# GPGPU Computing and SIMD

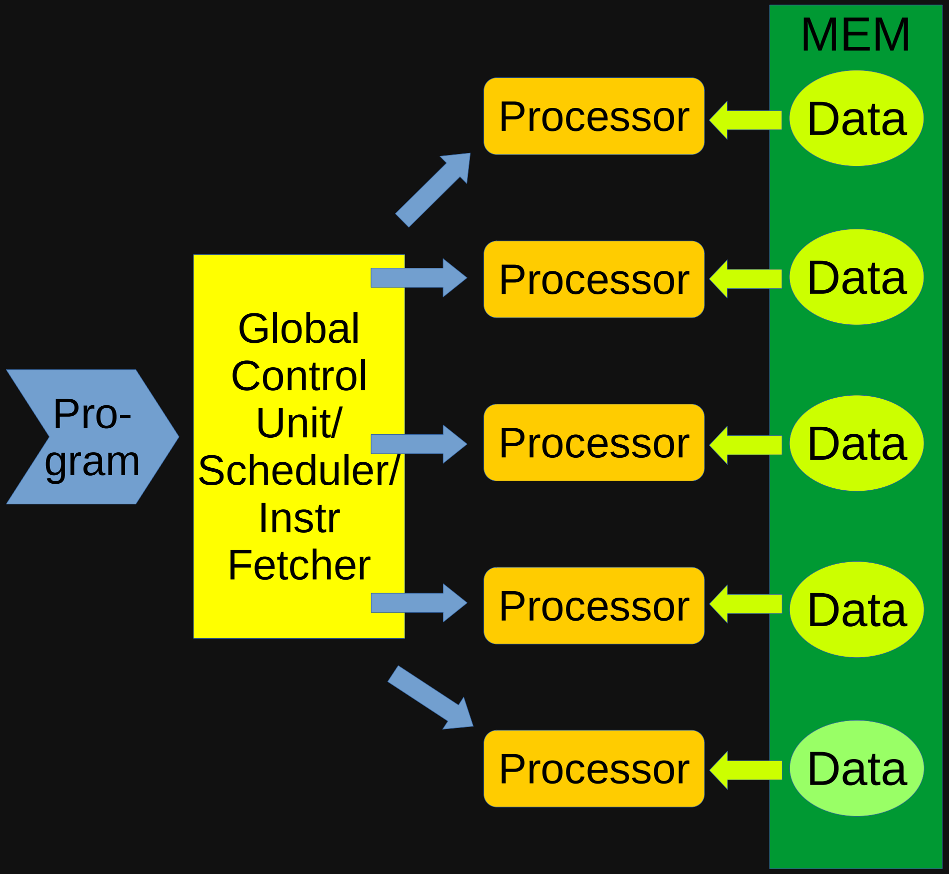
Wei Wang

# Control Structure of Parallel Platforms

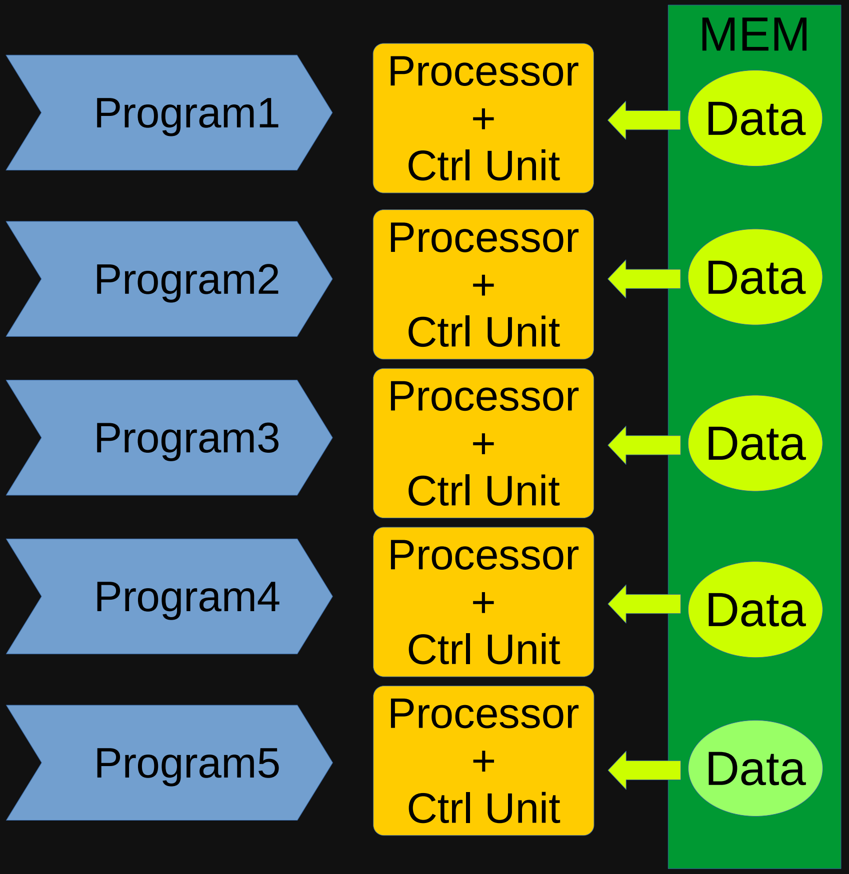
- Processor control structure alternatives
  - work independently
  - operate under the centralized control of a single control unit
- MIMD
  - Multiple Instruction streams
    - each processor has its own control control unit
    - each processor can execute different instructions
  - Multiple Data streams
    - processors work on their own data
- SIMD
  - Single Instruction stream
    - single control unit dispatches the same instruction to processors
  - Multiple Data streams
    - processors work on their own data
- SIMT
  - Similar to SIMD, single instruction stream and multiple data streams
  - SIMT is an extension of SIMD that allows programming SIMD with threads

