UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

GPUs
Heterogeneous Computing

Professor: Dr. Joel Fuentes - jfuentes@ubiobio.cl

Teaching Assistants:
- Daniel López - daniel.lopez1701@alumnos.ubiobio.cl
- Sebastián González - sebastian.gonzalez1801@alumnos.ubiobio.cl

Course website: http://www.face.ubiobio.cl/~jfuentes/classes/hc

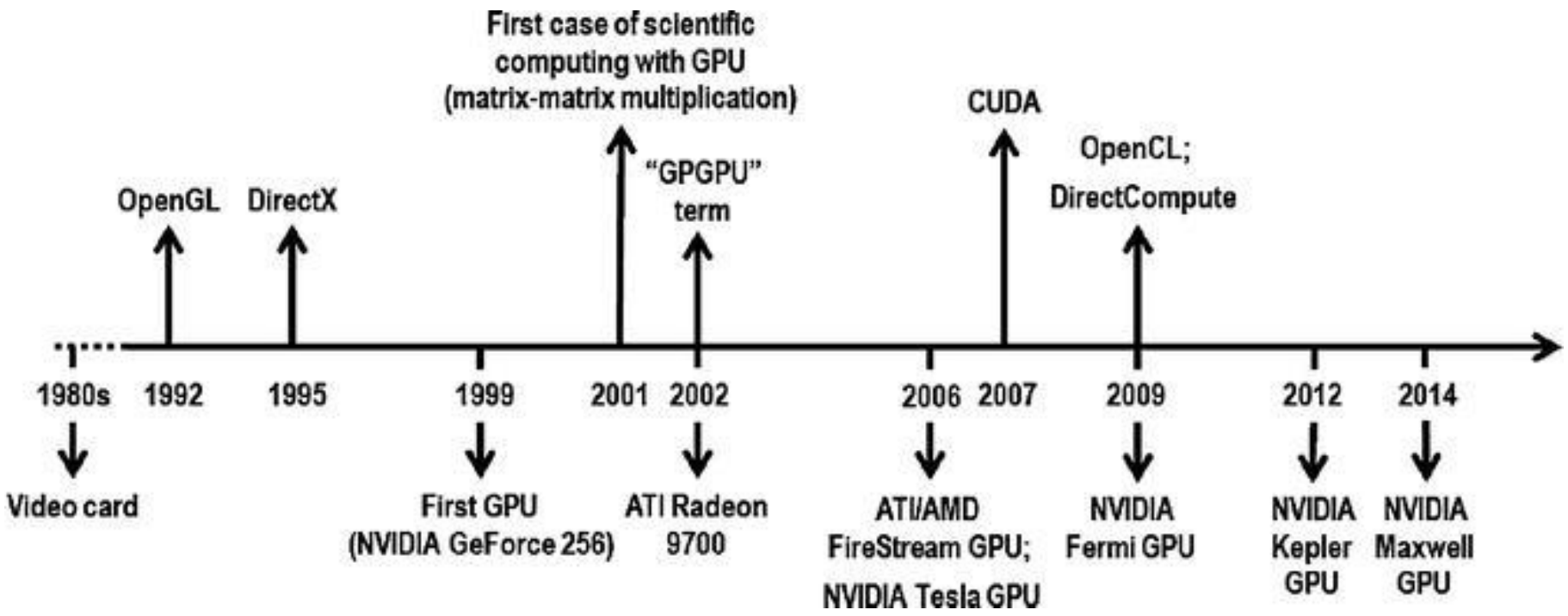
Contents

- History of GPUs
- Sample Application: 3D Rendering
- Nvidia, AMD, Intel architectures
- GPU Programming

