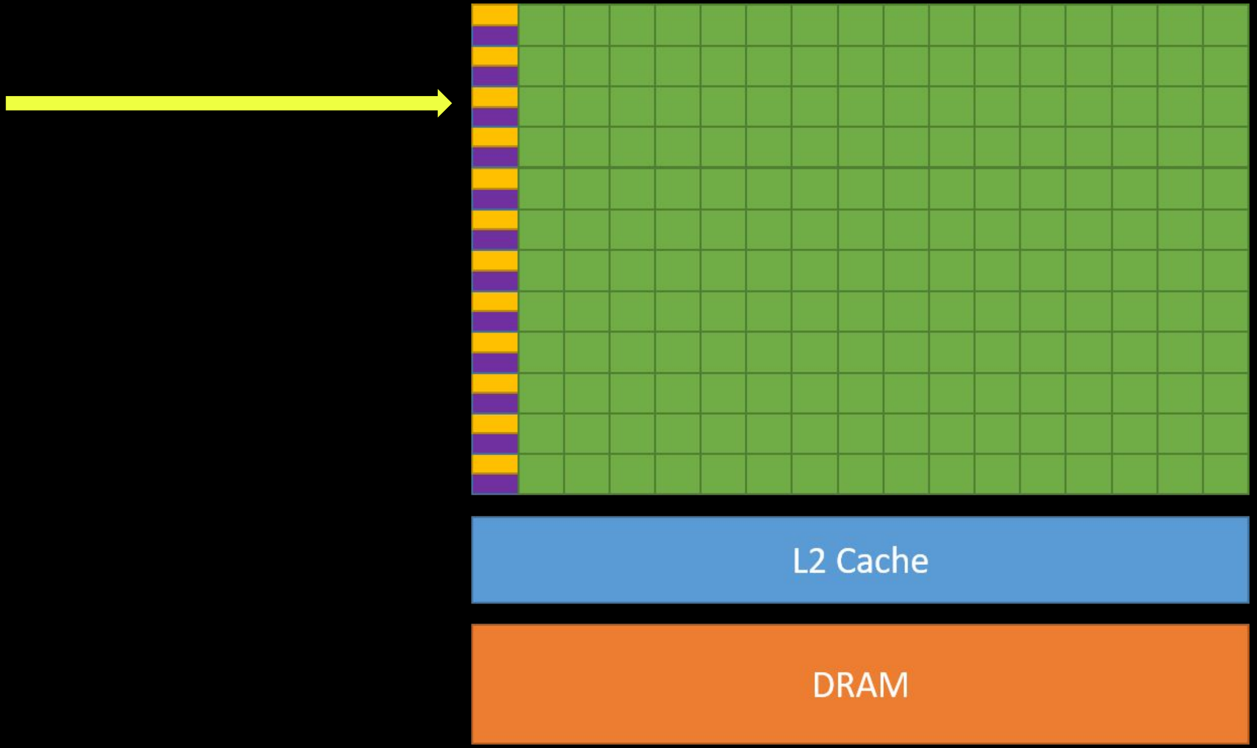


CPU vs GPU


Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
L2 Cache
L2 Cache
L3 Cache
DRAM
CPU
L2 Cache
DRAM
GPU

GPU

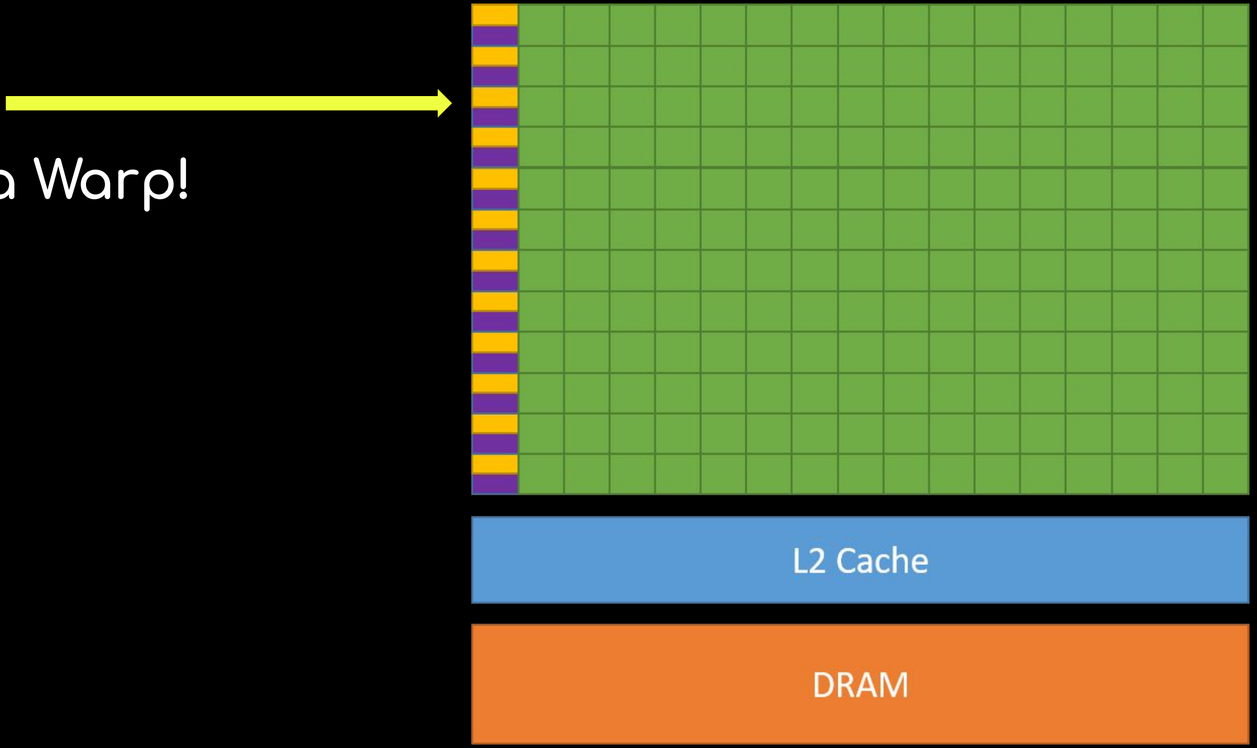
What is this?



GPU

What is this?

Obviously a Warp!