# SIMD and MIMD Processors

- SIMD




- MIMD



# SIMD Control

- SIMD excels for computations with regular structure
  - media processing, scientific kernels (e.g., linear algebra, FFT)
  - Image processing
  - Machine learning algorithms
  - These workloads are also parallel-friendly
- Most SIMD architectures forego complex branch/control logics and cache/memory management, and dedicate all transistors to processing units
  - Allowing a large number of processing units on a single chip

# SIMD/SIMT Example: Nvidia Pascal Tesla P100 (2016)

- 56 Streaming Multiprocessors (SM)
- Each SM
  - 64 single-precision (FP32) CUDA cores
  - 32 double-precision (FP64) units
  - 16 special functions units
- Total 3584 FP32 cores and 1792 FP64 cores
- Peak FP32 GFLOPS: 10600
- Peak FP64 GFLOPS: 5300

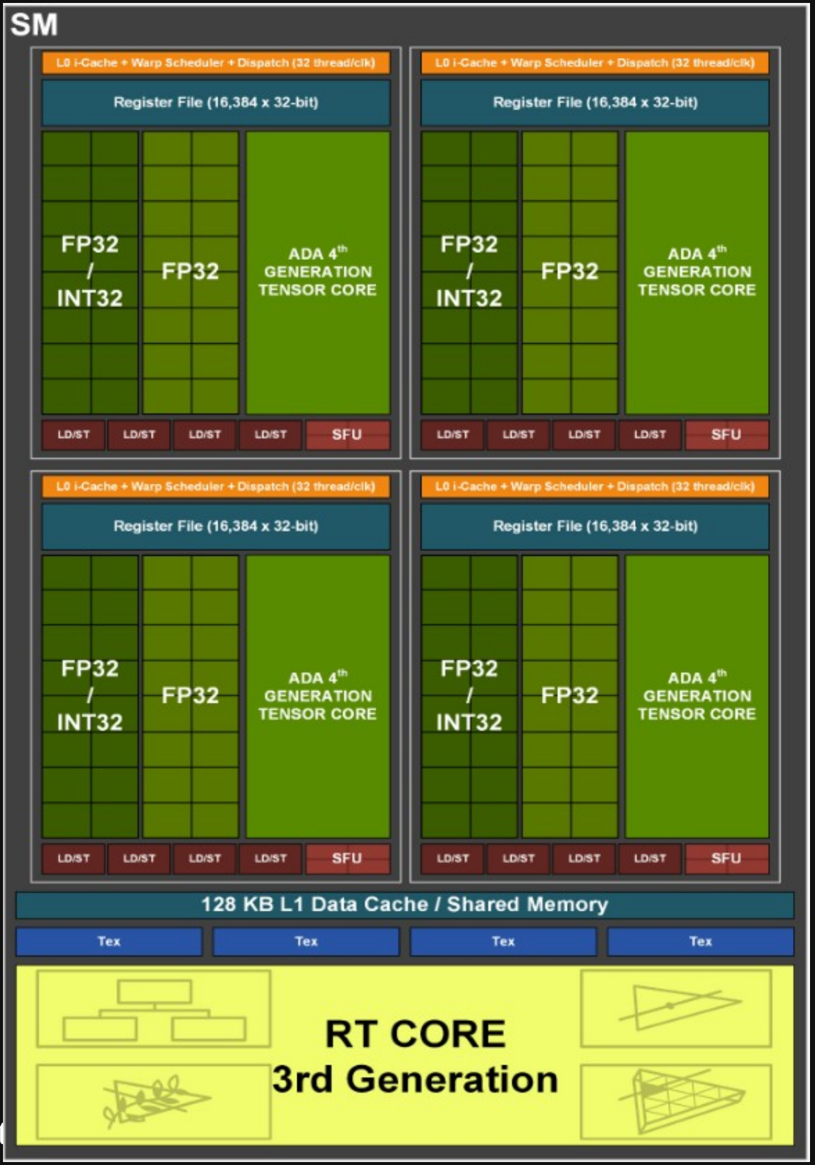
# SIMD/SIMT Example: Nvidia Ampere P102 (2020)