History of GPUs

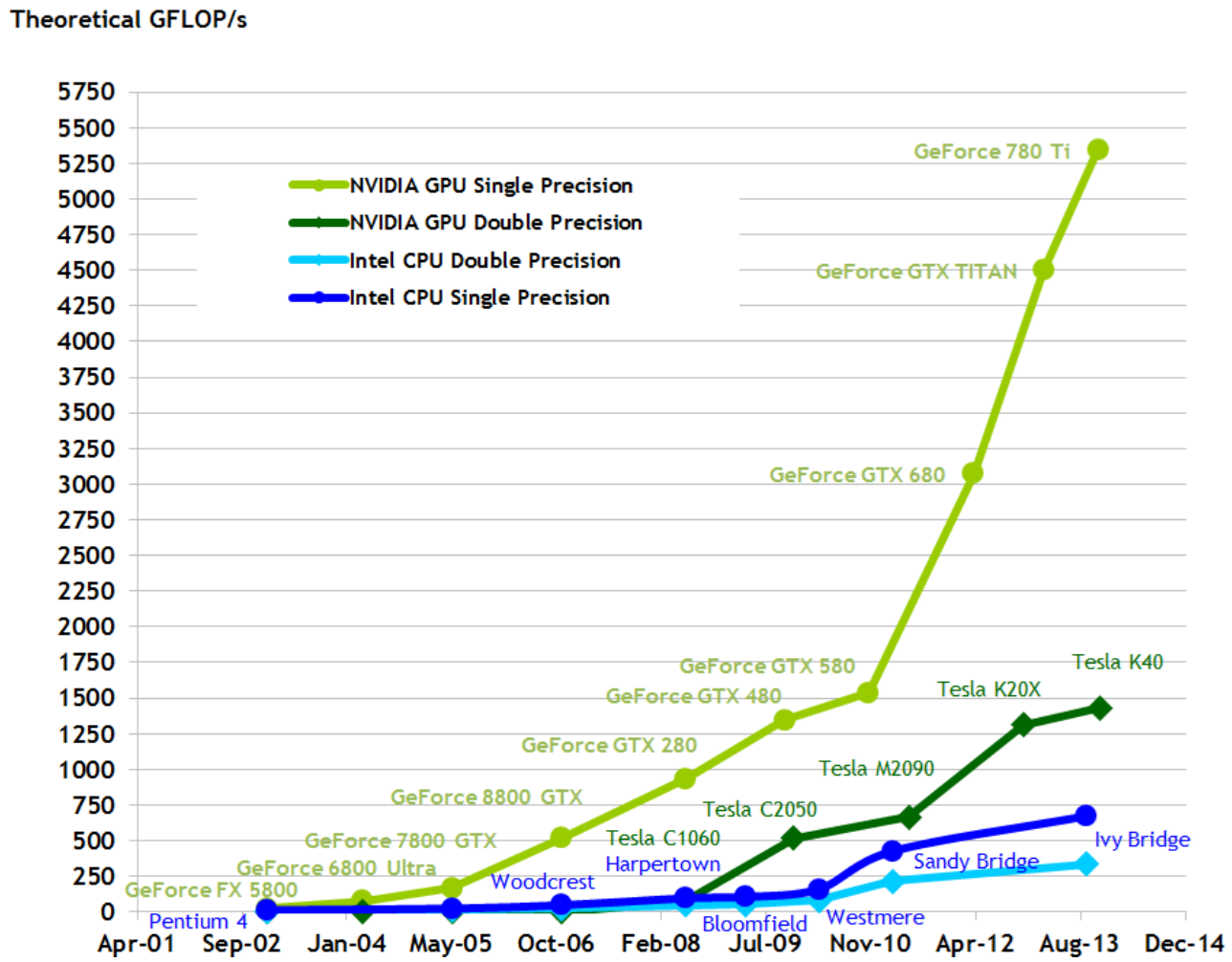
- GPUs (Graphics Processing Units) were initially created for 3D video and graphic acceleration.
- Initially designed with fixed functions without the possibility of flexible programming.
- Since 2001 it began to include the possibility of programming GPUs
- Today its uses have evolved and include:
- Computer vision
- Deep learning
- Scientific computing
- Cryptomining

History of GPUs



https://link.springer.com/referenceworkentry/10.1007%2F978-3-319-17885-1_1606

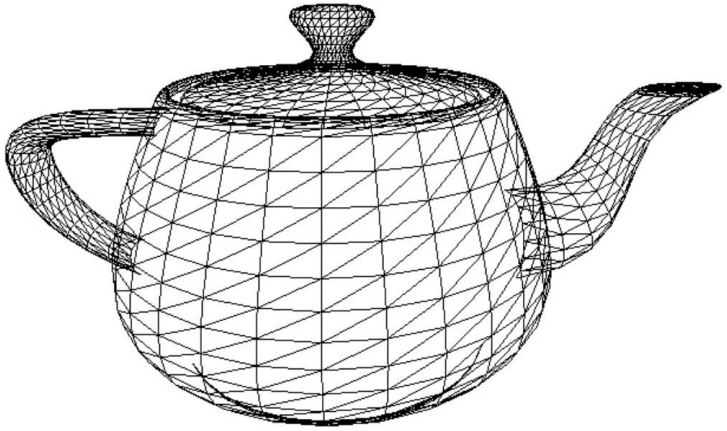
Performance en GFLOP/s vs CPU



http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz3X9Fwos00

Sample Application: 3D Rendering

- Task consists of calculating how each triangle in the 3D mesh contributes to the appearance of each pixel of the rendered image.



Description of the scene based on: mesh of triangles, lights, camera, etc.