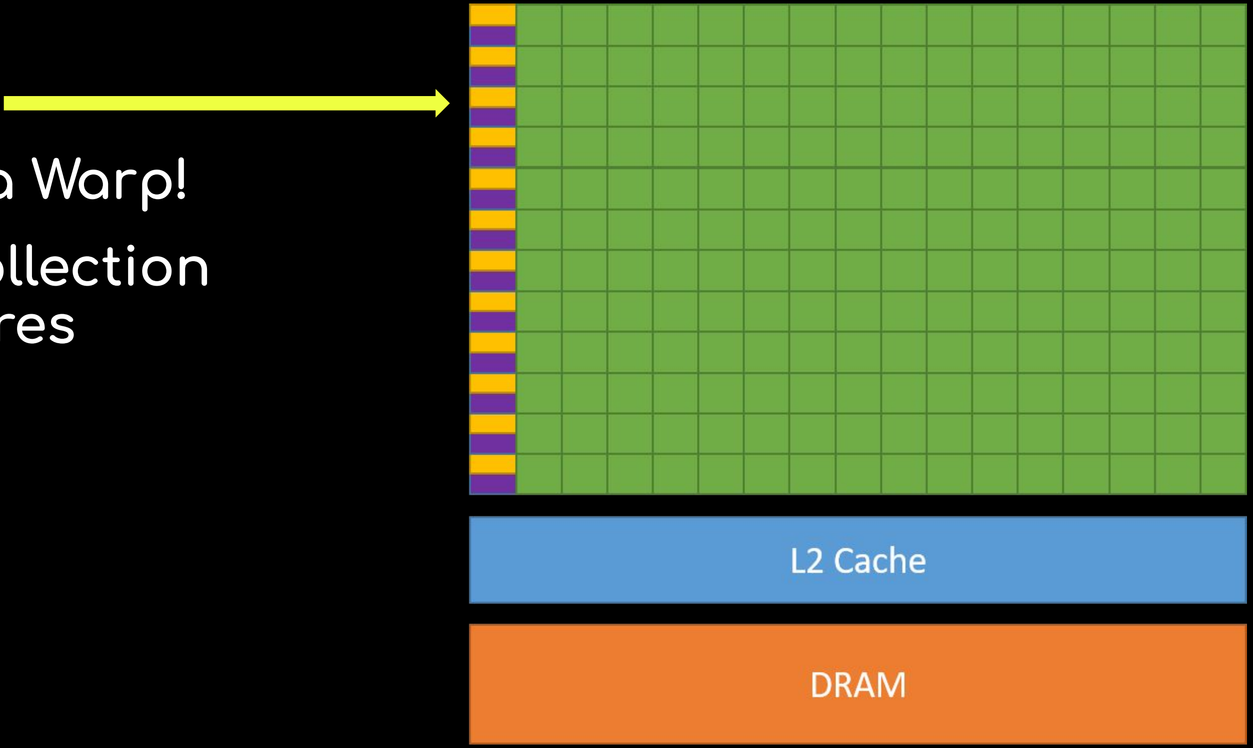


GPU

What is this?

Obviously a Warp!

A Warp is a collection of 32 cuda cores



GPU

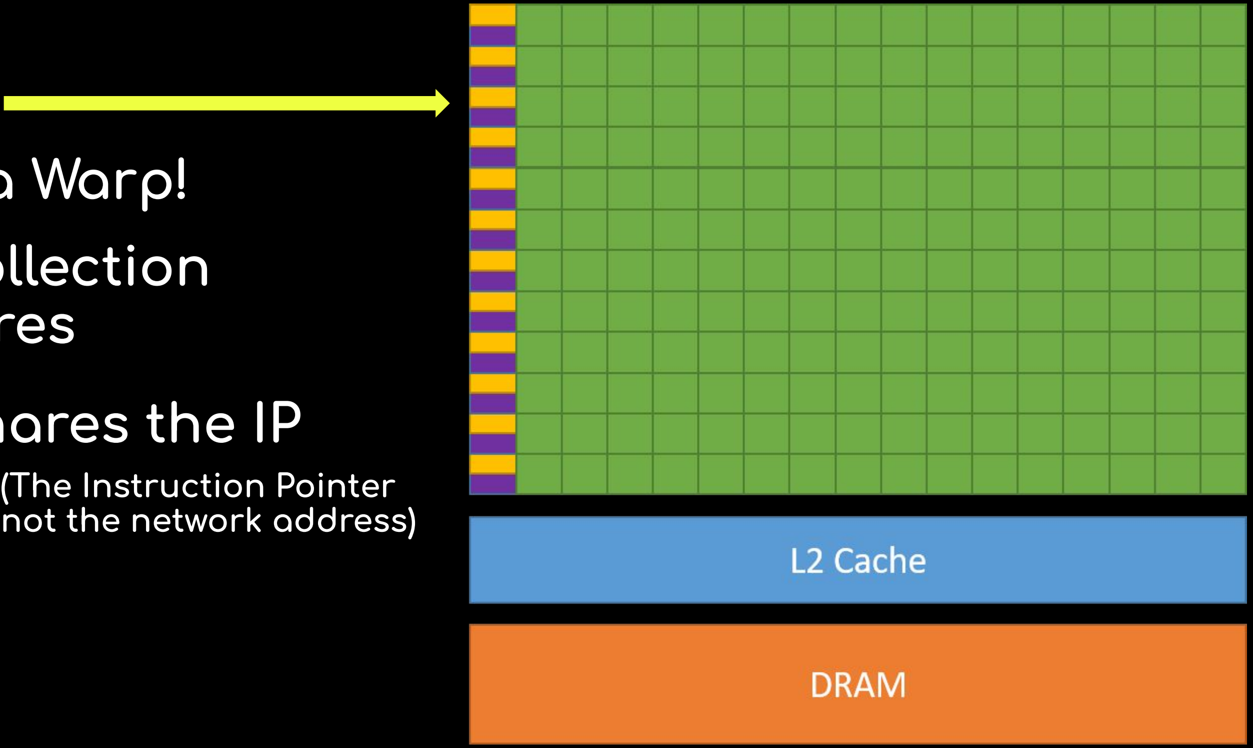
What is this?

Obviously a Warp!

A Warp is a collection of 32 cuda cores

Every Warp shares the IP

(The Instruction Pointer not the network address)



GPU

What is this?

