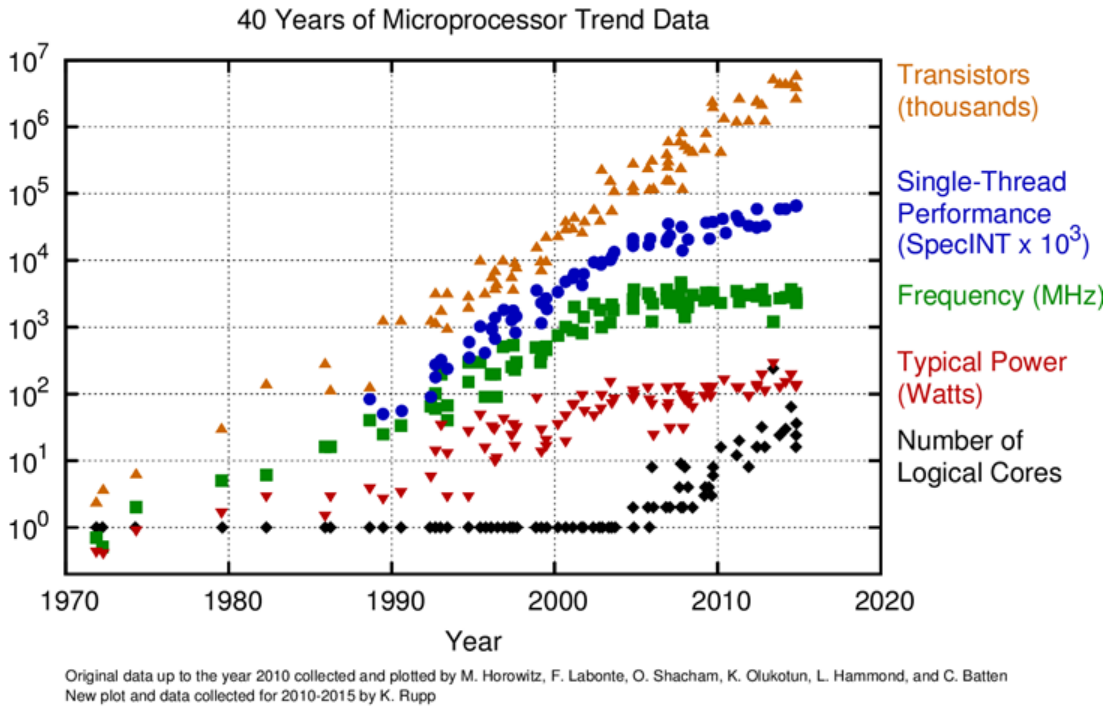
Introduction to GPU computing (1)

Computing Methods for Experimental Physics and Data Analysis
Lecture 3A

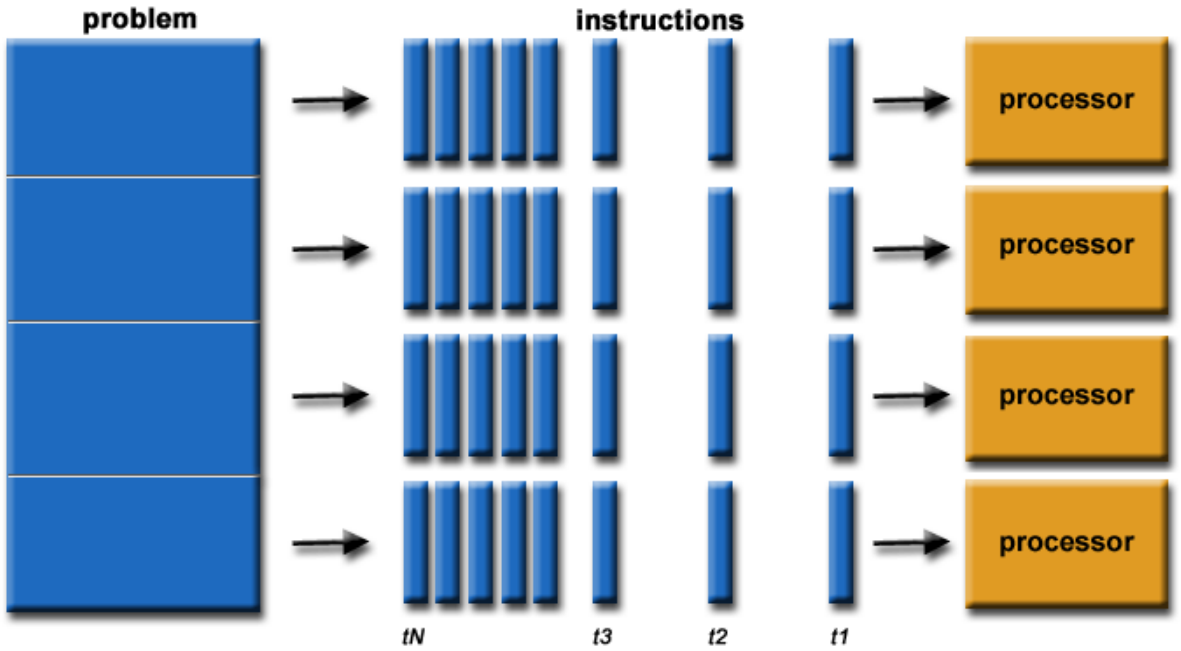
gianluca.lamanna@unipi.it

Moore's Law

- Moore's law: "The performance of microprocessors and the number of their transistors will double every 18 months"
- The increasing of performance is related to the clock
- Faster clock means higher dissipation → power wall