Image credit: Henrik Wann Jensen

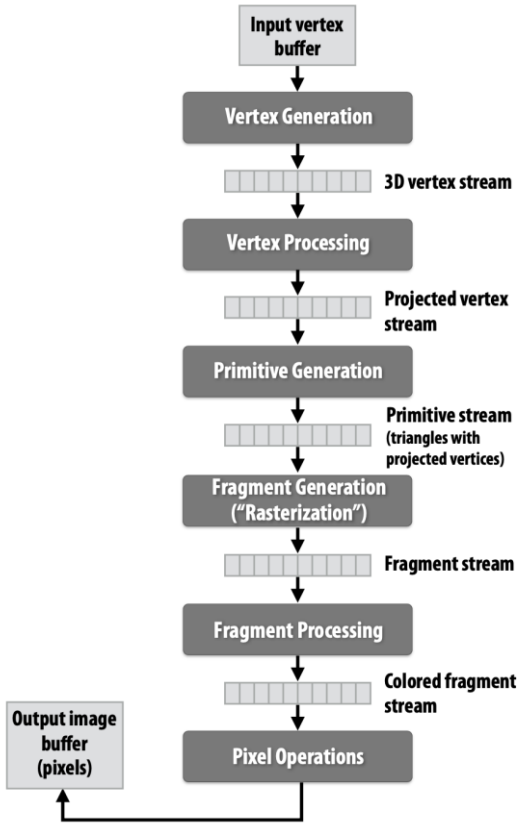
Image of the scene

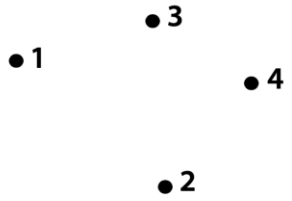
Real-time 3D rendering

Flight Simulator (2021)

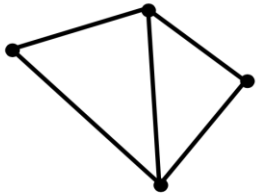
Sample Application: 3D Rendering

- Steps in rendering given a mesh of 3D triangles

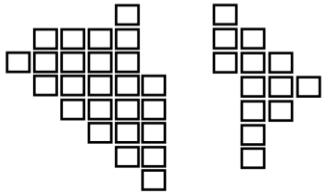




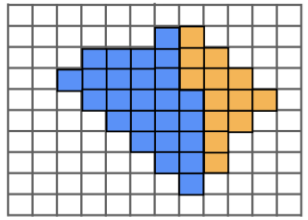
Vertices
(points in space)



Primitives
(e.g., triangles, points, lines)



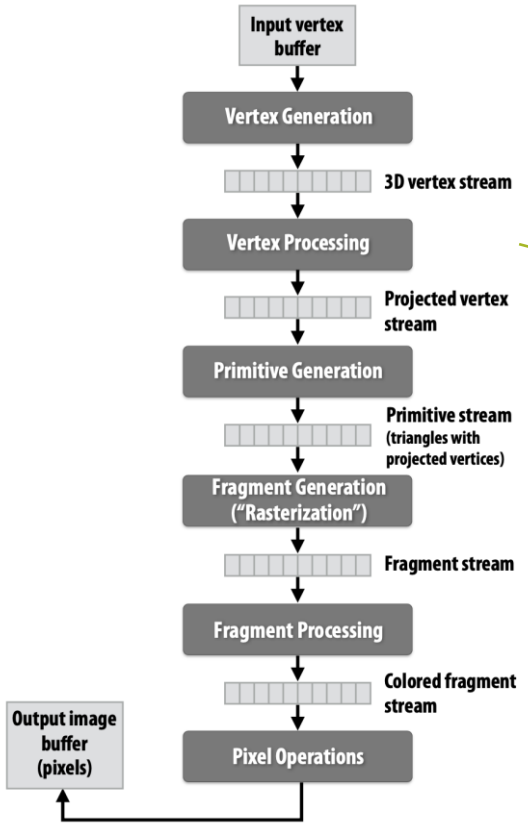
Fragments



Pixels (in an image)