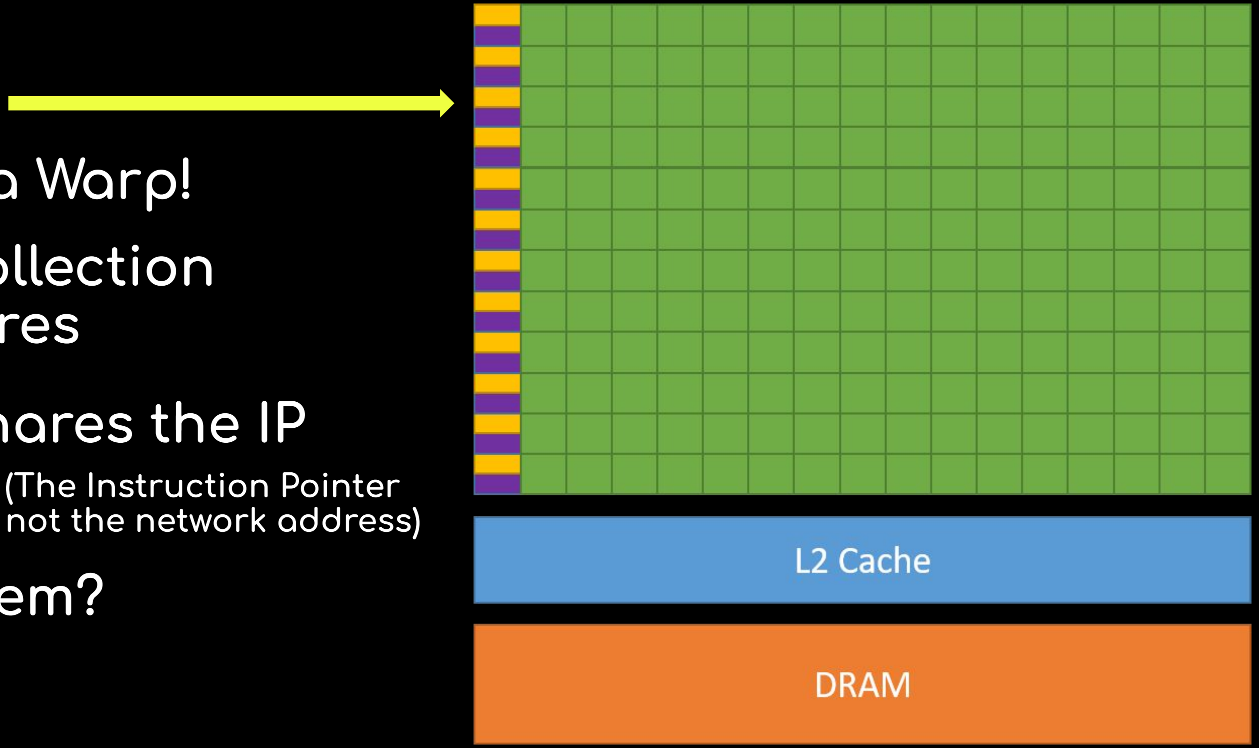
Obviously a Warp!

A Warp is a collection of 32 cuda cores

Every Warp shares the IP

(The Instruction Pointer not the network address)

Is this a problem?



GPU

What is this?

Obviously a Warp!

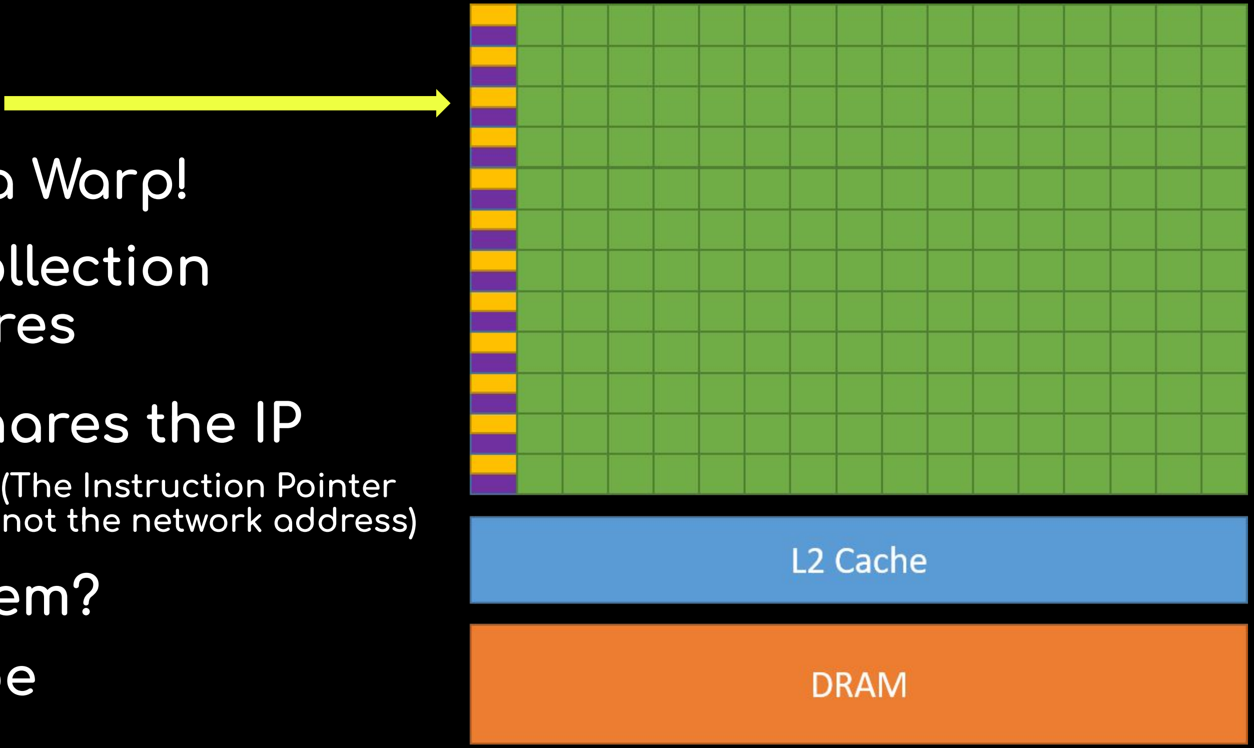
A Warp is a collection of 32 cuda cores

Every Warp shares the IP

(The Instruction Pointer not the network address)

Is this a problem?

Yes, it can be



The Enemy

STVDIVM VRBIS

The Enemy

IF

Divergence

What happens if there is a branch in the execution flow?