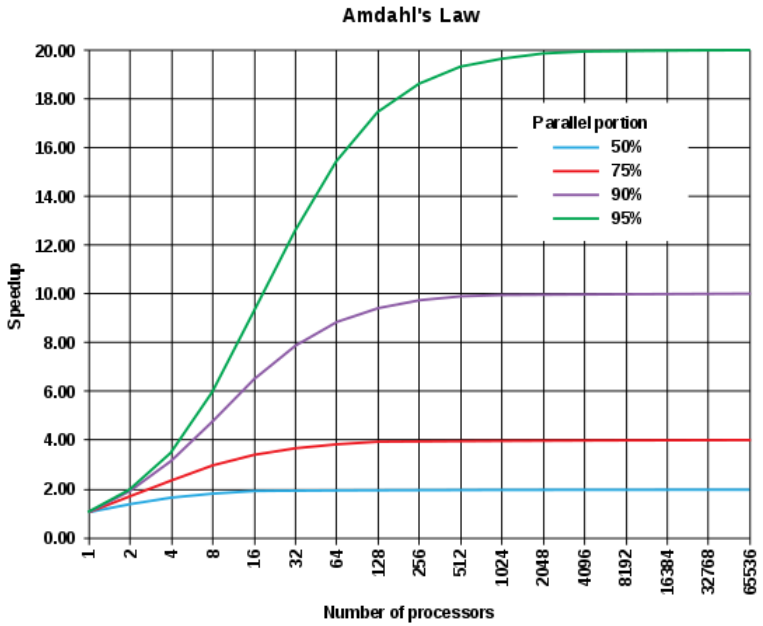
Parallel programming



- Parallel computing is no longer something for SuperComputers
  - All the processors nowadays are multicores
- The use of parallel architectures is mainly due to the physical constraints to frequency scaling

Limits of parallel programming

- Several problems can be split in smaller problems to be solved concurrently
- In any case the maximum speed-up is not linear , but it depends on the serial part of the code
  - ➡Amdahls's law
- The situation can improve if the amount of parallelizable part depends on the resources
  - ➡Gustafson's Law

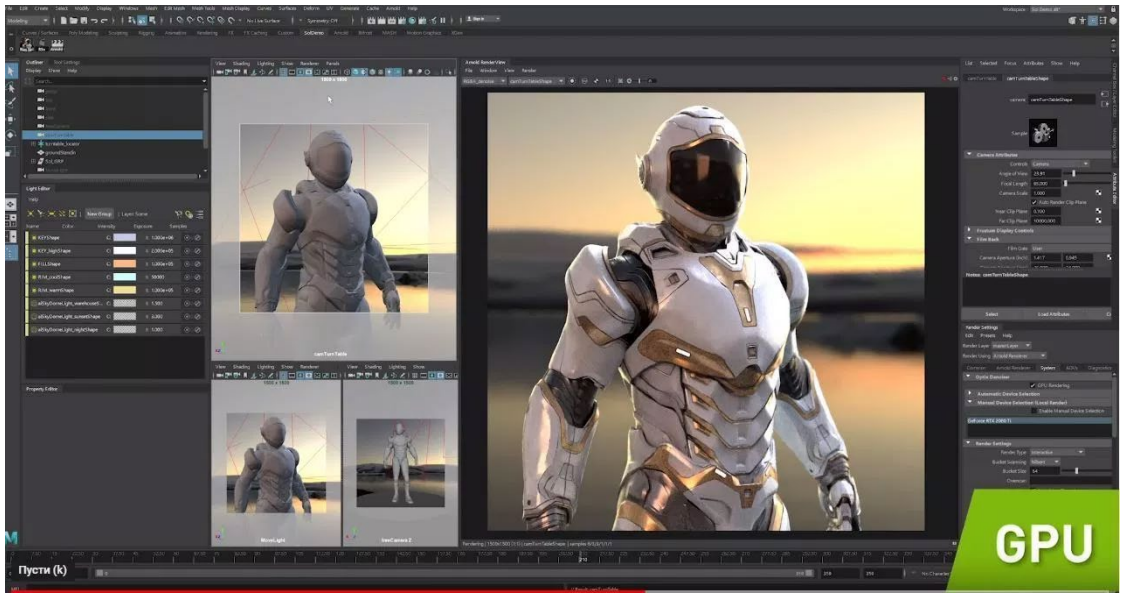


$$S_{latency} = \frac{1}{1 - p + \frac{p}{s}}$$

$$S_{latency} = 1 - p + sp$$

What are the GPUs?

- The GPUs are processors dedicated to parallel programming for graphical application.
- Rendering, Image transformation, ray tracing, etc. are typical application where parallelization can helps a lot.



Standard GPU pipeline

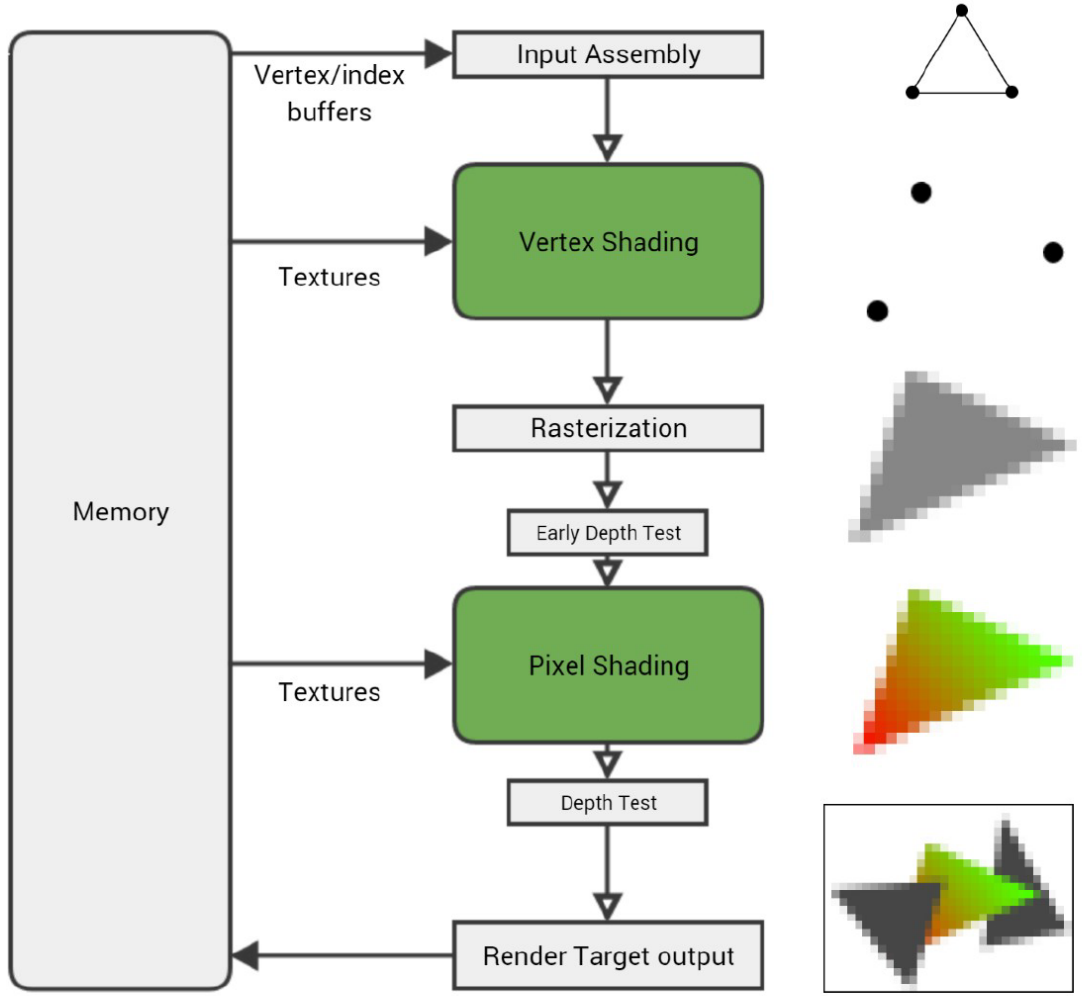
Vertex/index buffers:
Description of image with vertices and their connection to triangles