Sample Application: 3D Rendering

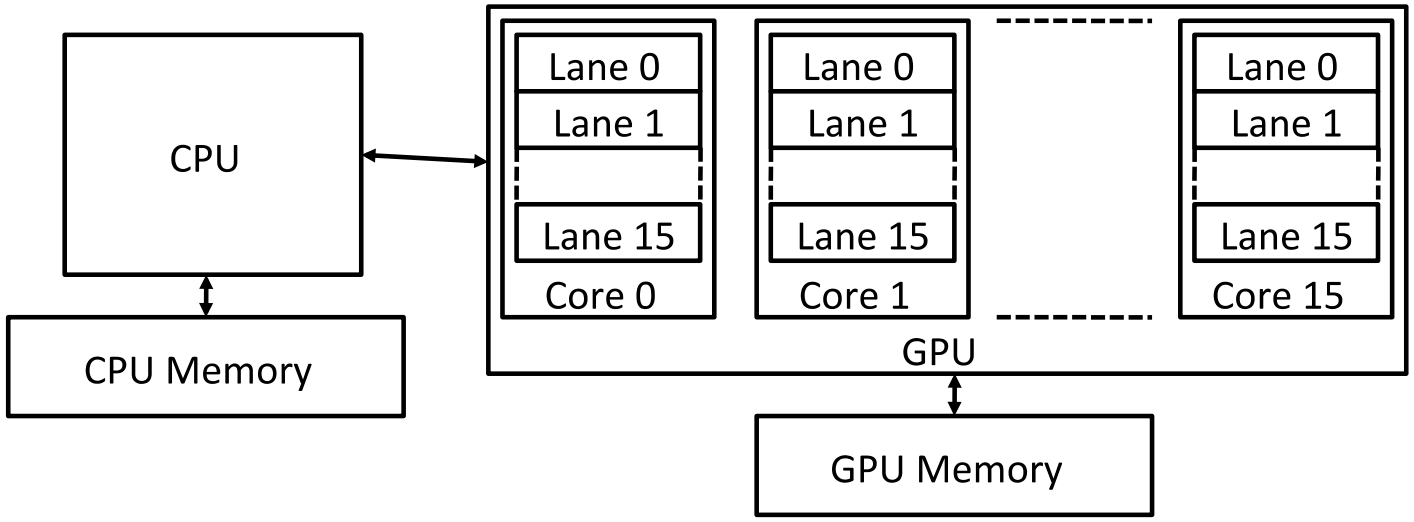
- Steps in rendering given a mesh of 3D triangles



programmers write mini-programs called "shaders" that describe the logic of these steps

GPU Architectures

- Execution model



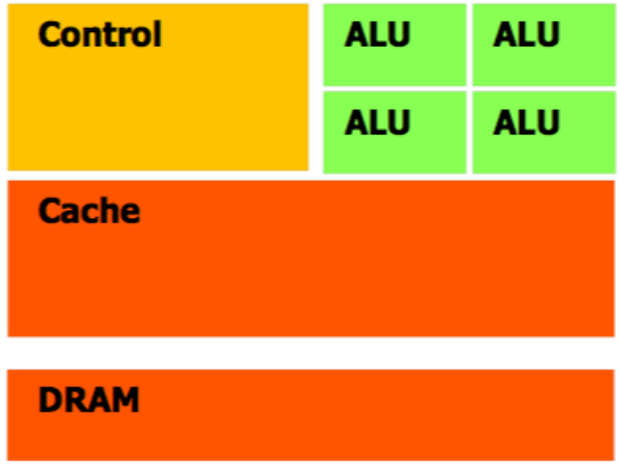
- GPU is built based on multiple simple parallel cores. Each core can execute SIMD (Single Instruction Multiple Data) instructions.
- The CPU sends processing tasks (meshes, buffers, etc.) to the GPU, which distributes the work to its cores.

GPU Architectures

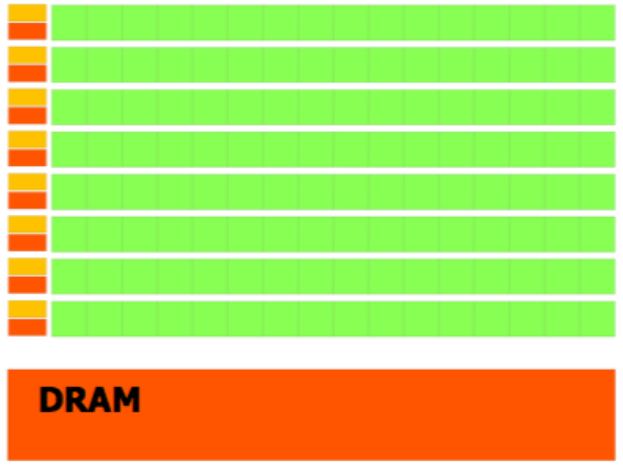
- The cores on the GPU work under the thread model in shared memory.
- GPUs can run hundreds or thousands of threads in parallel.
- GPUs usually have their own DRAM
- GPUs are good for:
  - Data-parallel processing: The same operation executed on many data elements in parallel.
  - Processing with high arithmetic intensity.

GPU Architectures

- Comparison with CPU:
- GPUs occupy more transistors in data processing
- GPUs have smaller caches
- GPU ALU is simpler than CPU ALU