Divergence

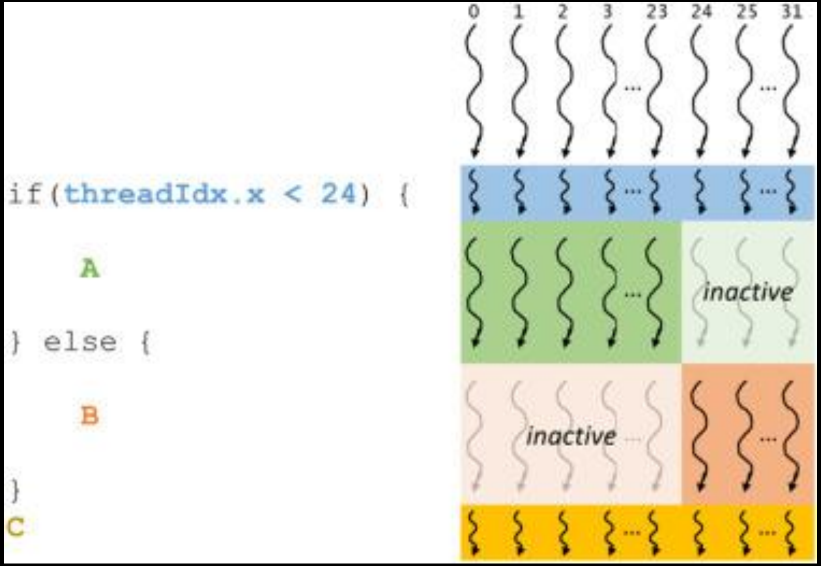
What happens if there is a branch in the execution flow?

- We have an event called divergence

Divergence

What happens if there is a branch in the execution flow?

- We have an event called divergence



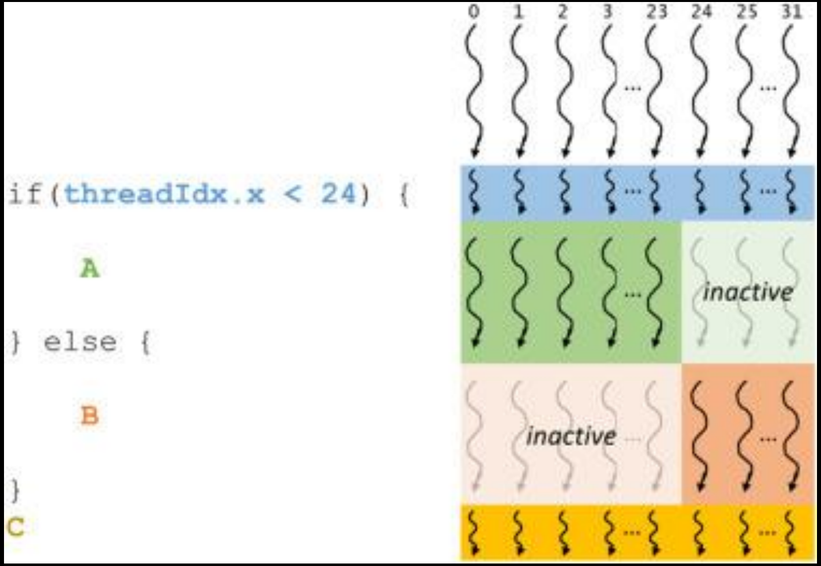
```
if(threadIdx.x < 24) {
    A
} else {
    B
}
C
```

Divergence

What happens if there is a branch in the execution flow?

We have an event called divergence

they are executed sequentially



```
if(threadIdx.x < 24) {
    A
} else {
    B
}
C
```

Some clarifications…

In gpus:

Some clarifications...

In gpus:

Threads = Cores (Cuda cores)

Some clarifications...

In gpus:

Threads = Cores (Cuda cores)

Thread block = Streaming Multiprocessor

Some clarifications...

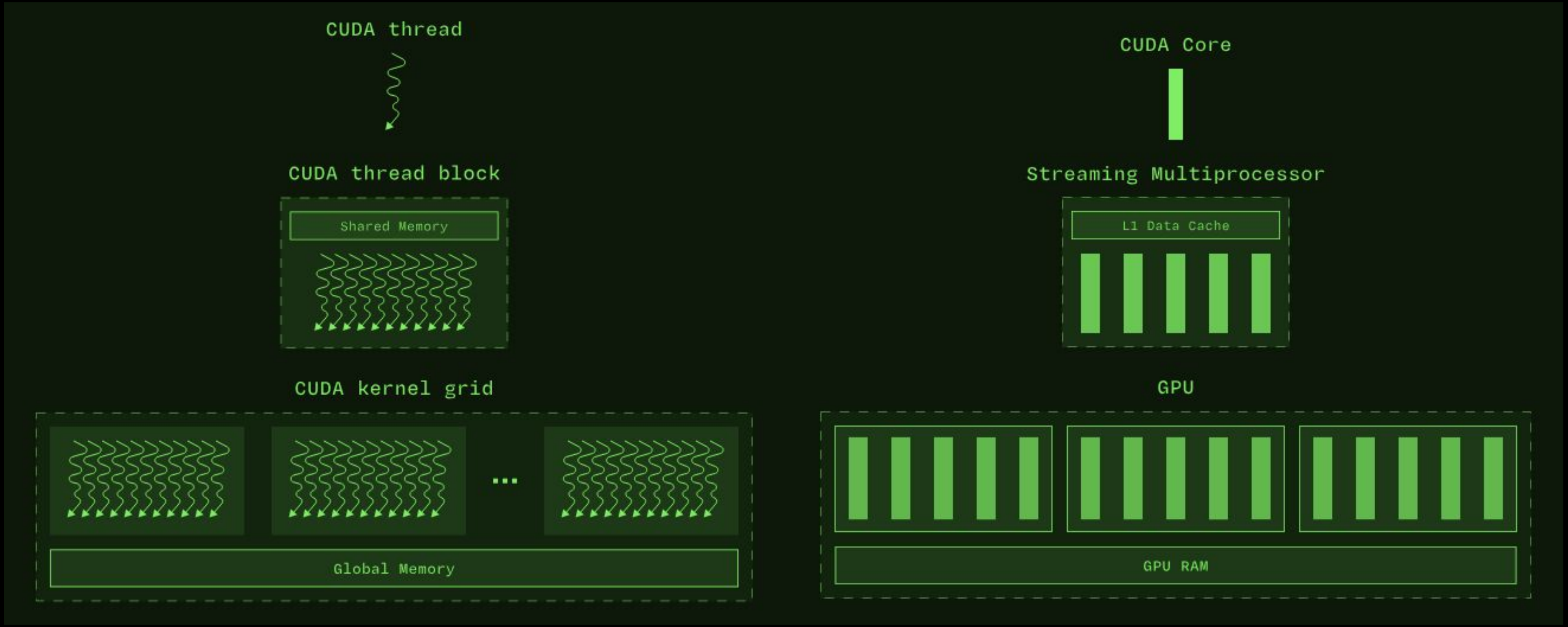
In gpus:

Threads = Cores (Cuda cores)

Thread block = Streaming Multiprocessor

Kernel Grid = The GPU

Little drawing on this



Scheduling

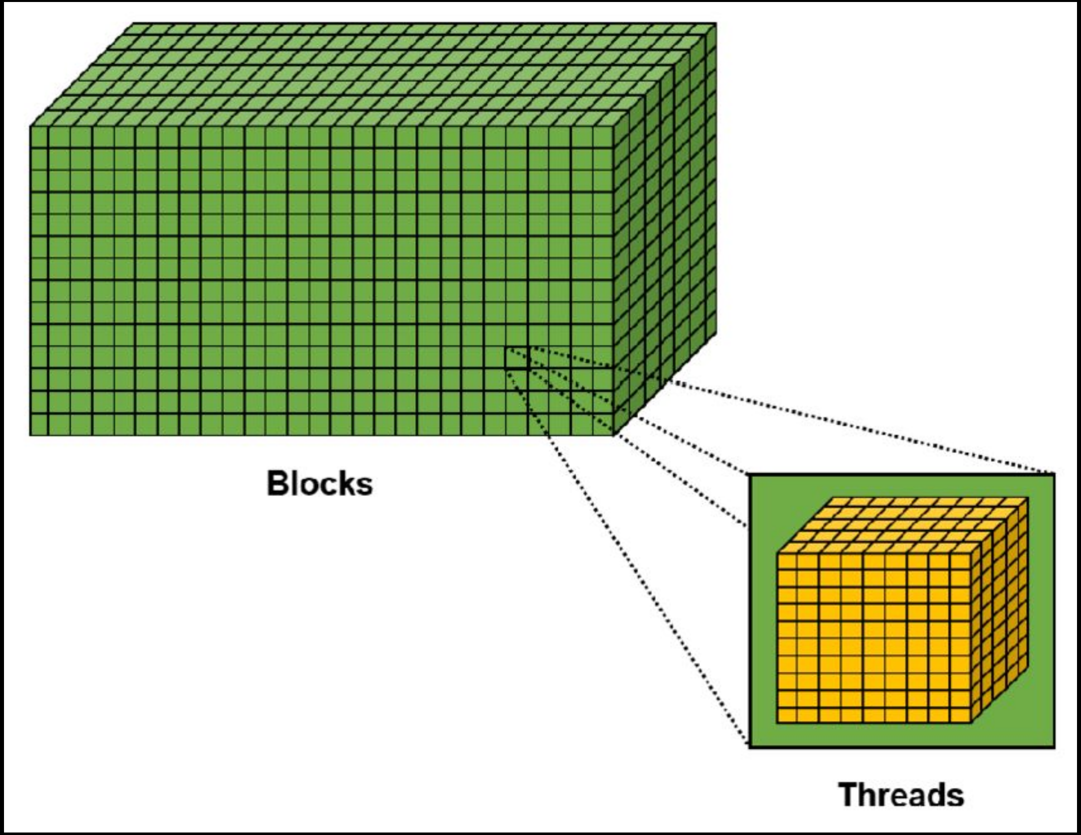
Now that we know this...
How we actually run threads?

Scheduling

Now that we know this...
How we actually run threads?

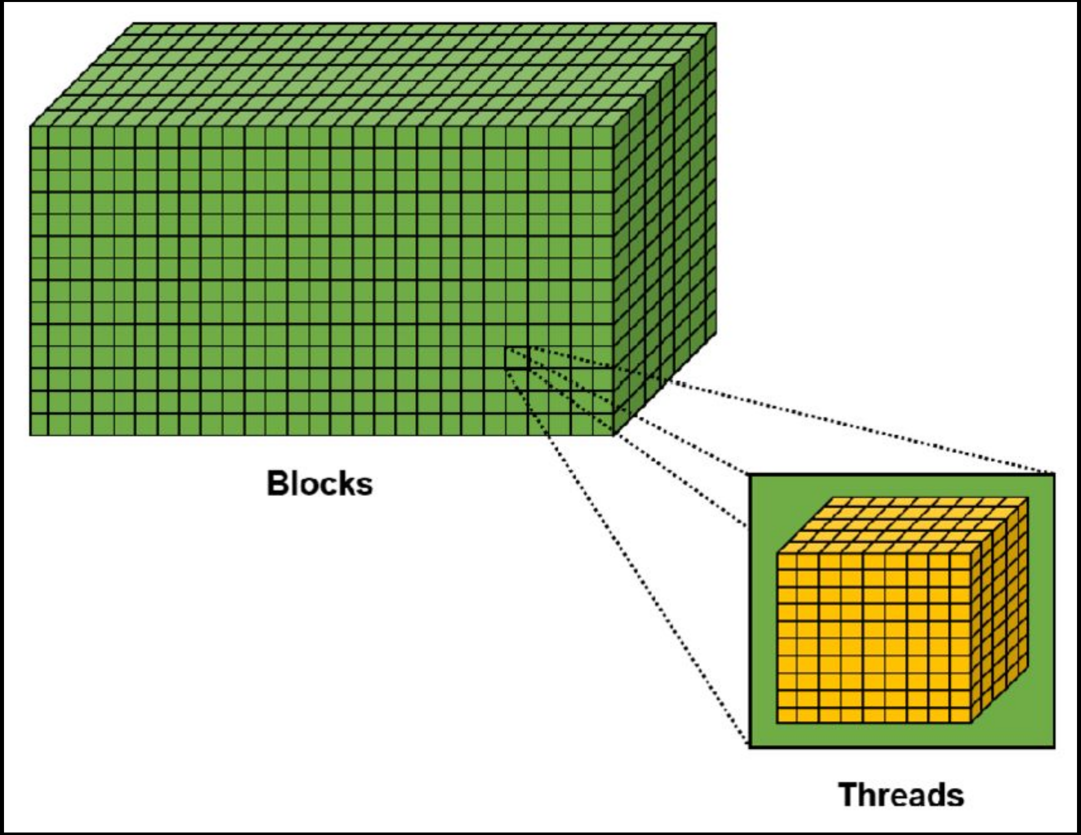
We can use...