Vertex shading
For every vertex: calculate position on screen based on original position and camera view point

Rasterization
Get per-pixel color values

Pixel shading
For every pixel: get color based on texture properties (material, light, ...)

Rendering
Write output to render target



http://fragmentbuffer.com/gpu-performance-for-game-artists/

Standard GPU requirements

- Graphics pipeline: huge amount of arithmetic on independent data:
  - Transforming positions
  - Generating pixel colors
  - Applying material properties and light situation to every pixel
- Hardware needs
  - Access memory simultaneously and contiguously
  - Bandwidth more important than latency
  - Floating point and fixed-function logic

What are the GPUs?

(1997)

(2019)

- The technical definition of a GPU is "a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that is capable of processing a minimum of 10 million polygons per second.“
- The possibility to use the GPU for generic computing (GPGPU) has been introduced by NVIDIA in 2007 (CUDA)
- In 2008 OpenCL: consortium of different firms to introduce a multi-platform language for manycores computing.

Why the GPUs?

- GPU is a way to cheat the Moore's law
- SIMD/SIMT parallel architecture
- The PC no longer get faster, just wider.
- Very high computing power for «vectorizable» problems
- Impressive derivative almost a factor of 2 in each generation
- Continuous development
- Easy to have a desktop PC with teraflops of computing power, with thousand of cores.
- Several applications in HPC, simulation, scientific computing…

A lot of cores…

	"Fermi"	"Fermi"	"Kepler"	"Kepler"	"Maxwell"	"Pascal"
Tesla GPU	**GF100**	**GF104**	**GK104**	**GK110**	**GM200**	**GP100**
Compute Capability	2.0	2.1	3.0	3.5	5.3	6.0
Streaming Multiprocessors (SMs)	16	16	8	15	24	56
FP32 CUDA Cores / SM	32	32	192	192	128	64
FP32 CUDA Cores	512	512	1536	2880	3072	3584
FP64 Units	-	-	512	960	96	1792
Threads / Warp	32	32	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16	32	32
32-bit Registers / Multiprocessor	32768	32768	65536	65536	65536	65536
Max Registers / Thread	63	63	63	255	255	255
Max Threads / Thread Block	1024	1024	1024	1024	1024	1024
Shared Memory Size Configurations	16 KB	16 KB	16 KB	16 KB	96 KB	64 KB
	48 KB	48 KB	32 KB	32 KB		
			48 KB	48 KB		
Hyper-Q	No	No	No	Yes	Yes	Yes
Dynamic Parallelism	No	No	No	Yes	Yes	Yes
Unified Memory	No	No	No	No	No	Yes
Pre-Emption	No	No	No	No	No	Yes

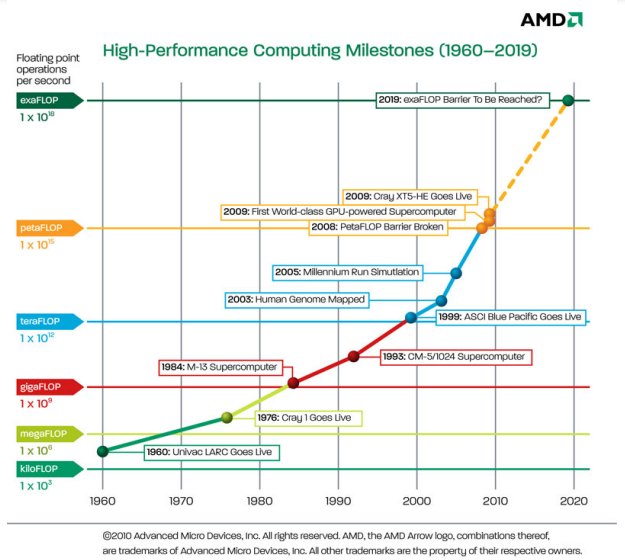
GPU Features	GTX 1080Ti	RTX 2080 Ti	Quadro P6000	Quadro RTX 6000
Architecture	Pascal	Turing	Pascal	Turing
GPCs	6	6	6	6
TPCs	28	34	30	36
SMs	28	68	30	72
CUDA Cores / SM	128	64	128	64
CUDA Cores / GPU	3584	4352	3840	4608
Tensor Cores / SM	NA	8	NA	8
Tensor Cores / GPU	NA	544	NA	576
RT Cores	NA	68	NA	72
GPU Base Clock MHz (Reference / Founders Edition)	1480 / 1480	1350 / 1350	1506	1455

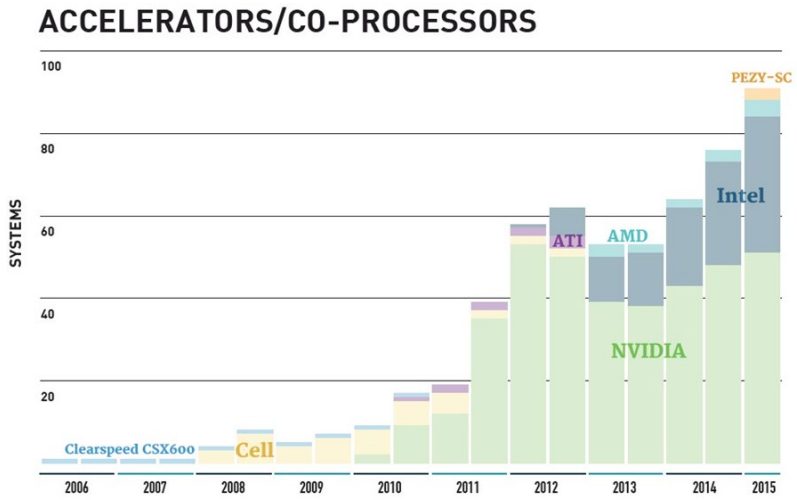
Metrics

- FLOPS (Floating Point operation per second)
  - ➡ It is a measurement of the computing power of a processor
  - ➡ Theoretically is defined as