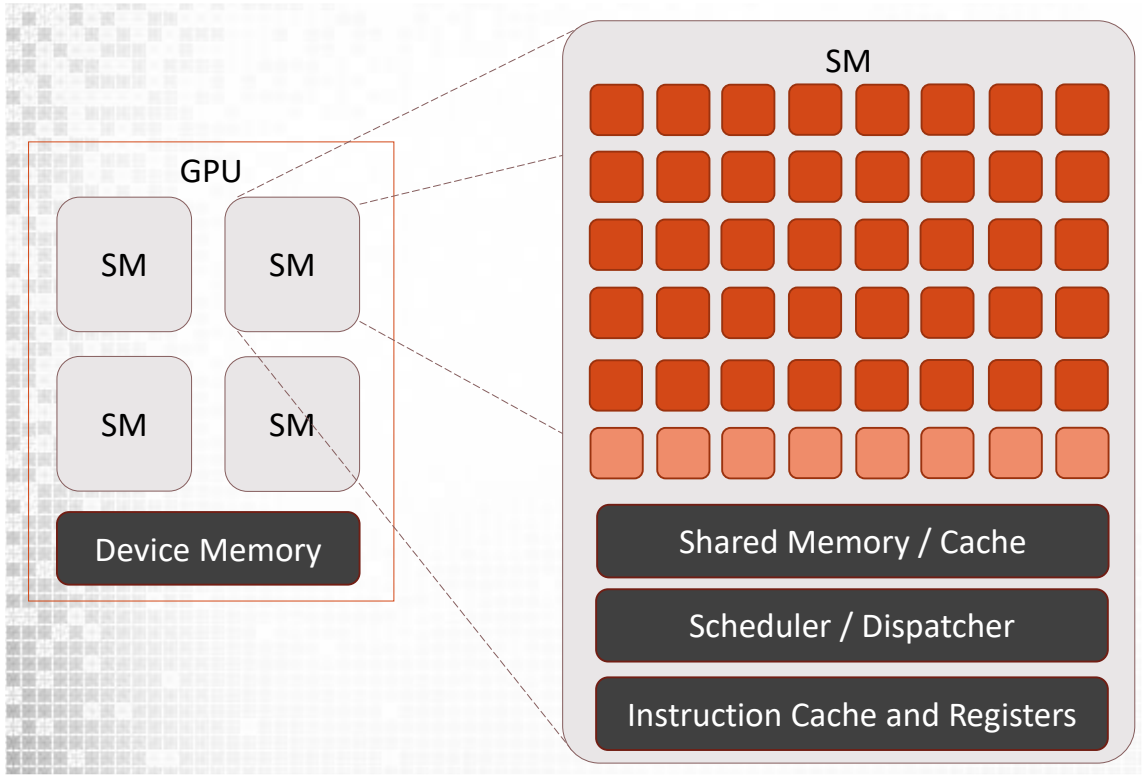
CPU



GPU

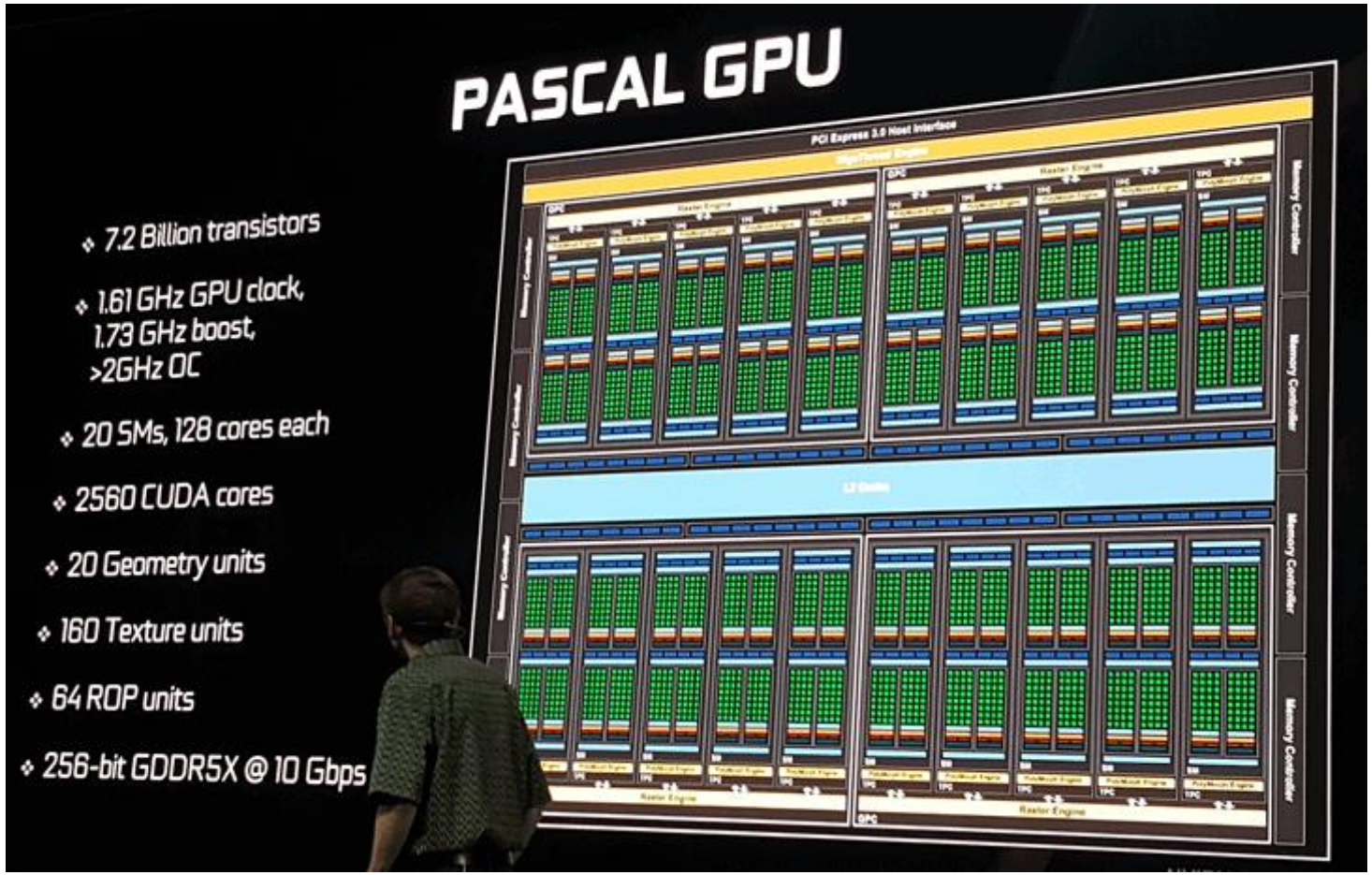
GPU Nvidia

- Have a 2-level hierarchy
- Each Streaming Multiprocessor (SM) has multiple CUDA cores
- The number of SMs varies depending on the type of GPU



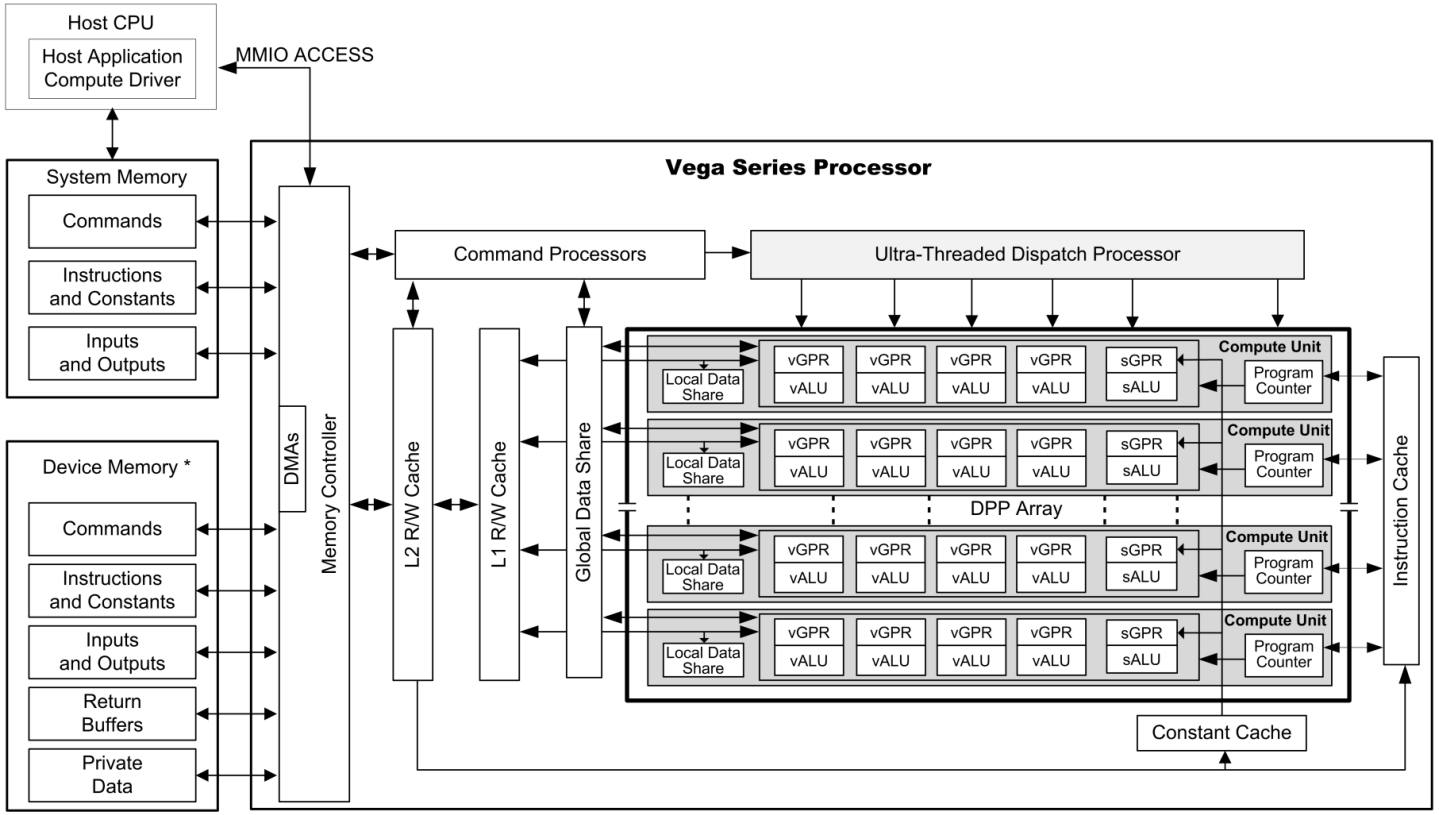
GPU Nvidia

- Image of Pascal architecture (1080 Ti)