- Each Streaming Multiprocessors (SM):
  - 64 FP32/INT32 Cores
    - INT32 cores support INT4, INT8 and INT32 operations
    - FP32 cores support FP32 and FP16 operations
  - 64 FP32 Cores
  - 2 FP64 cores (not in the figure)
  - 8 Tensor Cores
  - 1 RT (Ray Tracing) Core
  - 256KB Register File

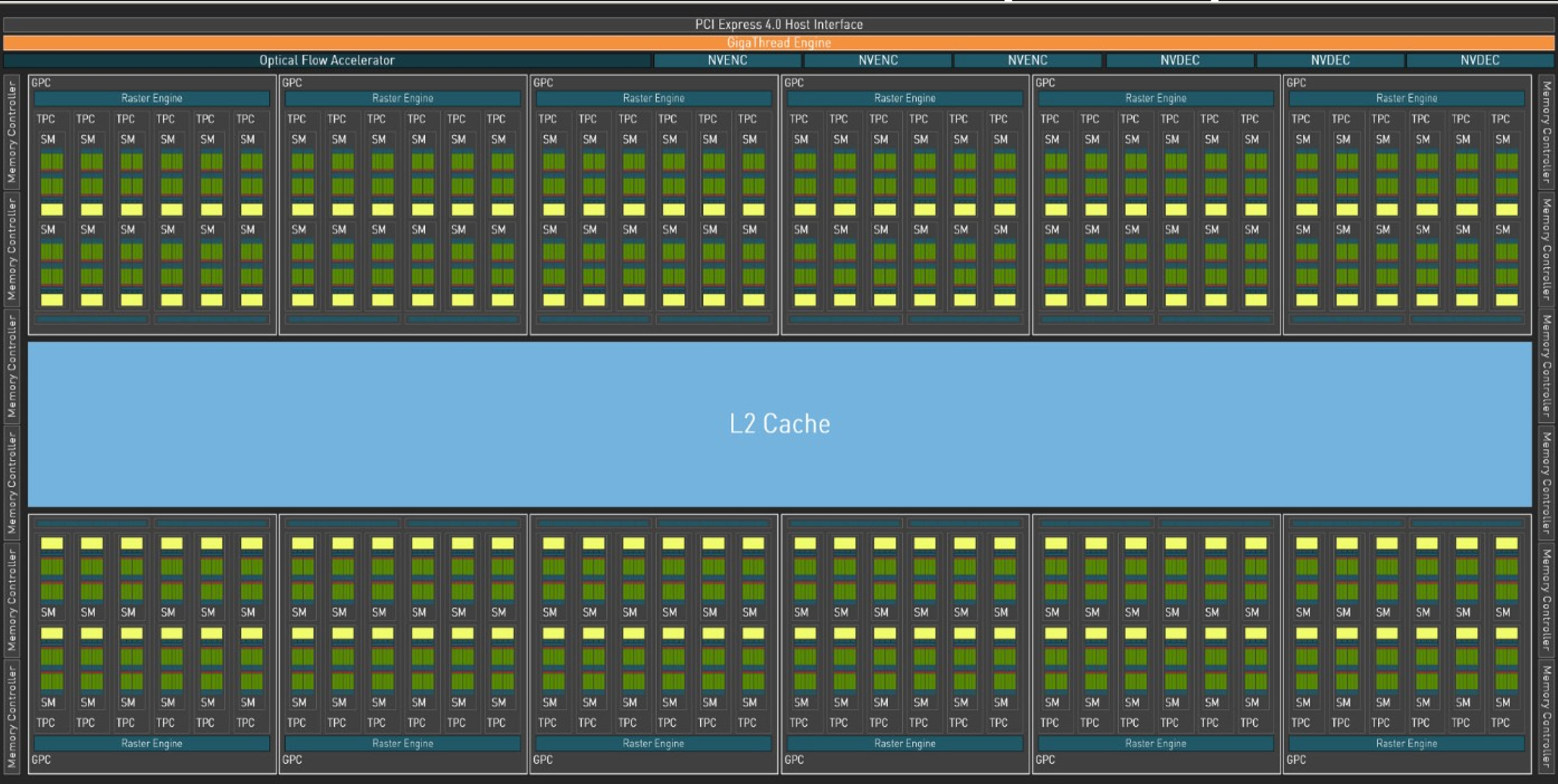
# SIMD/SIMT Example: Nvidia Ampere  P102 (2020)

# SIMD/SIMT Example: Nvidia Ada AD102/RTX4090 (2022)

- Each Streaming Multiprocessors (SM):
  - 64 FP32/INT32 Cores
    - INT32 cores support INT4, INT8 and INT32 operations
    - FP32 cores support FP32 and FP16 operations
  - 64 FP32 Cores
  - 2 FP64 cores (not in the figure)
  - 4 Tensor Cores
  - 1 RT (Ray Tracing) Core
  - 256KB Register File



# SIMD/SIMT Example: Nvidia Ada AD102/RTX4090 (2022)

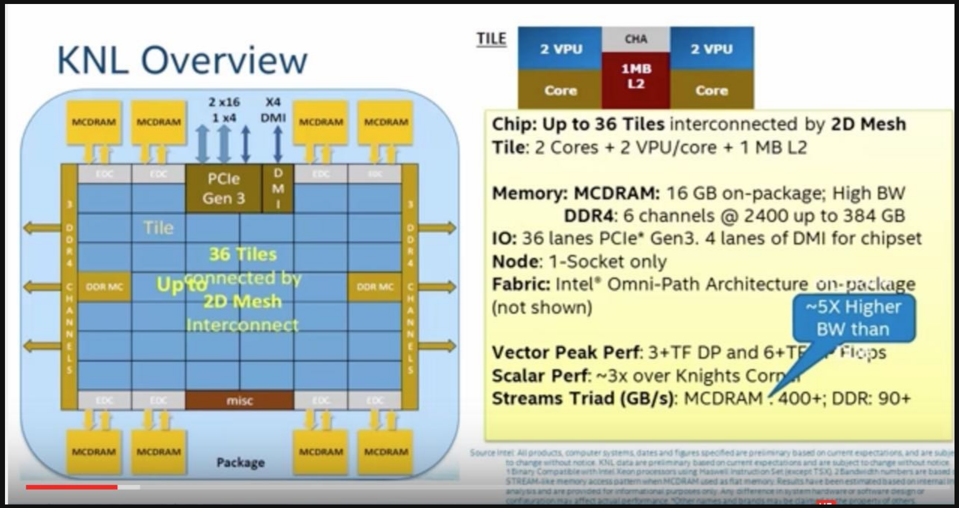


# SIMD/SIMT Example: Nvidia Ada AD102/RTX4090 (2022)


- 12 GPCs
- 64 TPCs
- 128 Sms
- 16384 CUDA cores
- 512 Tensor cores
- 82.6 FP32/FP16/BF16 TFLOPS (non-tensor)
- 41.3 INT32 TFLOPS
- Memory Bandwidth 1008GB/s
- L2 Cache 72MB
- 76.3 B transistors, 608.5mm, 450w

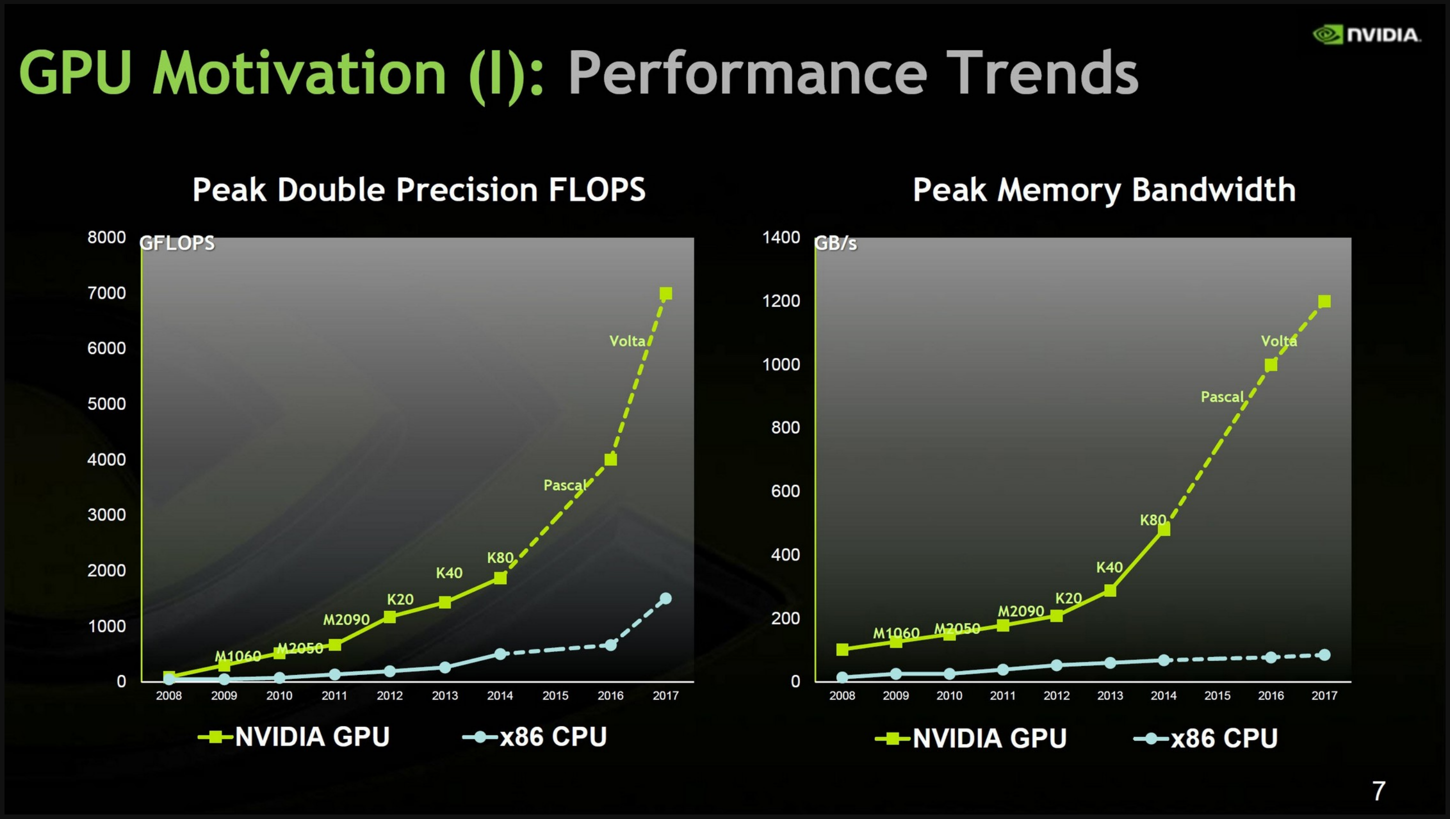
# SIMD Example: Intel Xeon Phi 7290 Knights Landing

- 72 cores
- Each core
  - Four SMT threads
  - 512-bit vector units
  - 32KB L1 cache
  - 1MB L2 cache
- Max GFLOPS: 3000



# Why SIMD?

- SIMD offers much higher theoretical peak performance over MIMD (CPU) per watt



# The Actual Difference Between CPU and GPU

- A 2010 Intel study suggests that GPU is only 2.5x faster than CPU on average
- A 2015 study shows that GPU is about 0 to 60x faster than CPU for several machine learning workloads
  - Note that the implementation is probably not optimized
  - These are the results of one GPU vs one CPU.

# CPU Core V.S. GPU Core

- For an Nvidia GPU, a core has
  - One 32-bit floating point unit (FPU)
  - One 32-bit Integer unit (ALU)
  - Additionally, a few 64-bit FPUs and functional units are located outside GPU cores
  - Newer version of GPU also supports L1 cache per SM
  - Designed and optimized for graphic processing
- For an Intel Processor, a core typically has
  - 4 ALUs
  - 2 256-bit FPU
  - 4 256-bit Vector ALU
  - 2-4 LD/ST units, LEAL units
  - Complex out-of-order execution management, branch prediction and memory disambiguation
  - 64KB L1 cache
  - 256KB L2 cache
  - Designed and optimized for general computing

# GPGPU Programming

- GPGPU Programming: General-purpose computing on graphics processing units
- Motivation
  - Certain problems are similar to graphic applications in that they involve significant number of linear algebra operations and stream data processing
  - These problems also have limited data reuse and branches, similar to graphic applications
  - GPUs are faster than CPUs with these problems because the large number of processing units
- Therefore, It is both viable and beneficial to solve these problems on GPU