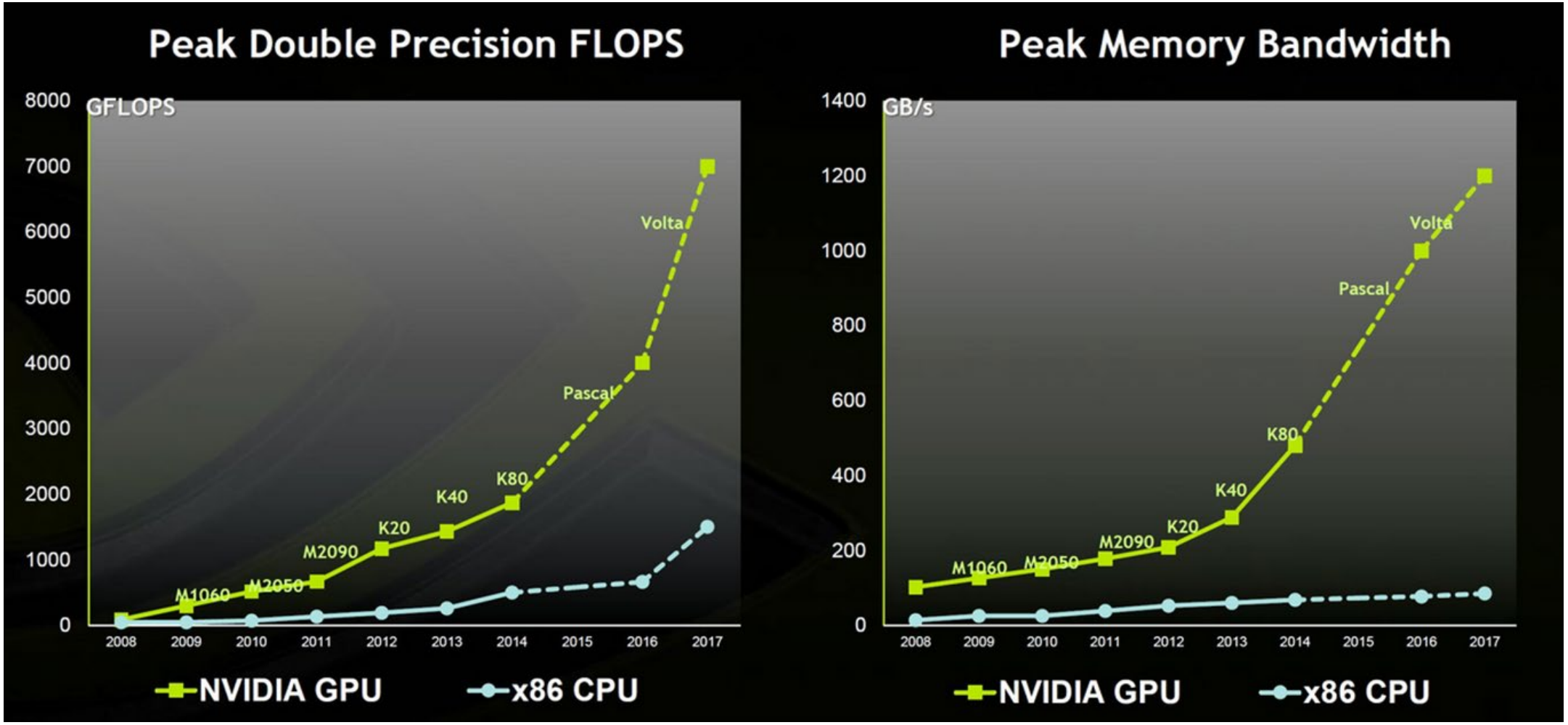
$$FLOPS = clock * cores * Operation/cycle$$

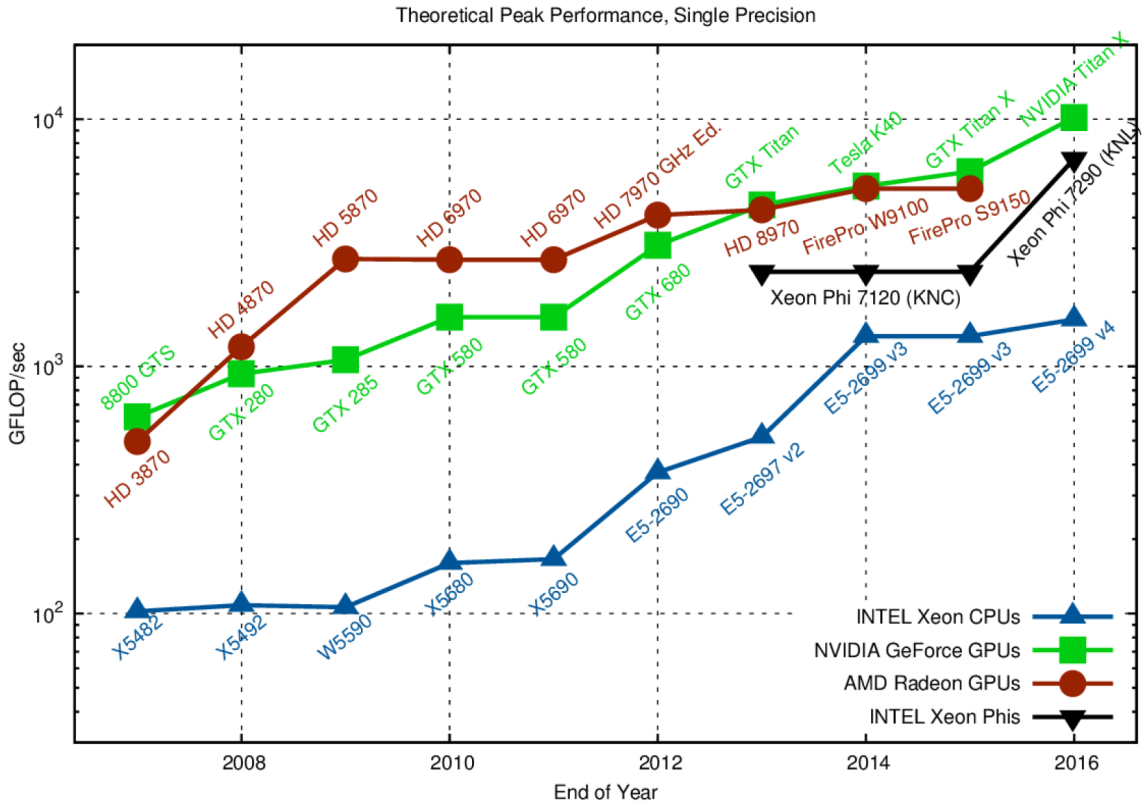
  - ➡ Actually this formula doesn't take into account several things
  - ➡ The real extimation is made experimentally by using standard packages (LINPACK, LAPACK)
  - ➡ The FLOPS is only a first indication of the computing power
  - ➡ Other metrics has been invented to avoid the limitations of the FLOPS
- SPECint and SPECf
  - ➡ They are suites of 12 benchmarks of diffent type (for integer and floating point)
  - ➡ The extimation is relative to a particular machine

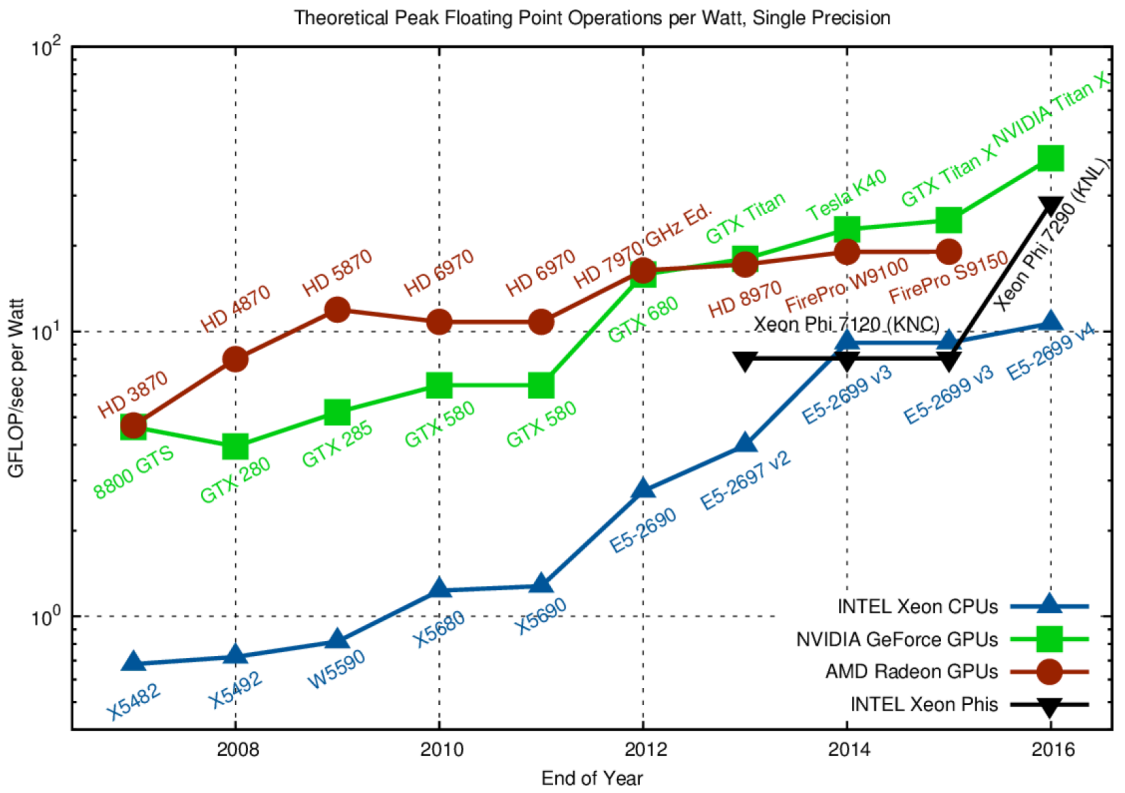




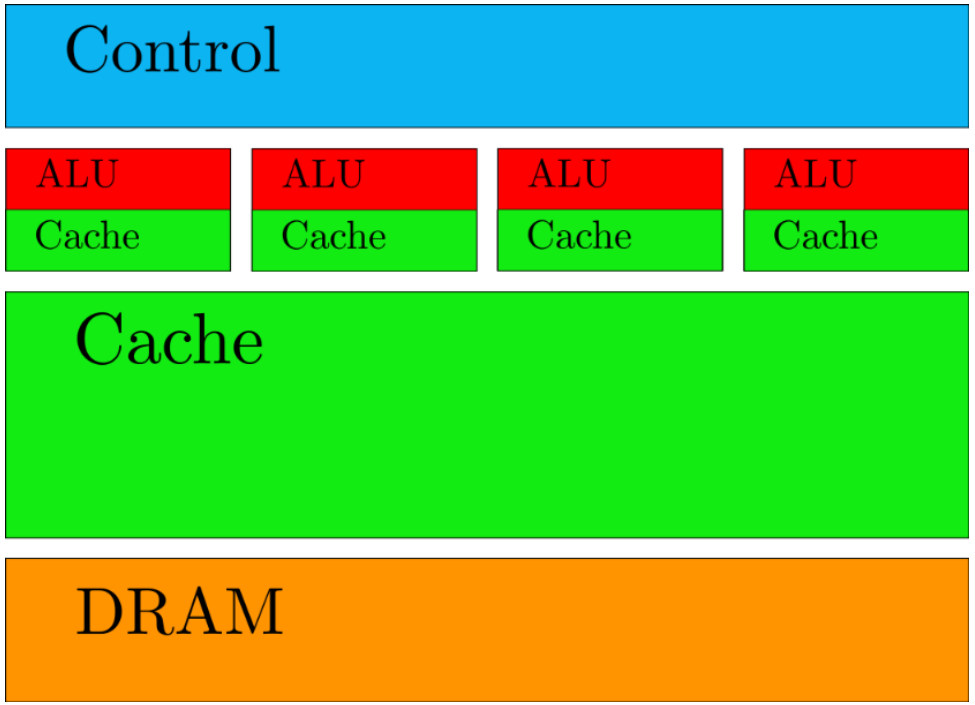
Computing power comparison




Theoretical Peak Performance, Single Precision
GFLOP/sec
End of Year
2008
2010
2012
2014
2016
8800 GTS
HD 4870
HD 5870
HD 6970
HD 6970
HD 7970 GHz Ed.
GTX Titan
Tesla K40
GTX Titan X
NVIDIA Titan X
HD 3870
GTX 280
GTX 285
GTX 580
GTX 580
GTX 680
HD 8970
FirePro W9100
FirePro S9150
Xeon Phi 7290 (KNL)
Xeon Phi 7120 (KNC)
X5482
X5492
W5590
X5680
X5690
E5-2690
E5-2697 v2
E5-2699 v3
E5-2699 v3
E5-2699 v4
INTEL Xeon CPUs
NVIDIA GeForce GPUs
AMD Radeon GPUs
INTEL Xeon Phis


Theoretical Peak Floating Point Operations per Watt, Single Precision
GFLOP/sec per Watt
End of Year
2008
2010
2012
2014
2016
HD 4870
HD 5870
HD 6970
HD 6970
HD 7970 GHz Ed.
GTX Titan
Tesla K40
GTX Titan X
NVIDIA Titan X
HD 3870
8800 GTS
GTX 280
GTX 285
GTX 580
GTX 580
GTX 680
HD 8970
FirePro W9100
FirePro S9150
Xeon Phi 7290 (KNL)
Xeon Phi 7120 (KNC)
X5482
X5492
W5590
X5680
X5690
E5-2690
E5-2697 v2
E5-2699 v3
E5-2699 v3
E5-2699 v4
INTEL Xeon CPUs
NVIDIA GeForce GPUs
AMD Radeon GPUs
INTEL Xeon Phis

CPU

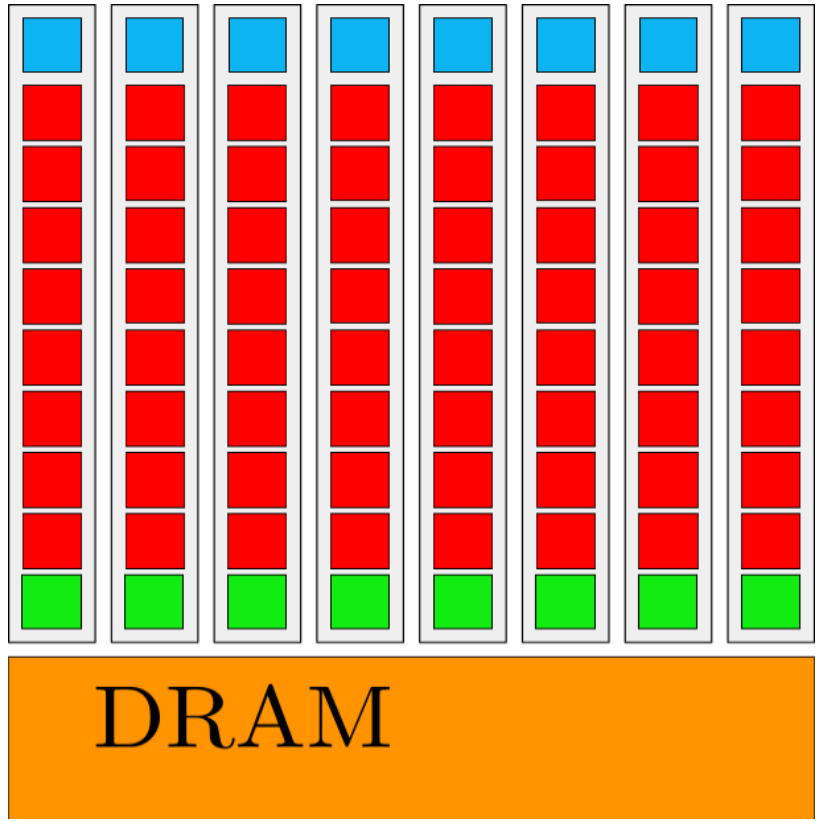


CPU: latency oriented design

- Multilevel and Large Caches
  - ➡Convert long latency memory access to short latency cache latency
- Branch prediction
  - ➡To reduce latency in branching
- Instruction level parallelism (ILP)
- Powerful ALU
  - ➡Reduced operation latency
- Memory management
- Large control part

GPU

- SIMT/SIMD (Single instruction Multiple Thread/Data) architecture
- SMX (Streaming Multi Processors) to execute kernels
- Thread level parallelism
  - ➡Massive threading to hide the latency
- Limited caching
  - ➡To boost memory throughput
- Limited control
- No branch prediction, but branch predication



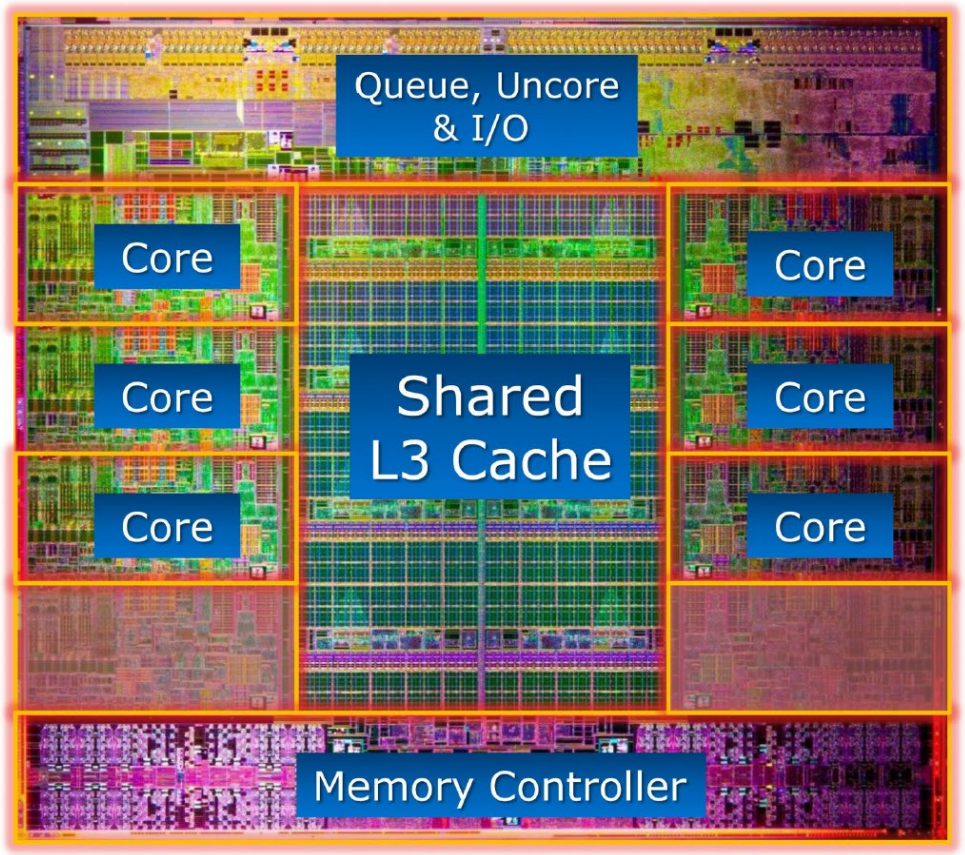
GPU: throughput oriented design

CPU vs GPU

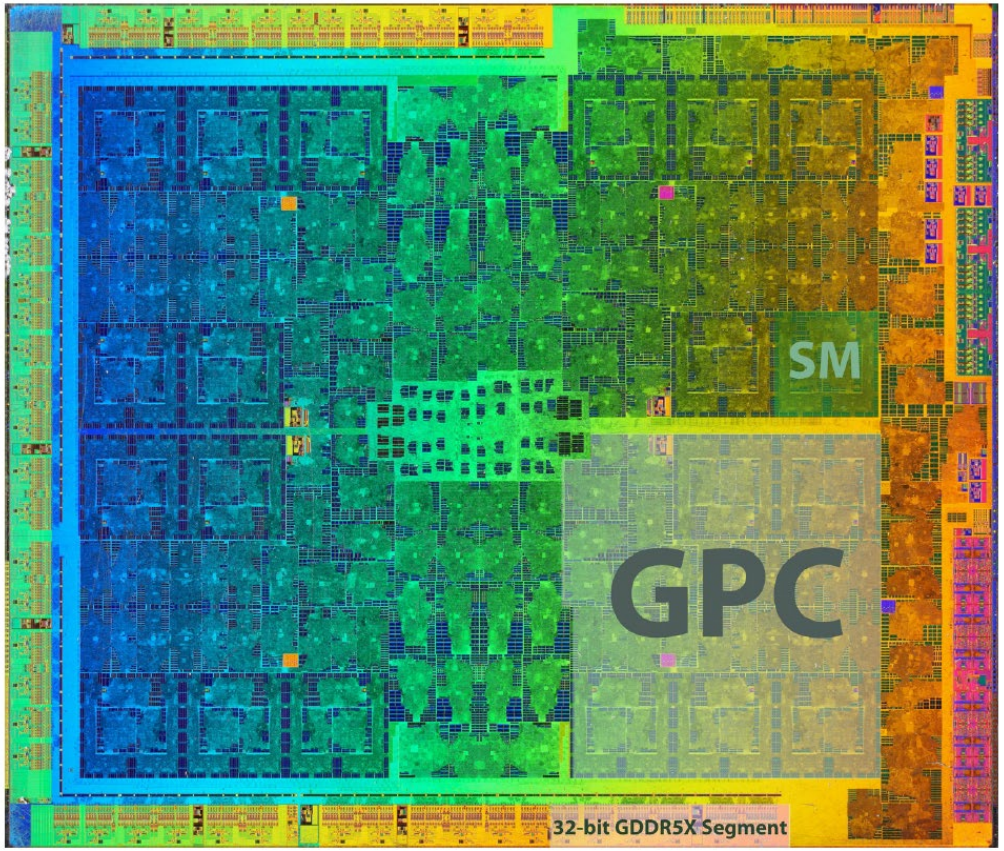
- + Large main memory
- + Fast clock rate
- + Large caches
- + Branch prediction
- + Powerful ALU
- Relatively low memory bandwidth
- Cache misses costly
- Low performance per watt

- + High bandwidth main memory
- + Latency tolerant (parallelism)
- + More compute resources
- + High performance per watt
- Limited memory capacity
- Low per-thread performance
- Extension card

CPU vs GPU



http://images.anandtech.com/reviews/cpu/intel/SNBE/Core_I7_LGA_2011_Die.jpg



http://wccftech.com/nvidia-gtx-1080-gp104-die-shot/

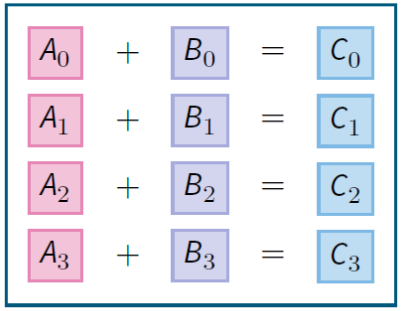
CPU vs GPU

	Intel Core E7-8890 v3	**GeForce GTX 1080**
Core count	18 cores / 36 threads	20 SMs / 2560 cores
Frequency	2.5 GHz	1.6 GHz
Peak Compute Performance	1.8 GFLOPs	8873 GFLOPs
Memory bandwidth	Max. 102 GB/s	320 GB/s
Memory capacity	Max. 1.54 TB	8 GB
Technology	22 nm	16 nm
Die size	662 mm^2	314 mm^2
Transistor count	5.6 billion	7.2 billion
Model	Minimize latency	Hide latency through parallelism

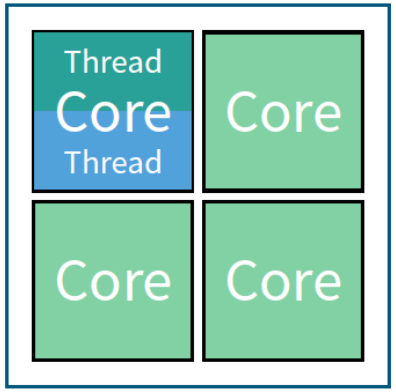
SIMT

- Standard CPU : Scalar processors
- SIMD CPU: vector processors
- Simultaneous threads in multicore processors
- SIMT (Single Instruction Multiple Threads)
  - ➡ CPU core ~ GPU multiprocessor (SMX)
  - ➡ Working unit: a set of threads (32, a warp)
  - ➡ Fast switching of threads

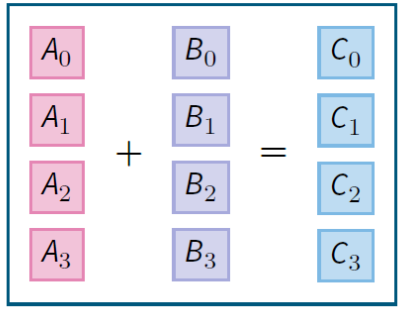
Scalar



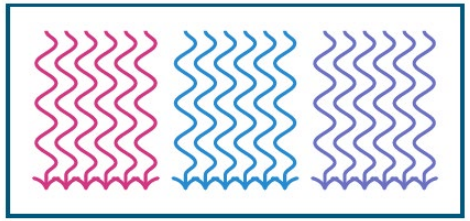
SMT



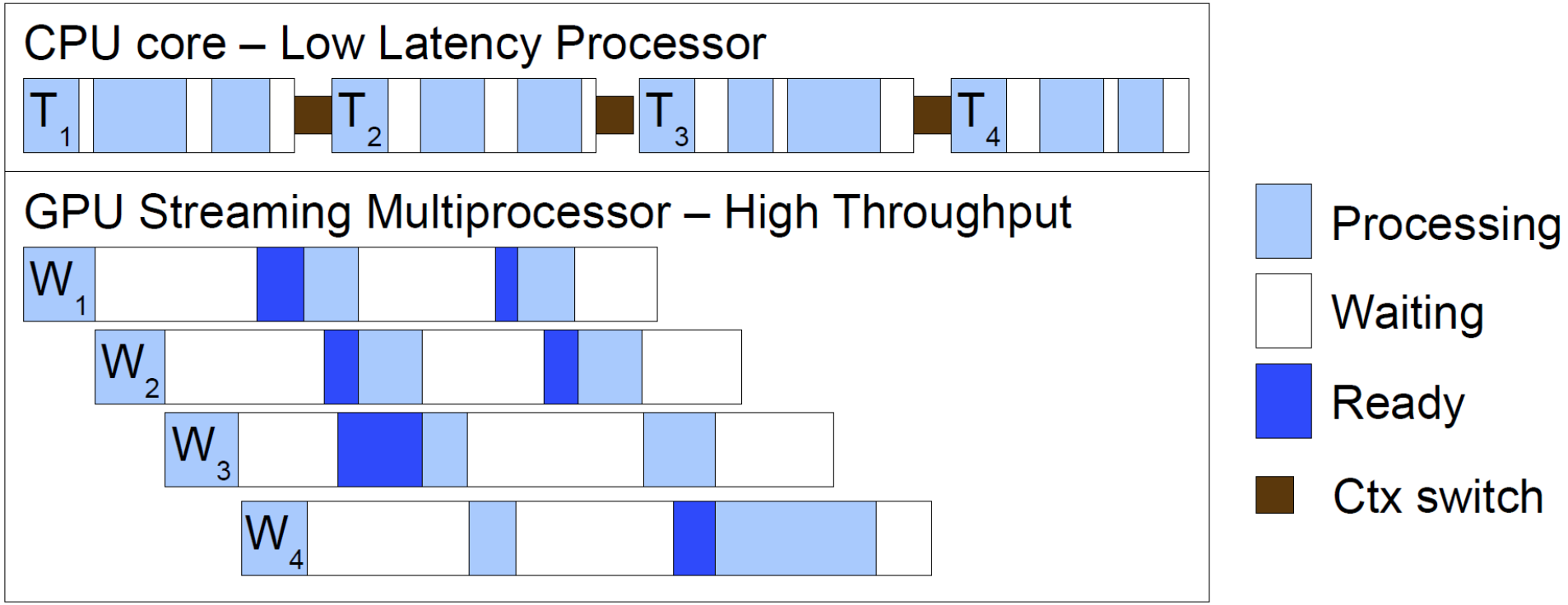
Vector



SIMT



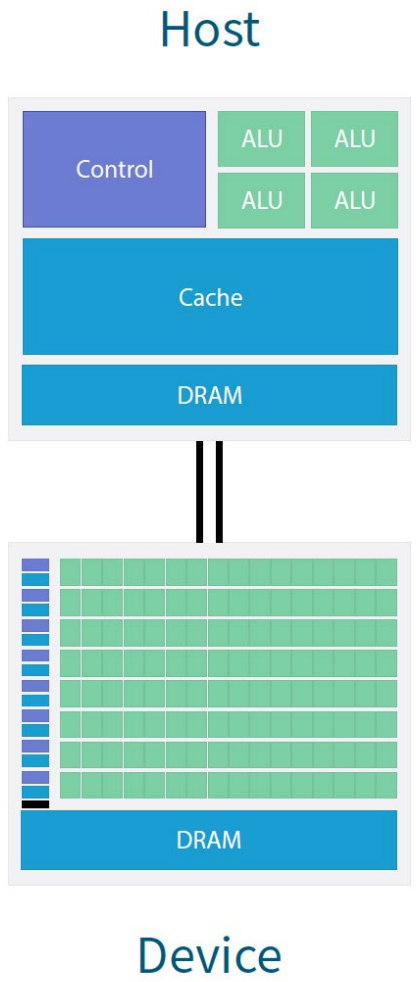
CPU core vs GPU SMX



- The latency in a SMX is hidden thanks to very deep pipelines
- The multithreading in a single CPU core is based on context switching

GPU+CPU

- The winning application uses both CPU and GPU
- CPUs for sequential parts
  - ➡ can be 10X faster than GPU for sequential code
- GPUs for parallel part where throughput wins
  - ➡ can be 100X faster than CPU for parallel code
- The Host-Device connection is done with PCIe-gen3 (16 GB/s) or NVLINK (80 GB/s)
  - ➡Relativelly slow
  - ➡Do as little as possible
- The bandwidth between GPU and video memory is HBM2 (720 GB/s in P100, 900 GB/s in V100)



License

- Part of the material used for this presentation comes from the «GPU Teaching KIT» licensed by NVIDIA and University of Illinois, and has been used and modified according to Creative Commons attribution not-commercial 4.0 (http://creativecommons.org/licenses/by-nc/4.0/legalcode)