The Grid



The Grid

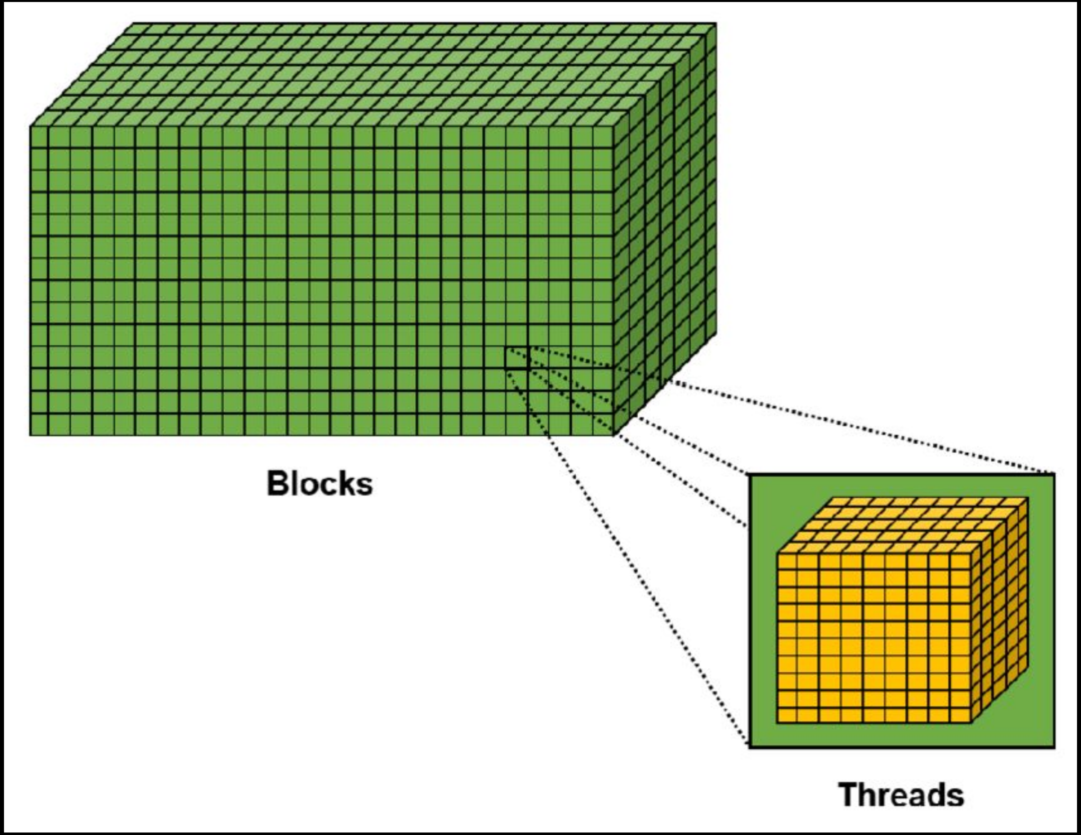
A grid is a 3D space composed by blocks



The Grid

A grid is a 3D space composed by blocks

A block is a 3D space made out of threads

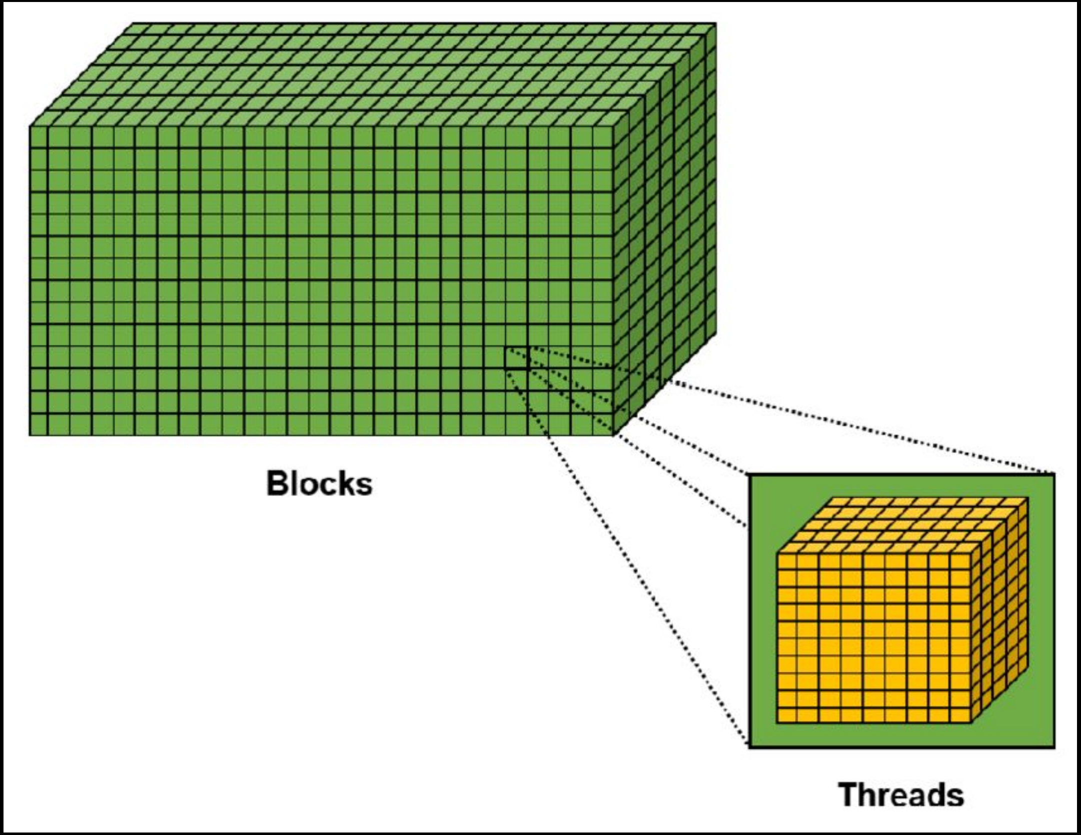


The Grid

A grid is a 3D space composed by blocks

A block is a 3D space made out of threads

From an hardware POV, a block is made out of warps

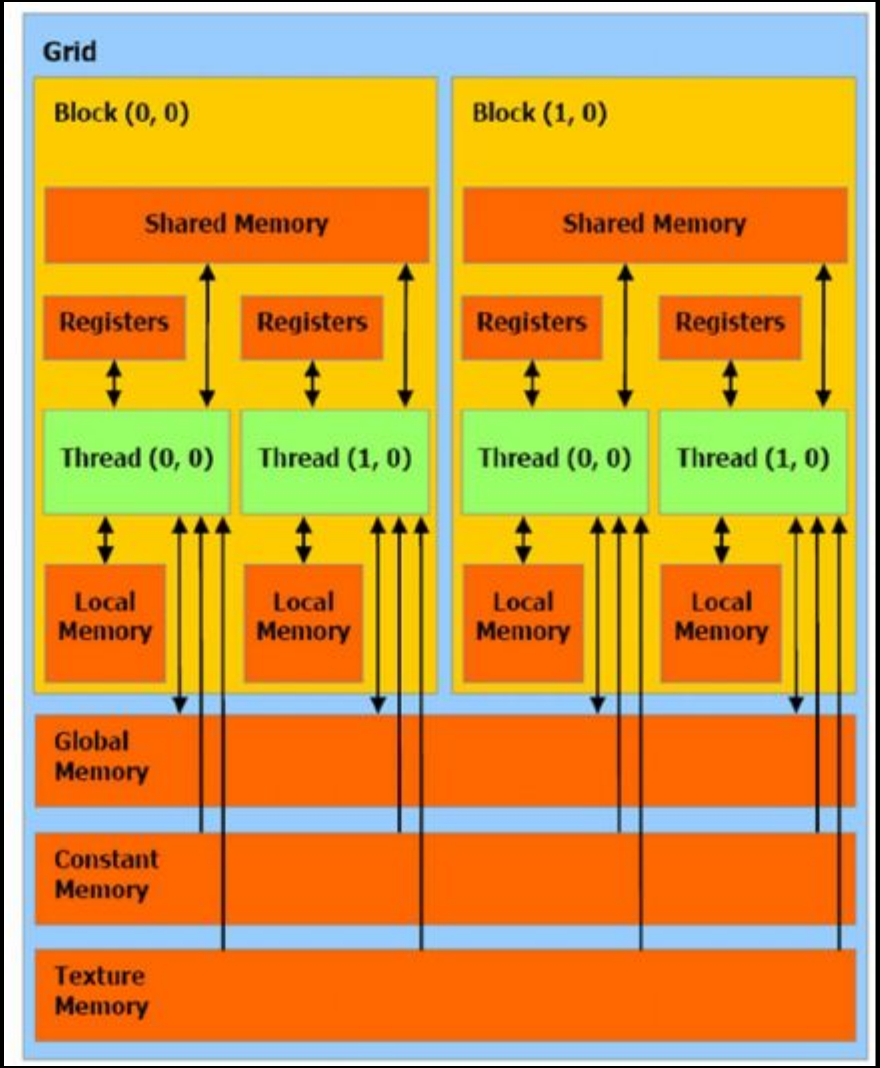


Memory

Ok… Now we can run programs, but where do we store the data?

Memory

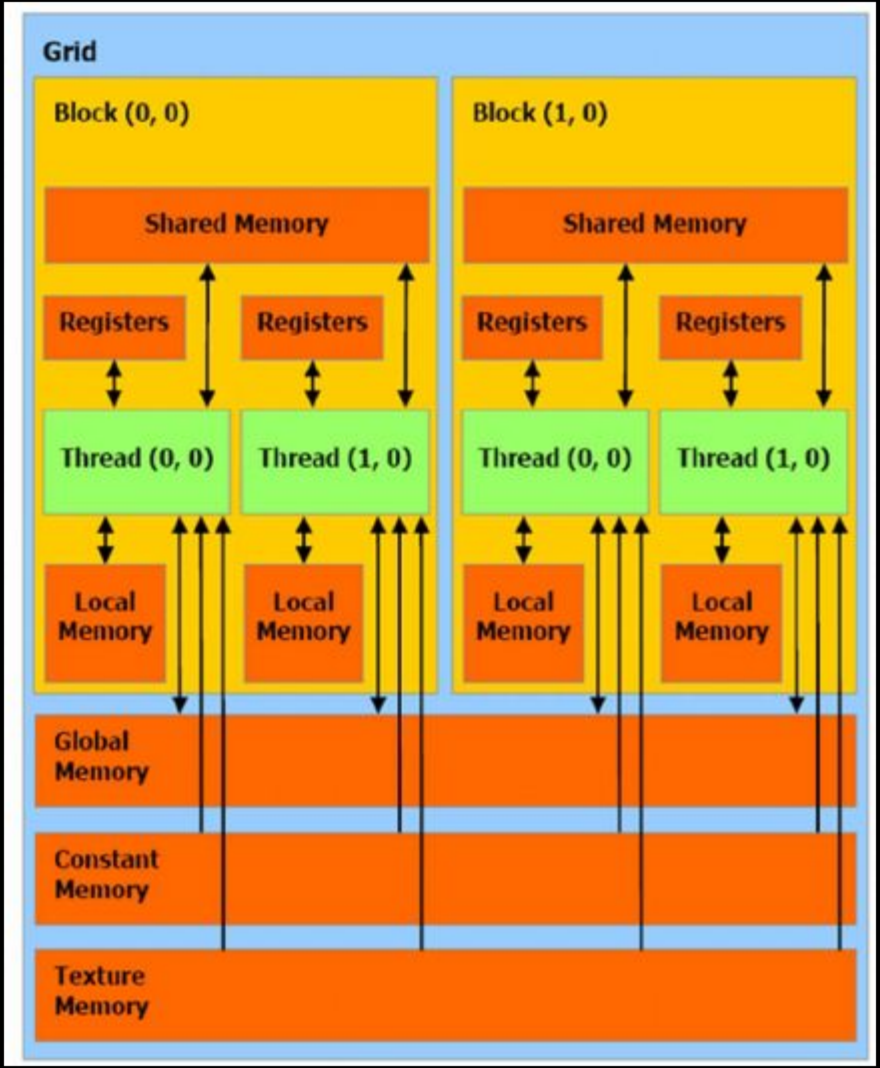
Ok... Now we can run programs, but where do we store the data?



Memory

We have a lot of memory types here:

- Texture memory
- Constant memory
- Global memory
- Shared memory
- Local memory
- Cache
- Registers



Memory

- Texture

It’s a global memory, and it’s used to store textures (it’s also optimized for them)

Memory

- Texture

It’s a global memory, and it’s used to store textures (it’s also optimized for them)

- Constant

It’s a read-only global memory

Memory

- Texture

It’s a global memory, and it’s used to store textures (it’s also optimized for them)

- Constant

It’s a read-only global memory

- Global

It’s a global memory similar to our standard heap

Memory

- Texture

It’s a global memory, and it’s used to store textures (it’s also optimized for them)

- Constant

It’s a read-only global memory

- Global

It’s a global memory similar to our standard heap

- Shared

It’s a memory shared between threads inside a block

Memory

- Texture

It's a global memory, and it's used to store textures (it's also optimized for them)

- Constant

It's a read-only global memory

- Global

It's a global memory similar to our standard heap

- Shared

It's a memory shared between threads inside a block

- Local

It's a thread local memory used like the traditional stack

Texture memory

What does it mean that this memory is “optimized for textures”?