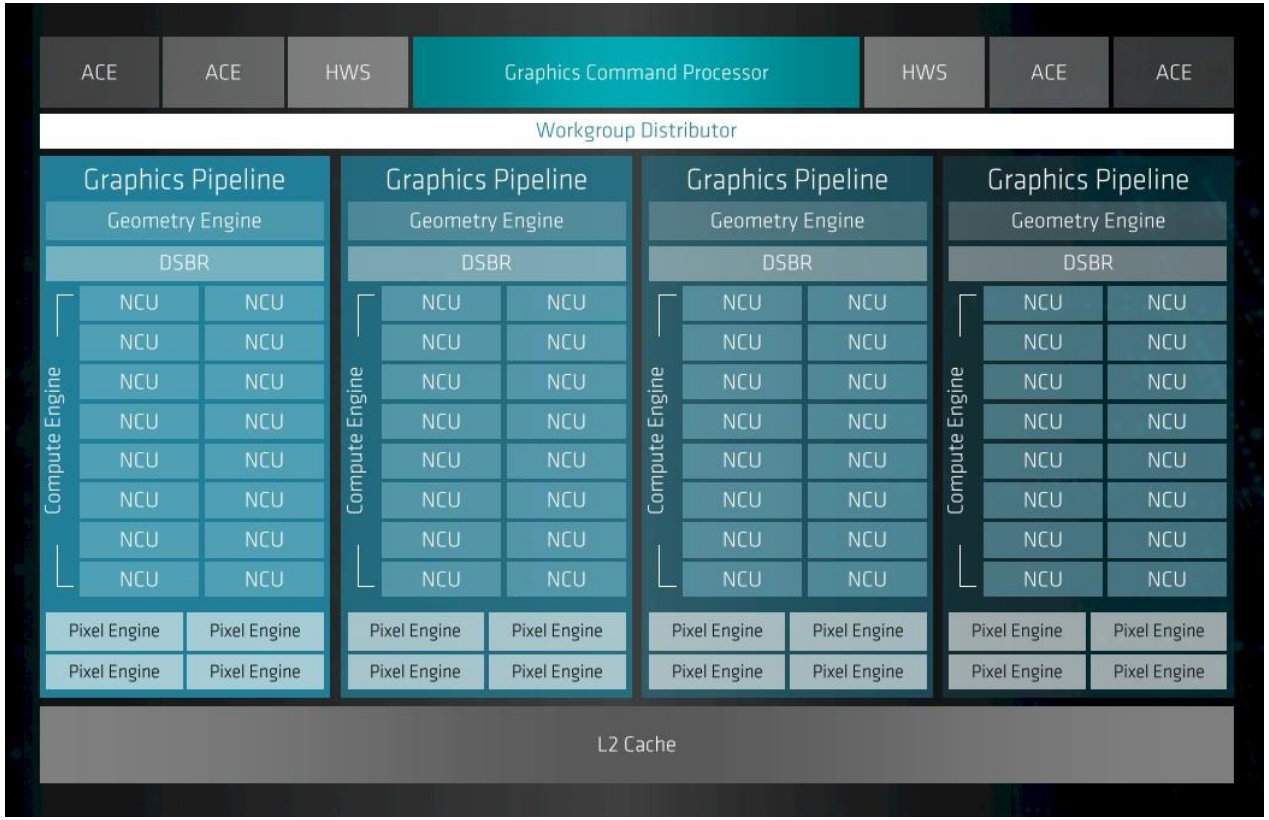
GPU AMD

- GPR: General Purpose Records
- vALU: ALU vector for SIMD processing



GPU AMD

- Vega 20 GPU image
- NCU: Next compute unit



GPU Intel

- Compute unit or core is called Execution Unit (EU)
- Multiple EUs grouped into Subslices
- Multiple Subslices grouped into Slices
- Memory hierarchy up to L3

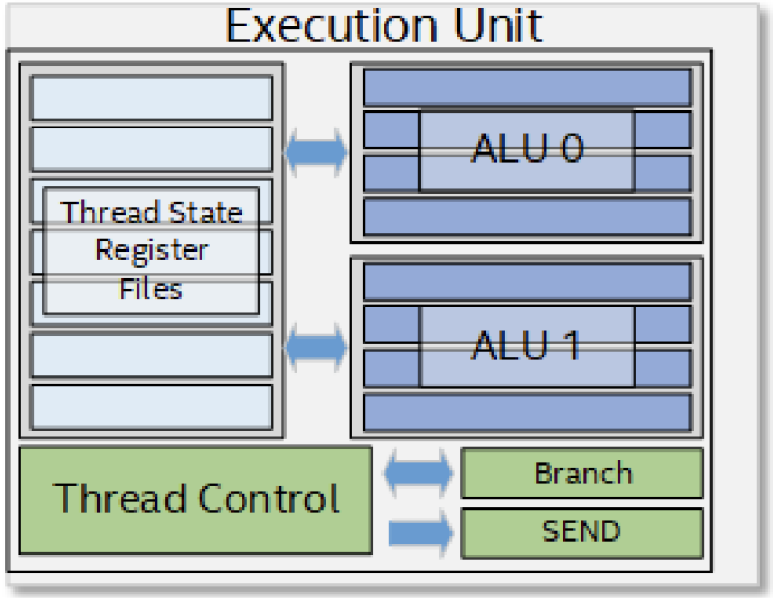
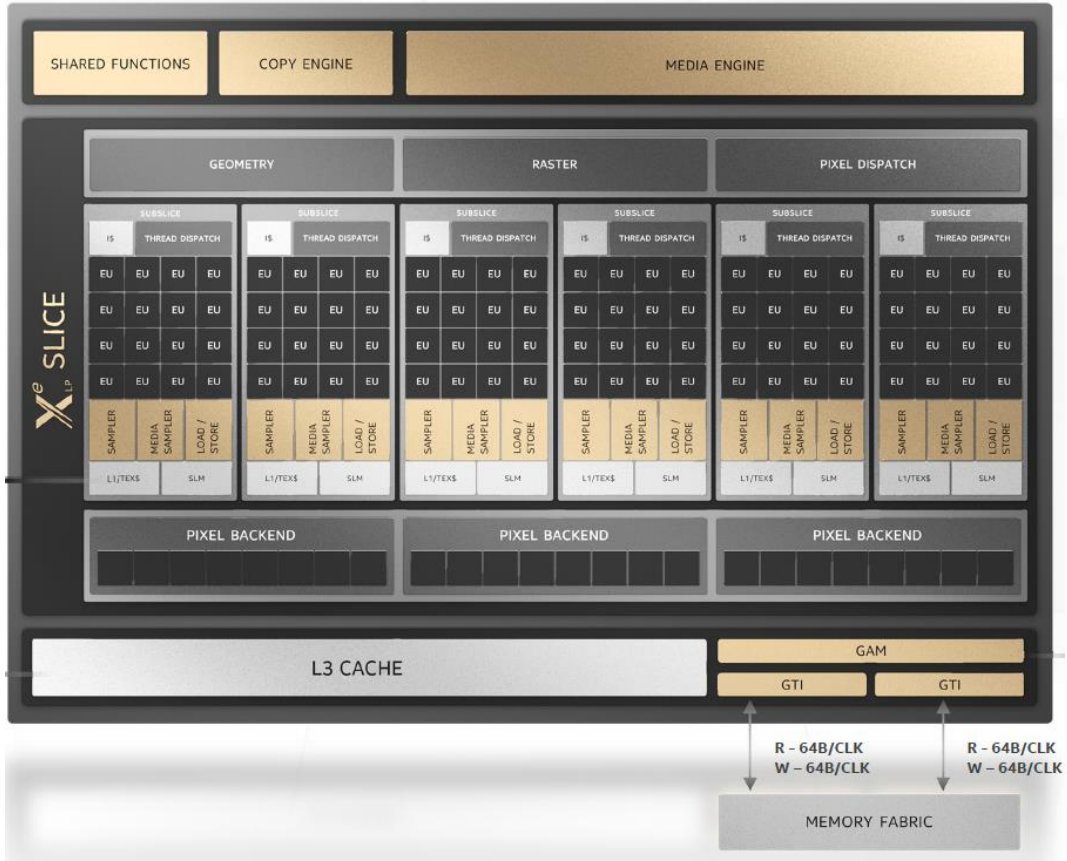


Figure 4: Gen11 detailed block diagram.

GPU Intel

- Xe architecture block diagram (integrated GPU)
- Soon Intel will launch discrete GPUs



GPU Programming

- What do we program GPUs in?
- CUDA (Nvidia)
- OpenCL (open standard)
- OpenACC
- SYCL (open standard)
- DPC++
- In the next unit we will see different programming models.

References

- Parallel Computing, CS 149 (Fall 2019), Stanford University
- Paul Richmond. GPU Arquitectures http://paulrichmond.shef.ac.uk/teaching/COM4521/
- Qin CZ. (2017) Cuda/GPU. In: Shekhar S., Xiong H., Zhou X. (eds) Encyclopedia of GIS. Springer, Cham. https://doi.org/10.1007/978-3-319-17885-1_1606