Texture memory

What does it mean that this memory is "optimized for textures"?

First, some debunking, texture memory as "dedicated memory for textures" does not exist

Texture memory

What does it mean that this memory is "optimized for textures"?

First, some debunking, texture memory as "dedicated memory for textures" does not exist

We refer to texture memory as global memory for which there's a dedicated cache that uses a spatial locality policy, which in case of texture workloads (not only), can really speed up reads.

Texture memory

What does it mean that this memory is “optimized for textures”?

First, some debunking, texture memory as “dedicated memory for textures” does not exist

We refer to texture memory as global memory for which there’s a dedicated cache that uses a spatial locality policy, which in case of texture workloads (not only), can really speed up reads.

Such cache is READ ONLY, so not every application will benefit, also has some hardware enhancements to deal with on-fly decompression etc…

Constant memory

Constant memory follows the same fate, in a sense that it does not really exist and is part of global memory…

Constant memory

Constant memory follows the same fate, in a sense that it does not really exist and is part of global memory…

but

Constant memory

Constant memory follows the same fate, in a sense that it does not really exist and is part of global memory…

but

It is cached in special 64K read only block

Constant memory

Constant memory follows the same fate, in a sense that it does not really exist and is part of global memory...

but

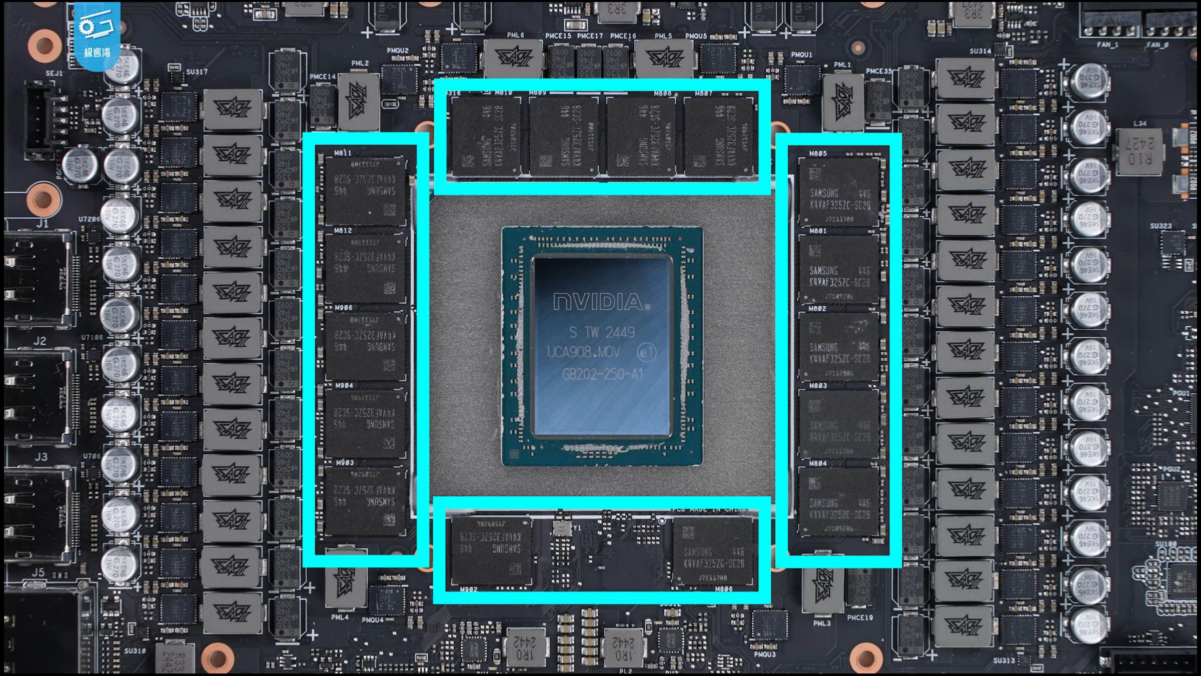
It is cached in special 64K read only block

It supports broadcasting of a single value within all the elements of a warp, providing near-register-speed access when all threads access the same element at the same time

STVDIVM VRBIS

Global memory

VRAM.



STVDIVM VRBIS

Shared memory

Blazingly fast on-chip memory.

Shared memory

Blazingly fast on-chip memory.

According to NVIDIA, memory latency can get down to 100x smaller *compared to uncached Global memory

Shared memory

Blazingly fast on-chip memory.

According to NVIDIA, memory latency can get down to 100x smaller *compared to uncached Global memory

Threads within a thread block can access shared memory loaded in from other threads, giving the user the ability to create and manage caches

Local memory

Threads can have their own local memory, isolated from other threads

Local memory

Threads can have their own local memory, isolated from other threads

Such memory isn’t particularly fast…
(similar speeds to the global memory)

Memory

Are we done with memories?

Memory

Are we done with memories?

- Well…

Zero-Copy Memory

What is this?

Zero-Copy Memory

What is this?

This is a page-locked memory

It’s pinned in memory, so it cannot be swapped

Zero-Copy Memory

What is this?

- This is a page-locked memory

It’s pinned in memory, so it cannot be swapped

The GPU access this memory directly into the CPU’s RAM

Memory

ARE WE DONE NOW??

Memory

ARE WE DONE NOW??

Unified Memory

Also known as Managed Memory

Unified Memory

Also known as Managed Memory

This maps memory in both CPU and GPU memory

Unified Memory

Also known as Managed Memory

This maps memory in both CPU and GPU memory

On page fault copies automatically the memory

Unified Memory

Also known as Managed Memory

This maps memory in both CPU and GPU memory

On page fault copies automatically the memory

- Cons: initial fault latency

Memories

Now We're done FR

Memories

Now We're done FR

Atomic

What if we want to do something simple like

var += result

with var as something shared

Atomic

What if we want to do something simple like

var += result

var ->

Atomic

What if we want to do something simple like

var += result

We can go Atomic!

Atomic

What if we want to do something simple like

var += result

We can go Atomic!

We have the standard atomic operations

Atomic

What if we want to do something simple like

var += result

We can go Atomic!

We have the standard atomic operations

- Bitwise
- Arithmetical
- Compares
- And more...

The End

Now Practice!

Me ssh-ing into toms docker and performing sudo rm -rf /: