JORNADA DE DIVULGACIÓN DE APLICACIONES CIENTÍFICAS SOBRE PROCESADORES GRÁFICOS Y QUANTUM COMPUTING

JGPUQC 2024 – UNIVERSIDAD DE ALICANTE

Manuel Benavent-Lledó <mbenavent@dtic.ua.es>
David Mulero-Pérez <dmulero@dtic.ua.es>
José García-Rodríguez <jgarcia@dtic.ua.es>

DAVID

MULERO PÉREZ

- **Grado en Ingeniería Multimedia**
  - Universidad de Alicante
  - 2017 – 2021

- **Máster en Ciencia de Datos**
  - Universidad de Alicante
  - 2021 – 2022

- **Doctorado en Informática (Virtual Reality and Deep Learning)**
  - Univerisdad de Alicante
  - 2022 – Actualidad

MAIL: **DMULERO @ DTIC.UA.ES**

MANUEL

BENAVENT LLEDÓ

- **Grado en Ingeniería Informática**
  - Universidad de Alicante
  - 2017 – 2021

- **Máster en Automática y Robótica**
  - Universidad de Alicante
  - 2021 – 2022

- **Doctorado en Informática (Deep Learning and Computer Vision)**
  - Univerisdad de Alicante
  - 2022 – Actualidad

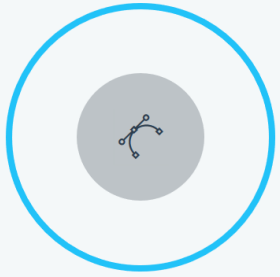
MAIL: **MBENAVENT @ DTIC.UA.ES**

DEDICACIÓN

INVESTIGACIÓN

3DPLAB

The 3D Perception Lab at the University of Alicante is a group of researchers interested in the intersection of machine learning and computer vision. Our research mission focuses on various aspects of perception often related with mobile robotics in which we exploit 3D data as the main source of information. Some of our research lines include object recognition, semantic segmentation, rigid and non-rigid registration, visual localization and mapping, behavior analysis, and depth estimation. Apart from those general lines we are also highly interested in making those solutions run efficiently by leveraging GPU acceleration using CUDA. Aside from 3D data as our backbone, we are also tied together by our shared vision in the great potential of artificial intelligence, mainly deep learning, which we try to apply and push its limits in every project we work on.



DEEP LEARNING

Artificial intelligence applied to computer vision and robotics (semantic segmentation, depth estimation, scene understanding, object recognition, localization and maping).

3D COMPUTER VISION

Traditional computer vision methods and new challenges for 3D data (rigid registration, non-rigid registration, reconstruction).

GPU COMPUTING

Acceleration of computer vision methods and artificial intelligence pipelines for real-time execution and maximum efficiency.

BEGINNINGS AND EVOLUTION OF GRAPHICS PROCESSORS (GPU'S)

Manuel Benavent-Lledó <mbenavent@dtic.ua.es>
David Mulero-Pérez <dmulero@dtic.ua.es>
José García-Rodríguez <jgarcia@dtic.ua.es>

CONTENIDO

MOORE'S LAW

GORDON MOORE

Cramming More Components onto Integrated Circuits. Gordon E. Moore, 1965

“ THE NUMBER OF TRANSISTORS ON A CHIP DOUBLES EVERY 12 MONTHS”

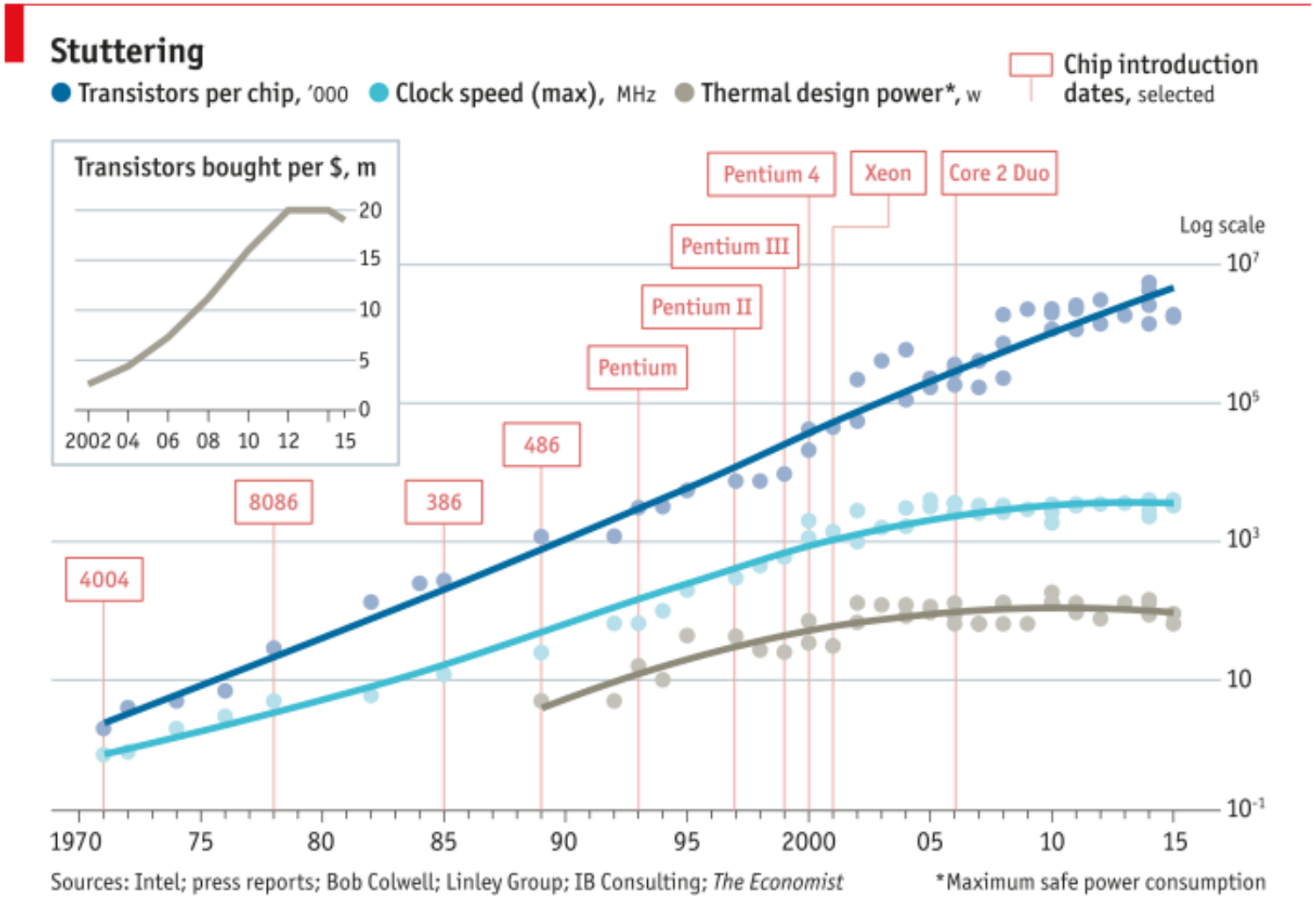
– GORDON MOORE, COFUNDADOR DE INTEL, 1965

“ THE NUMBER OF TRANSISTORS ON A CHIP DOUBLES EVERY 24 MONTHS”

– GORDON MOORE, COFUNDADOR DE INTEL, 1975

MOORE'S LAW

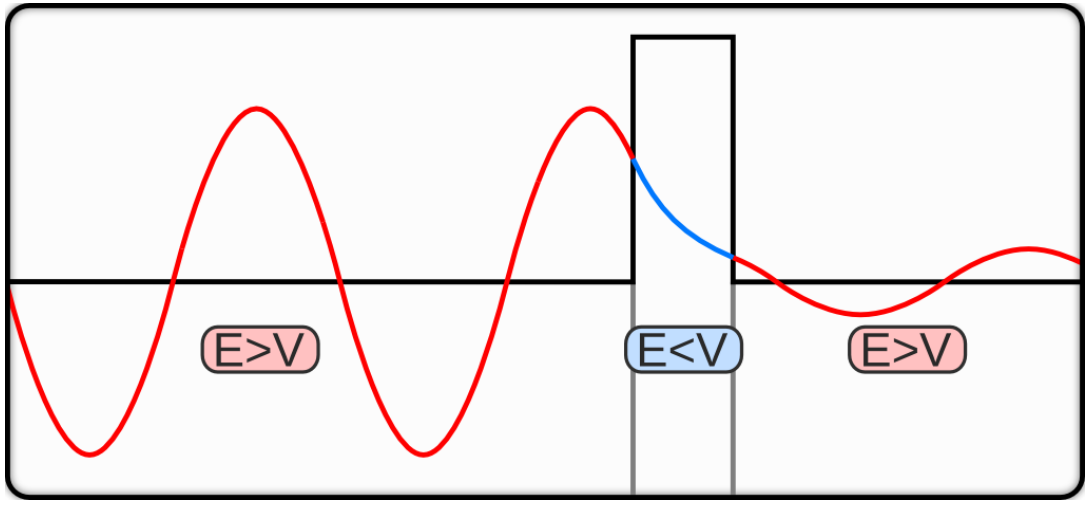
EVERY TWO YEARS THE NUMBER OF TRANSISTORS DOUBLES

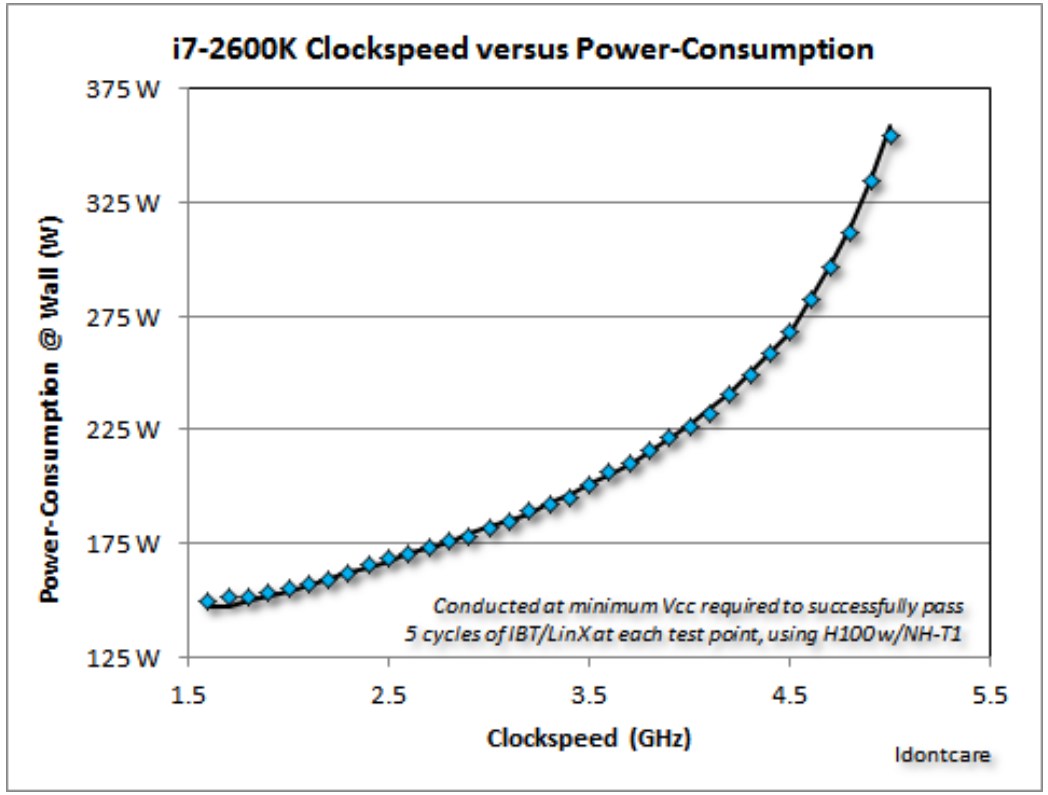


MOORE'S LAW

PROBLEMS: TRANSISTOR SIZE AND TUNNEL EFFECT
ENERGY CONSUMPTION AND HEAT DISSIPATION

Transistors are reaching their size limits, which can negatively affect the efficiency and reliability of integrated circuits.



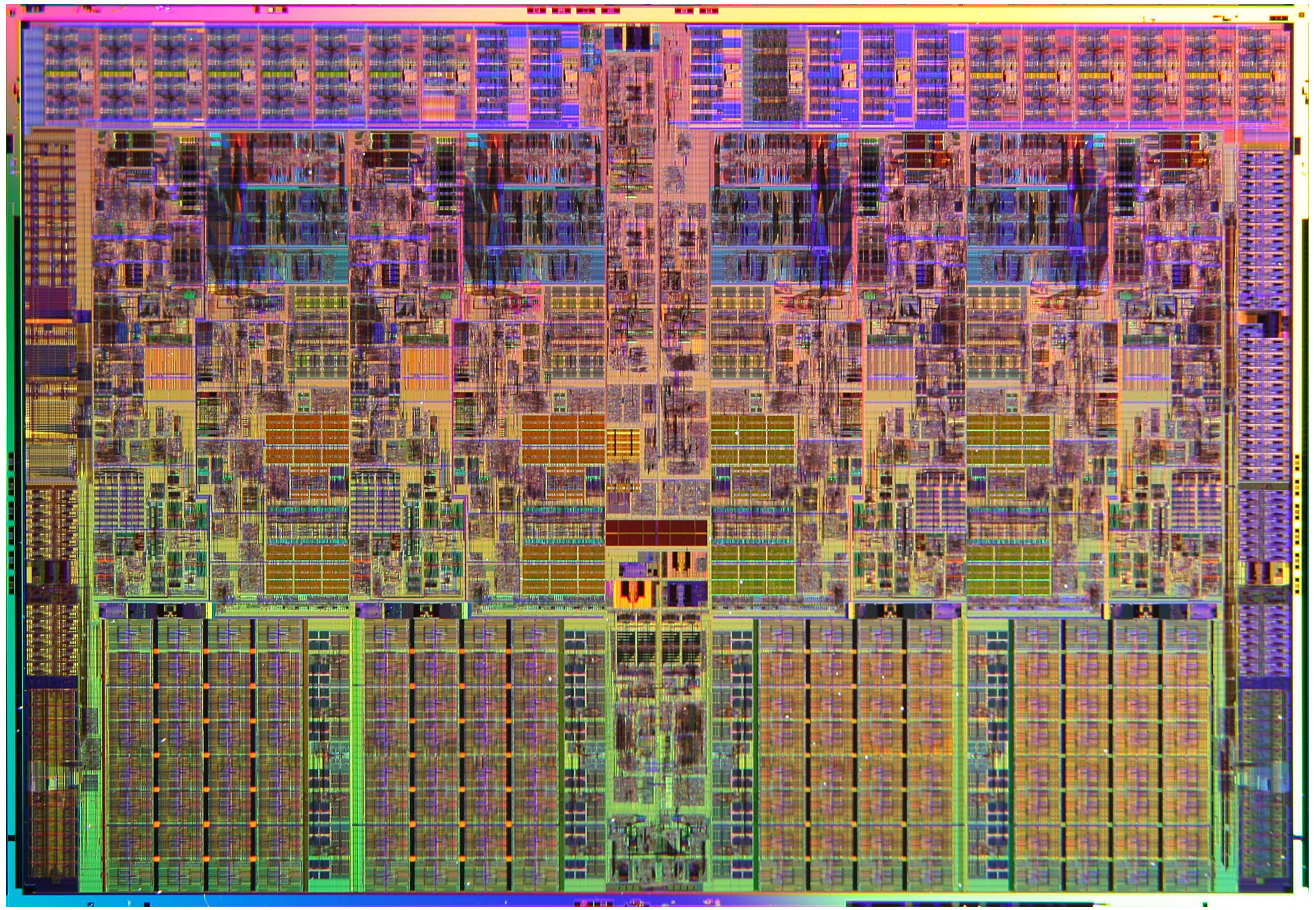


MOORE'S LAW

HOW TO CONTINUE SCALING? MULTICORE

The need to continue scaling processing capacity has led to the adoption of a **multi-core approach** to processor architecture.

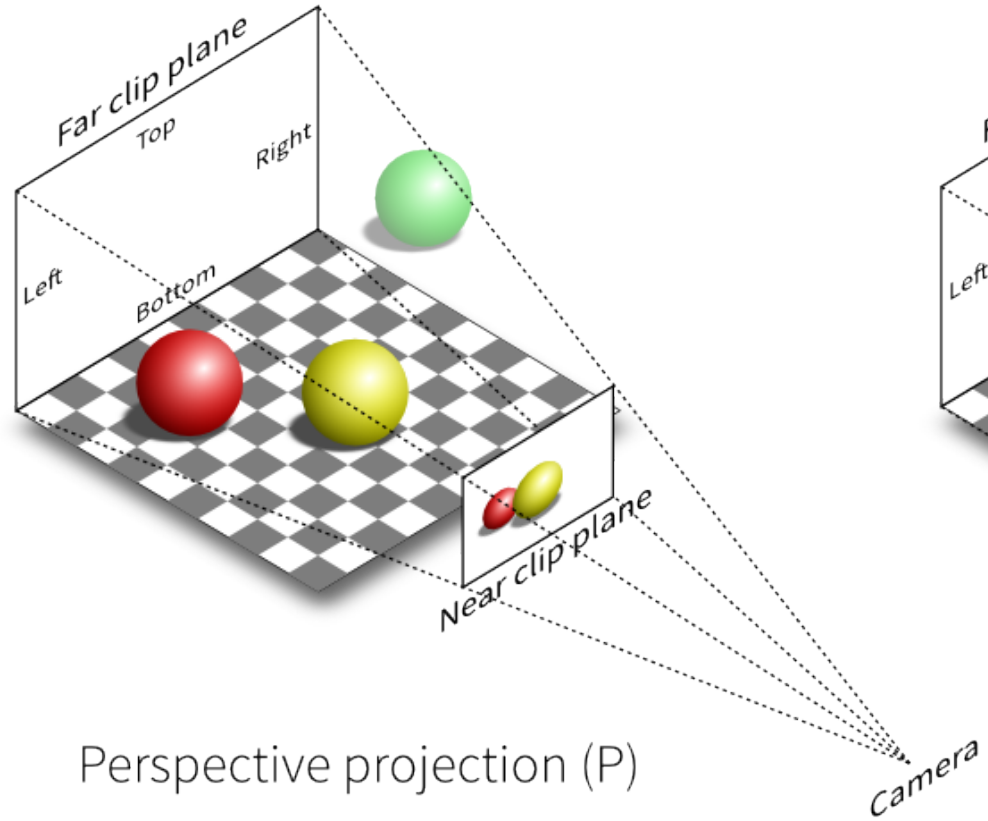
Multi-core processors have multiple processing cores on a single chip, allowing for increased performance and better energy consumption utilization.



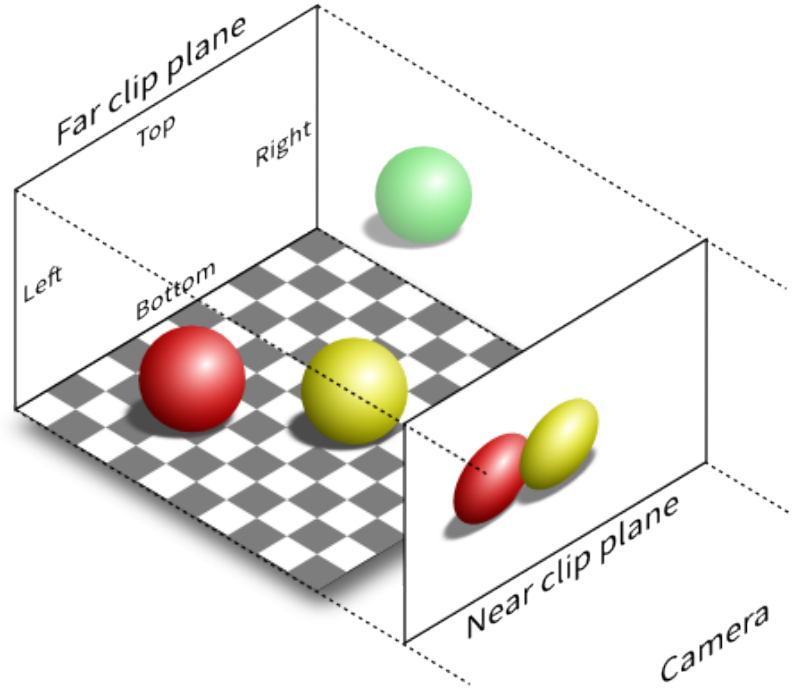
THE GRAPHICS PROCESSING UNIT (GPU)

THE GRAPHIC PIPELINE: RENDERING

The graphic pipeline is the set of operations that are performed to generate an image or graphic in a computer system.



Perspective projection (P)

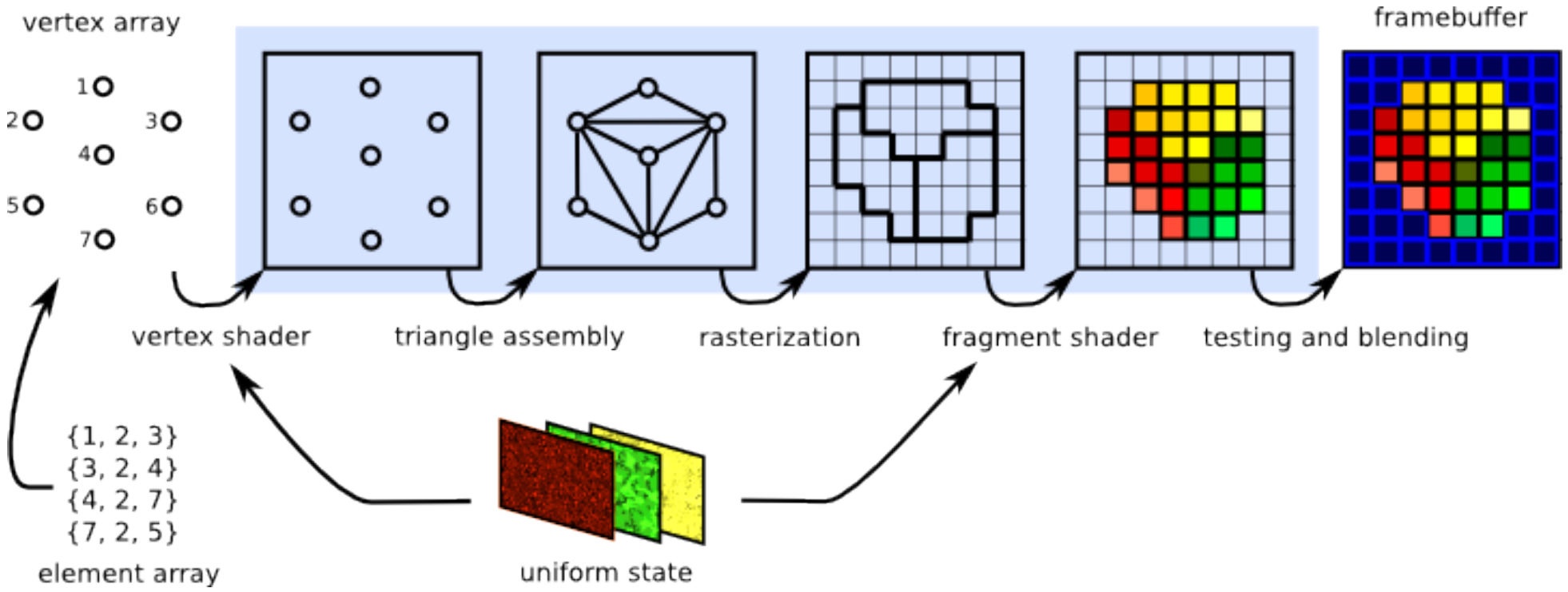


Orthographic projection (O)

THE GRAPHICS PROCESSING UNIT (GPU)

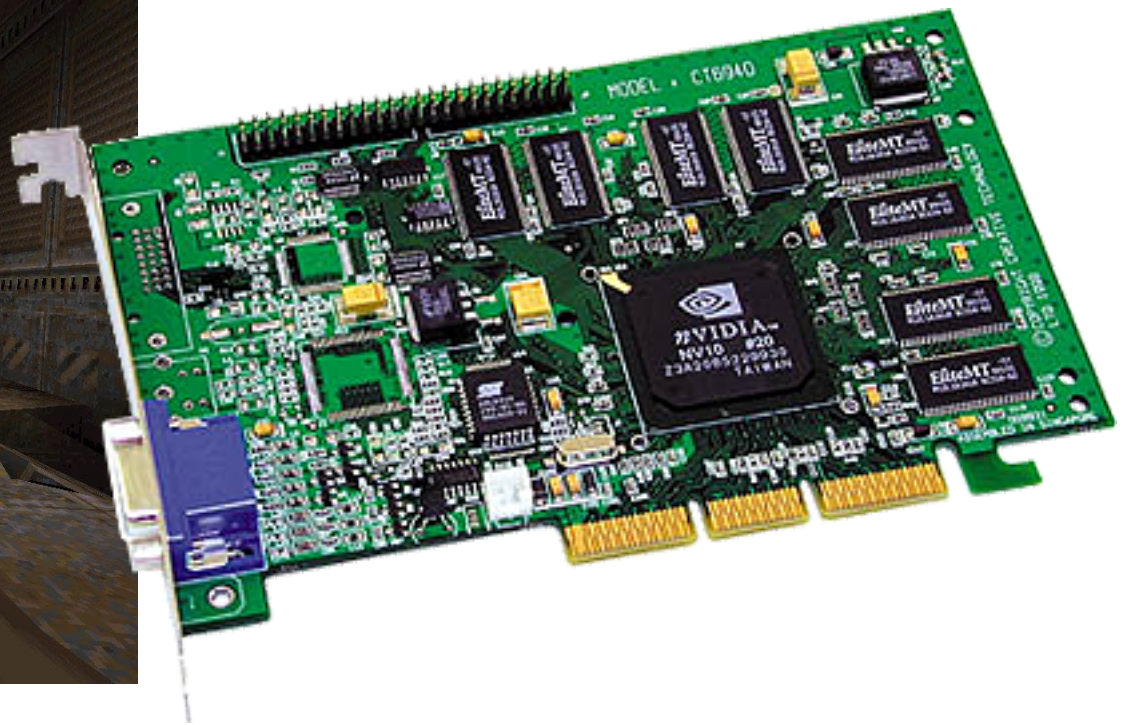
THE GRAPHIC PIPELINE: RENDERING

The rendering stage is a crucial part of the graphic pipeline, as it is responsible for **calculating and processing the image in real time**.



THE GRAPHICS PROCESSING UNIT (GPU)

GEFORCE 256 (1999)



GeForce 256 was marketed as "the world's first GPU"

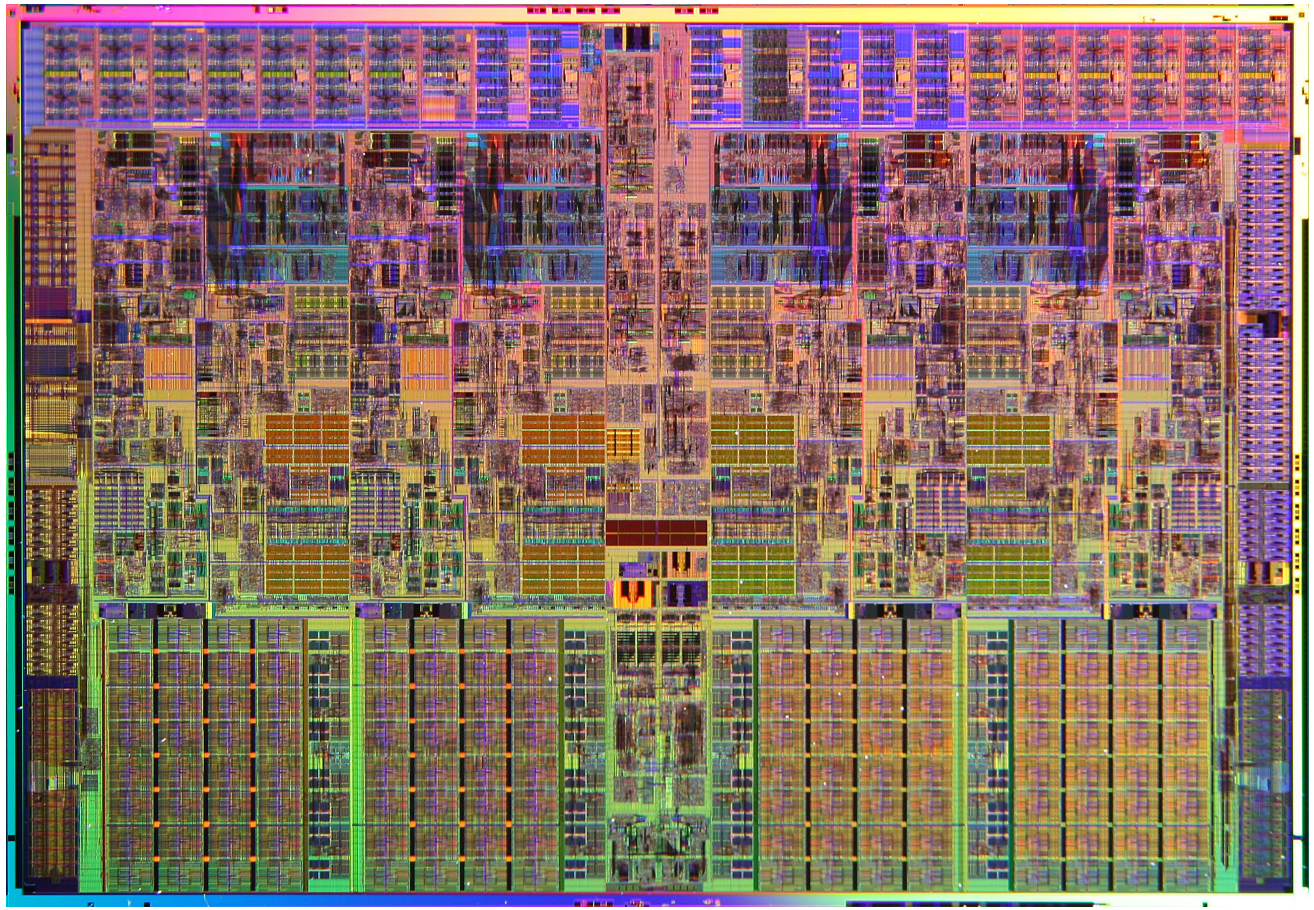
THE GRAPHICS PROCESSING UNIT (GPU)

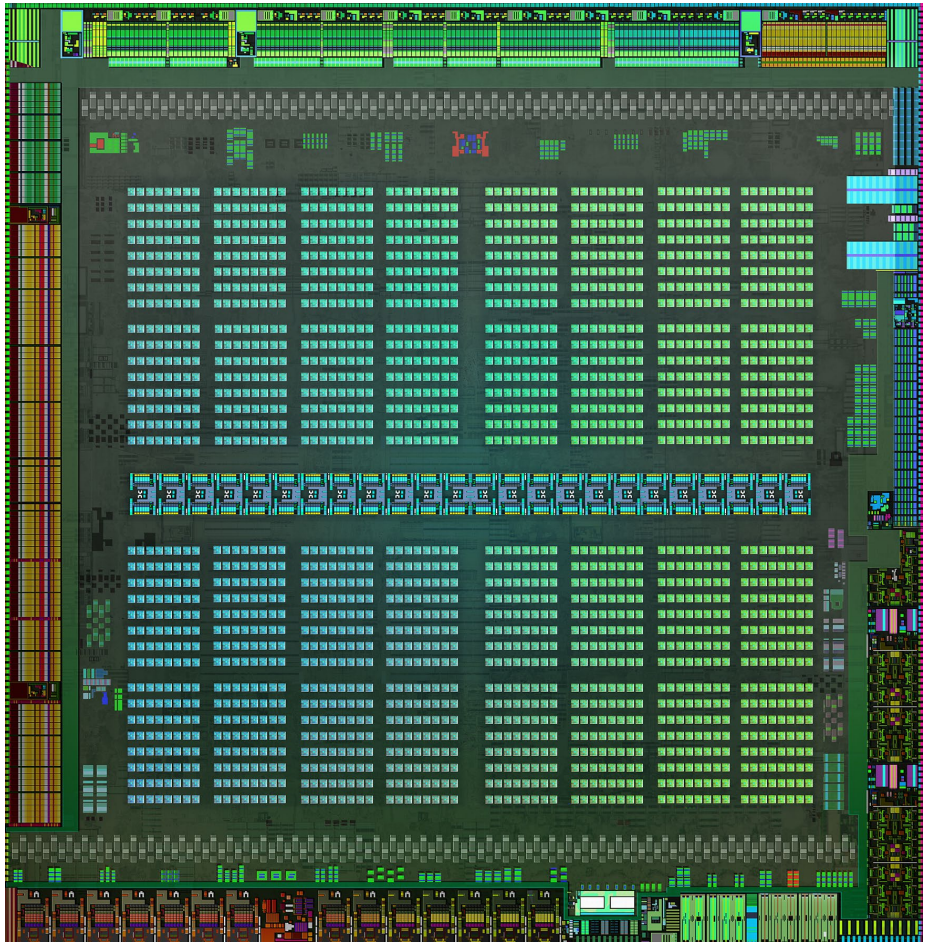
GEFORCE 256 (1999)

"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 GRAPHICS PROCESSING UNIT (GPU)

THE ESSENCE OF THE GPU



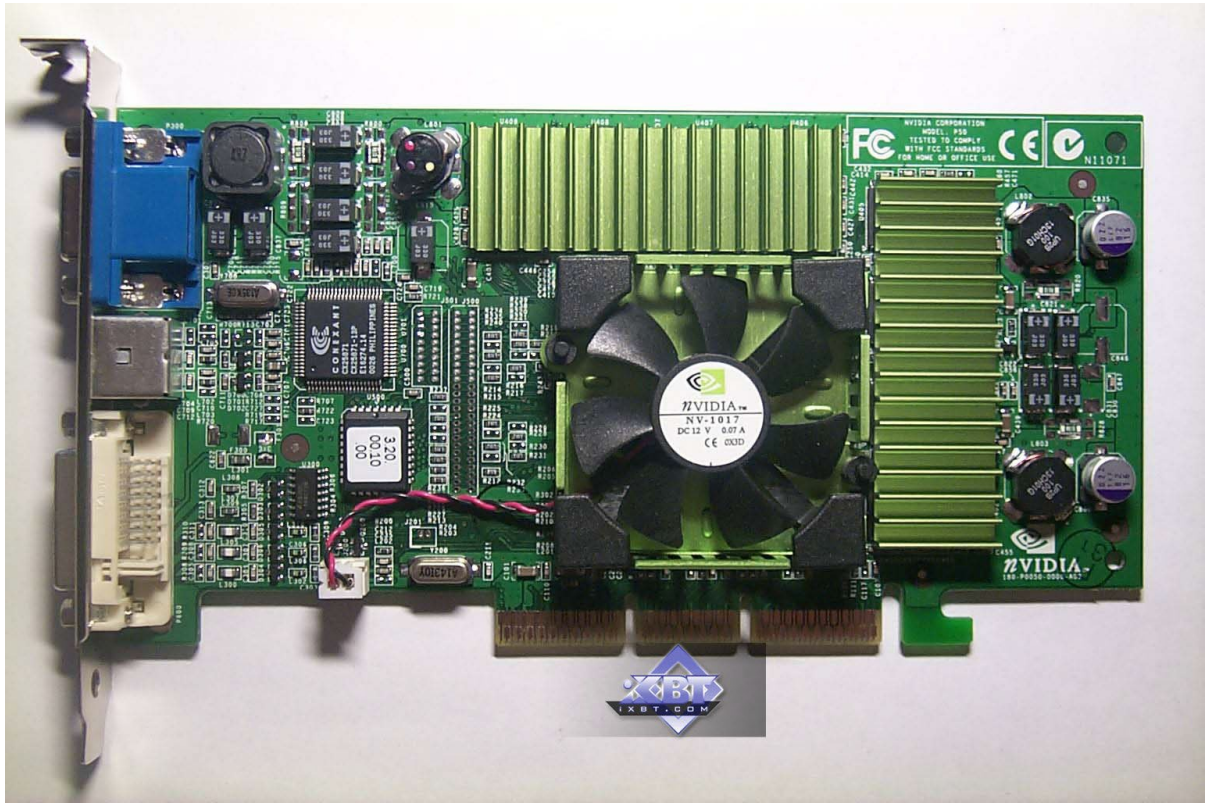


FIRST STEPS IN GPU COMPUTING

GEFORCE 3 WITH PROGRAMMABLE VERTEX AND PIXEL SHADERS (2001)

The inclusion of programmable shaders in the GeForce 3 paved the way for a wide range of applications beyond video games, from scientific simulation to cryptography and deep learning.

This marked the beginning of parallel computing accessible through GPUs.



FIRST STEPS IN GPU COMPUTING

DECEIVING THE GPU USING GRAPHICAL APIs (OPENGL OR DIRECTX)

```
float saxpy (
        float2 coords : TEXCOORD0,
    uniform sampler2D textureY,
    uniform sampler2D textureX,
    uniform float alpha ) : COLOR
{
        float result;
        float yval=y_old[i];
        float y = tex2D(textureY,coords);
        float xval=x[i];
        float x = tex2D(textureX,coords);
        y_new[i]=yval+alpha*xval;
        result = y + alpha * x;
        return result;
}
```

Y = Y + alpha * X

FIRST STEPS IN GPU COMPUTING

LIMITATIONS THAT HINDERED PROGRESS

LEARNING CURVE OF OPENGL/DIRECTX AND TRANSLATION EFFORT

NEED TO LEARN SHADING LANGUAGES (CG, GLSL)

FLOAT OR DOUBLE SUPPORT NOT GUARANTEED

LIMITATIONS IN MEMORY READ AND WRITE PATTERNS

LACK OF DEBUGGING TOOLS AND ERROR CONTROL

LIMITED RESOURCES: MEMORY, SPEED, FLEXIBILITY…

CUDA ARCHITECTURE

2007

CUDA (Compute Unified Device Architecture) is a parallel computing platform on GPUs developed by Nvidia

NVIDIA GEFORCE 8800 GTX (2007)

CUDA ARCHITECTURE

ECOSYSTEM

OWN HARDWARE ARCHITECTURE

SPECIALIZED GPU DRIVER

FLEXIBLE PROGRAMMING LANGUAGE (INITIALLY BASED ON C++)

COMPILER AND DEVELOPMENT AND DEBUGGING ENVIRONMENT

DOCUMENTATION, TUTORIALS, GRANTS

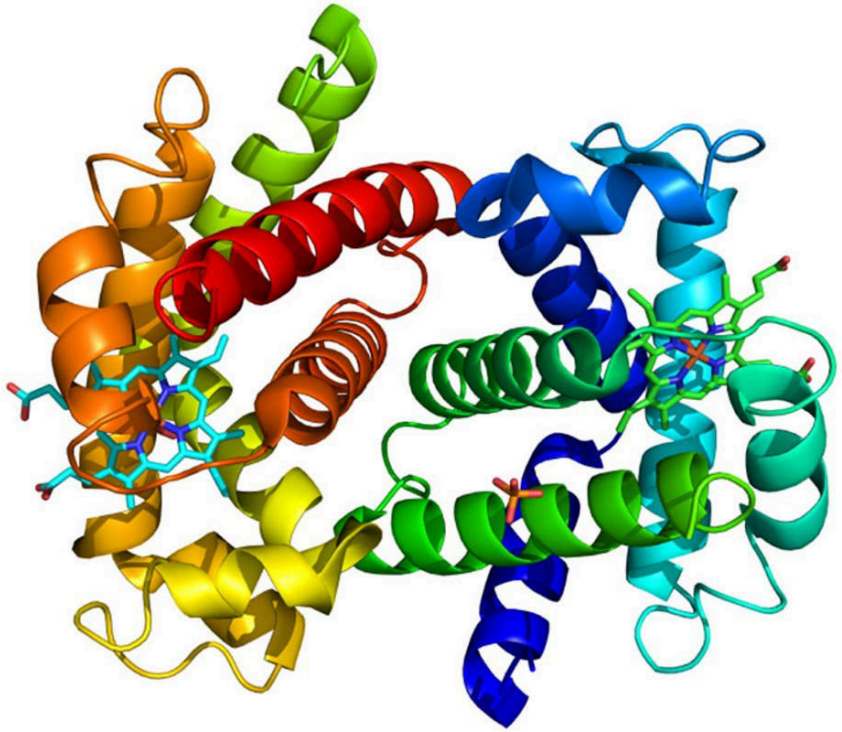
INTRODUCTION TO CUDA

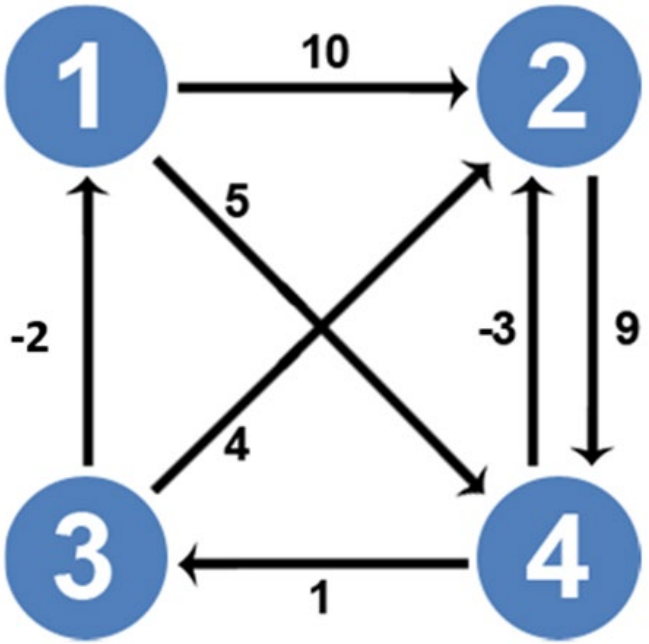
WHAT IS CUDA?

NVIDIA RTX 4080 (2022)

WHAT IS CUDA?

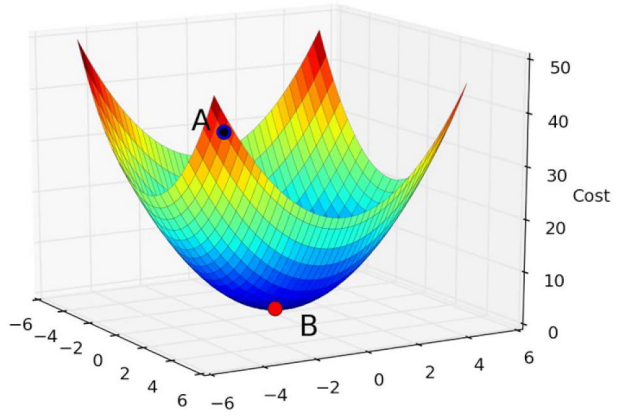
PROTEIN FOLDING, FLOYD WARSHALL ALGORITHM

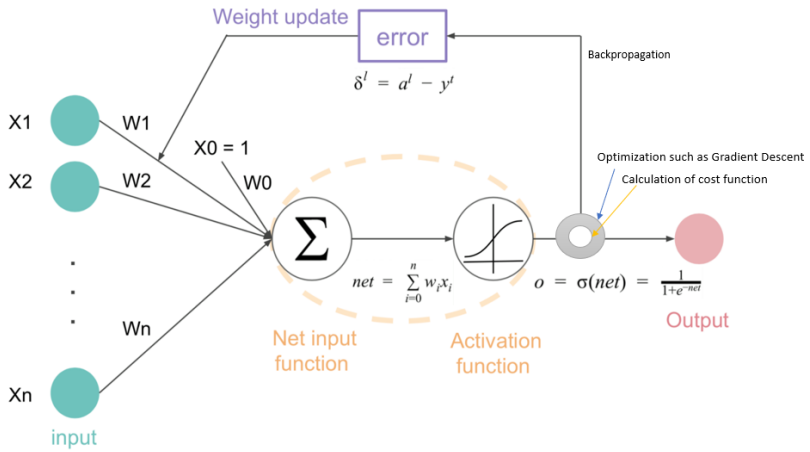




WHAT IS CUDA?

GRADIENT DESCENT AND BACKPROPAGATION





Machine Learning

Chat GPT
Whisper
DALL-E 2

AlphaGo
AlphaFold 2
AlphaTensor

BARD
Imagen
PaLM 2

StableDifussion

Claude 2

WHAT IS CUDA?

LET THE MYTHBUSTERS EXPLAIN

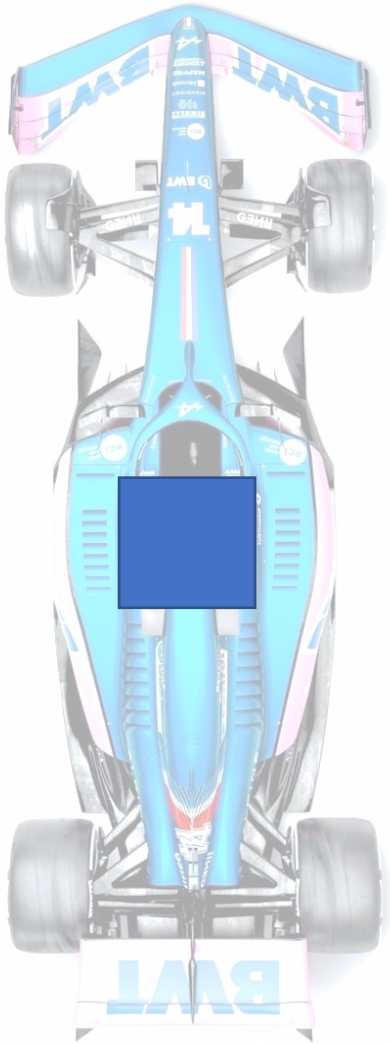
https://youtu.be/-P28LKWTzrI

WHAT IS CUDA?

ANALOGY

WHAT IS CUDA?

ANALOGY



CPU Single core

"Fast"

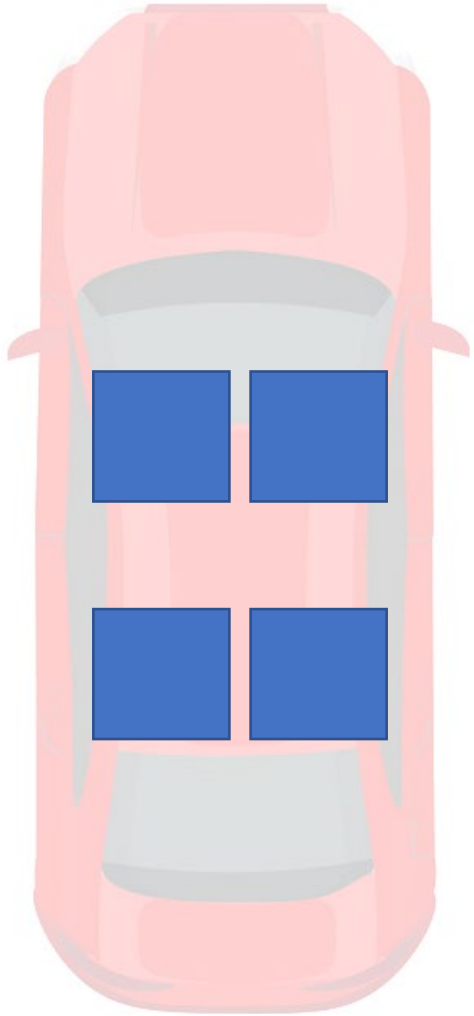
Only one seat

WHAT IS CUDA?

ANALOGY

WHAT IS CUDA?

ANALOGY



CPU Multicore

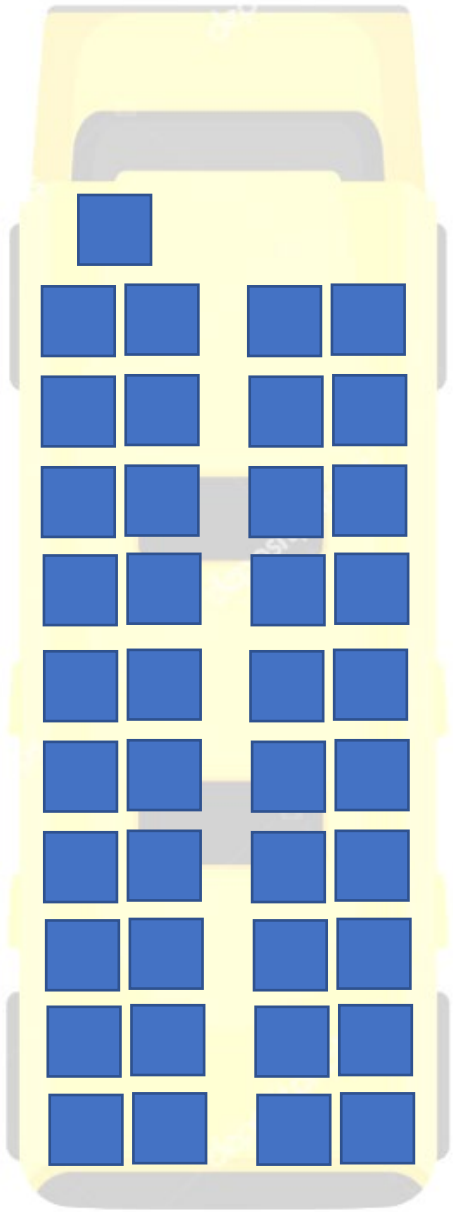
"Less fast than the F1 car"
Four seats

WHAT IS CUDA?

ANALOGY

¿QUÉ ES CUDA?

ANALOGY



GPU

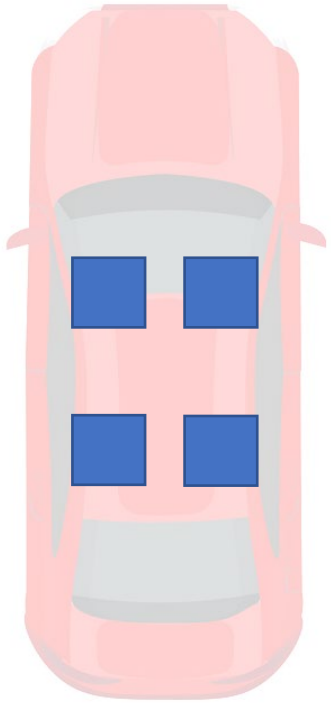
Slow
Many seats

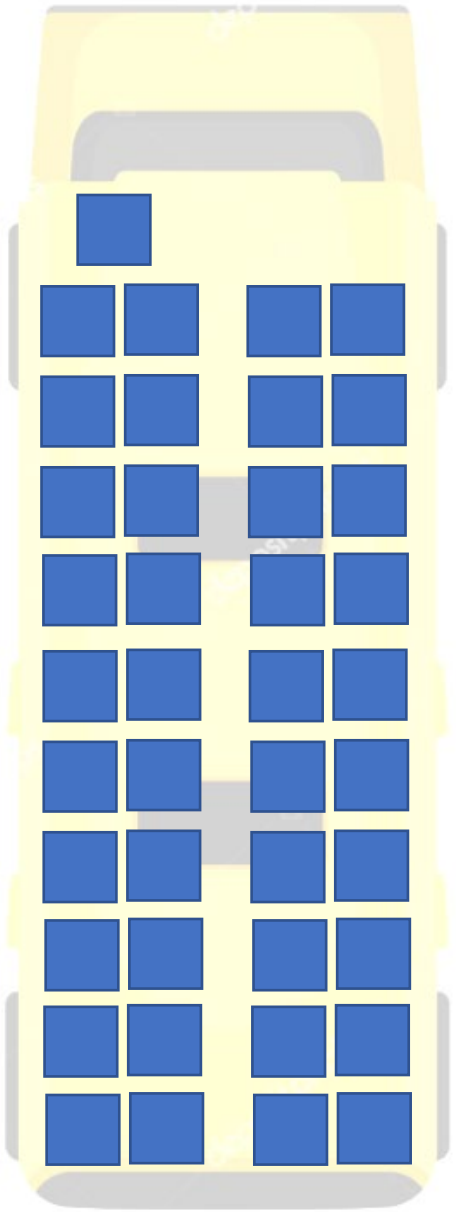
¿QUÉ ES CUDA?

ANALOGY

CPU

Fast
Multicore



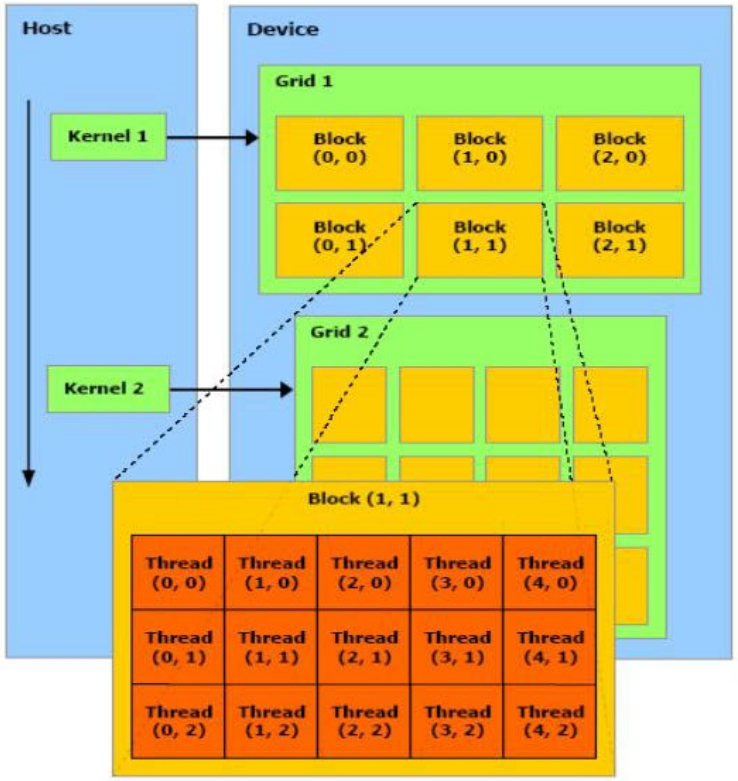


GPU

Slow
Many cores

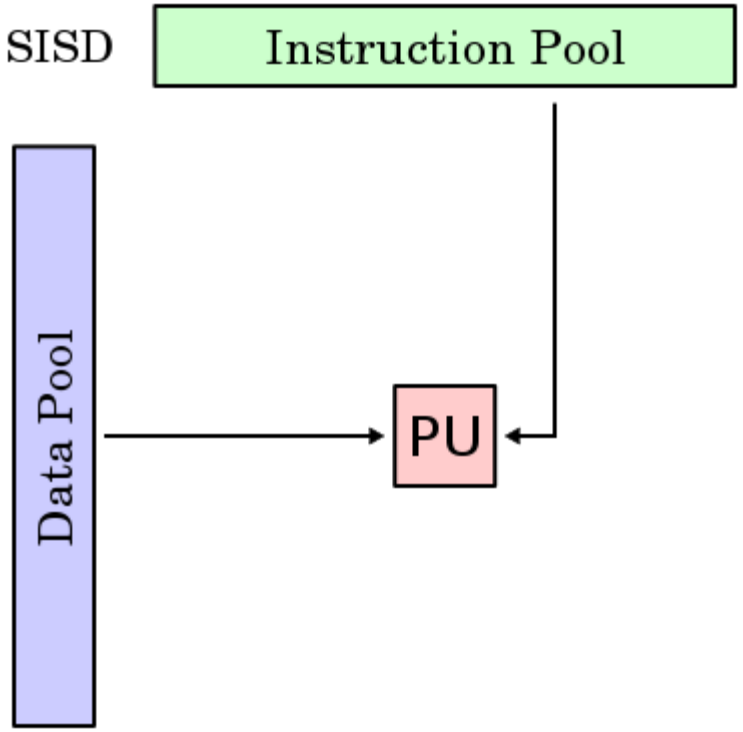
CUDA HARDWARE ARCHITECTURE

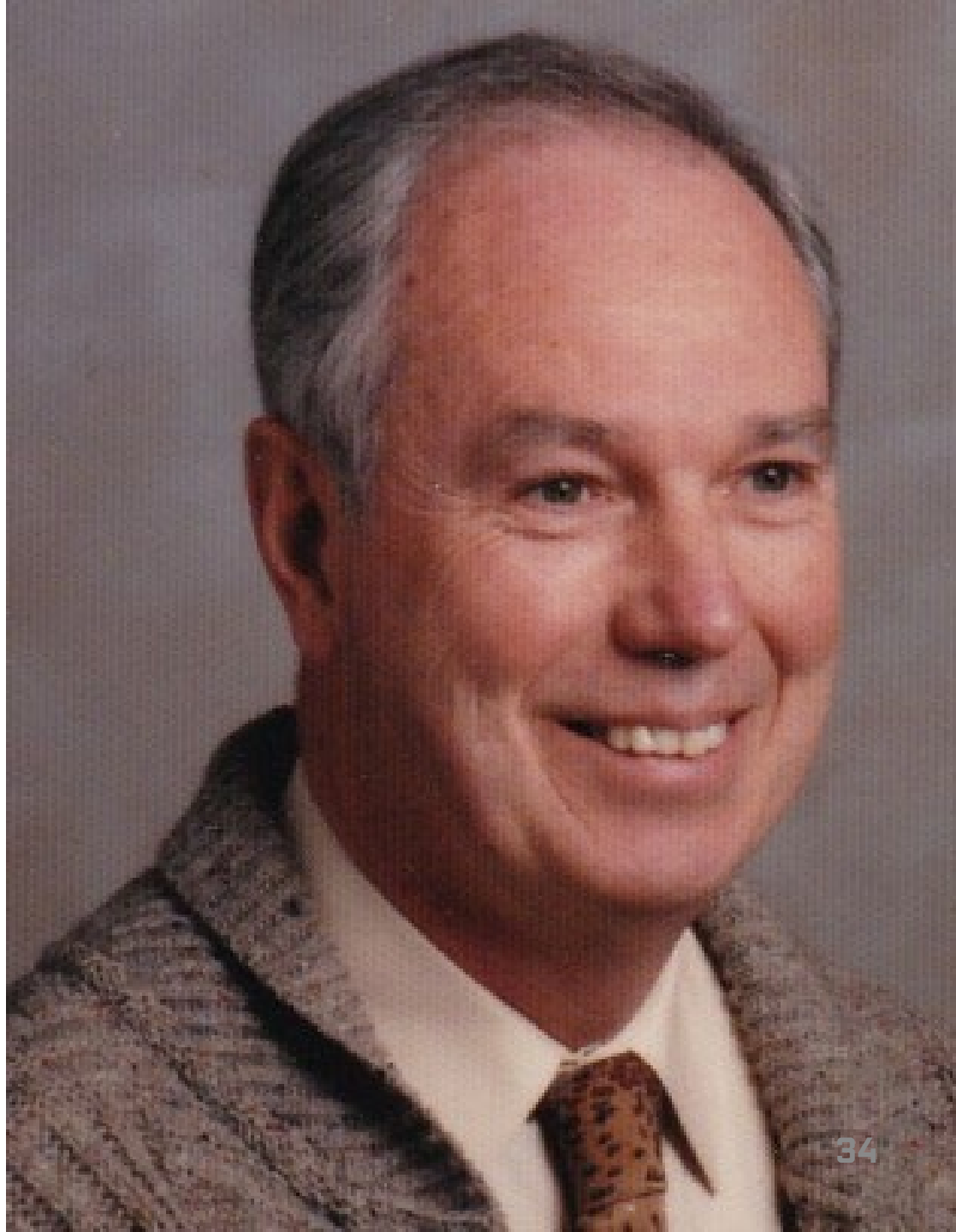
SUBTÍTULO QUE NADIE LEE



HARDWARE ARCHITECTURE

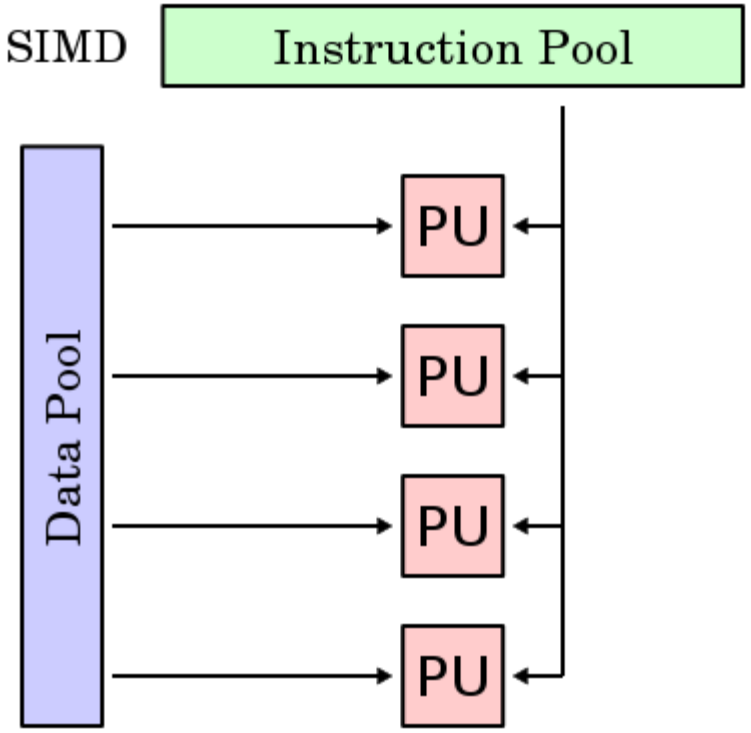
FLYNN'S TAXONOMY (SISD)

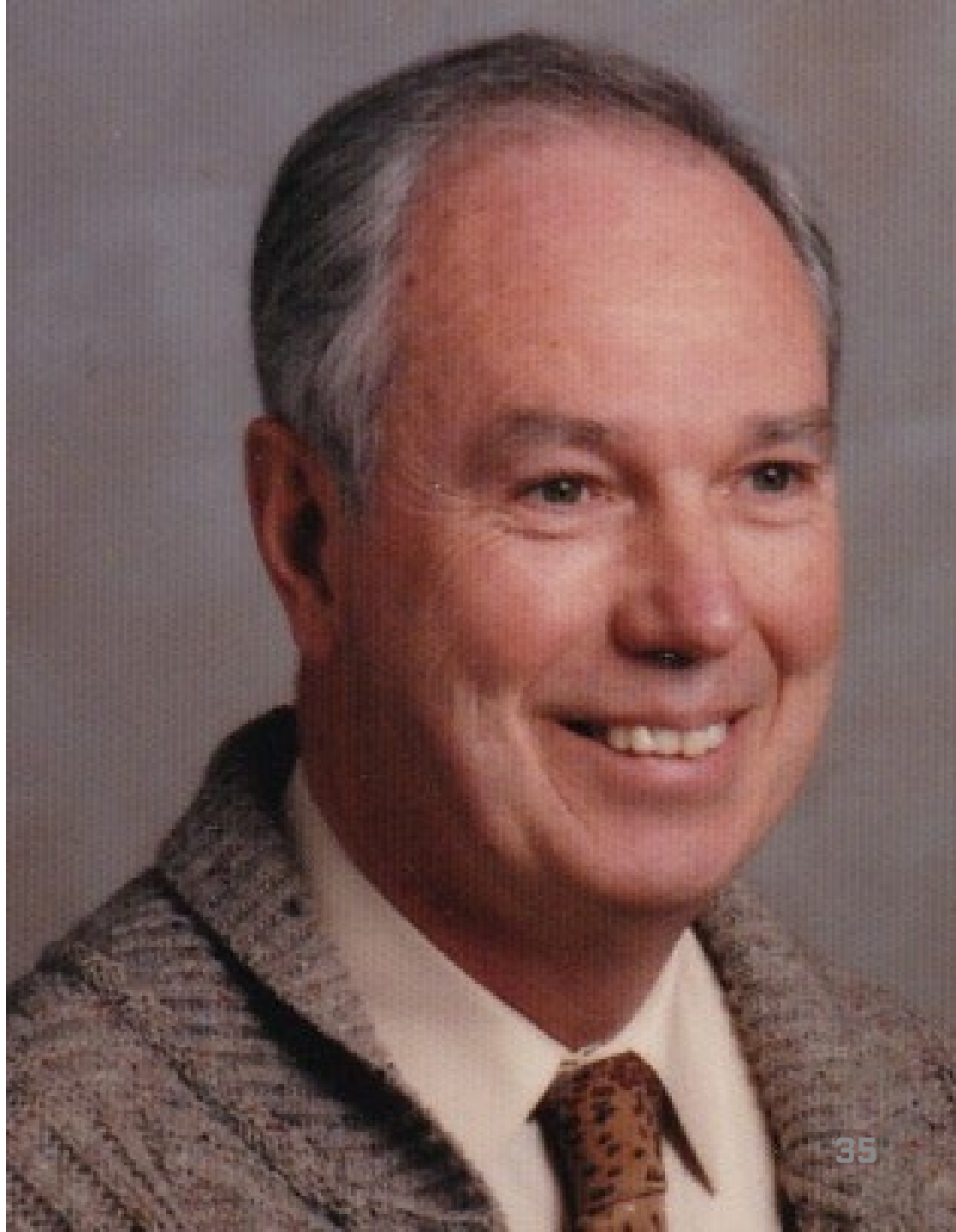




HARDWARE ARCHITECTURE

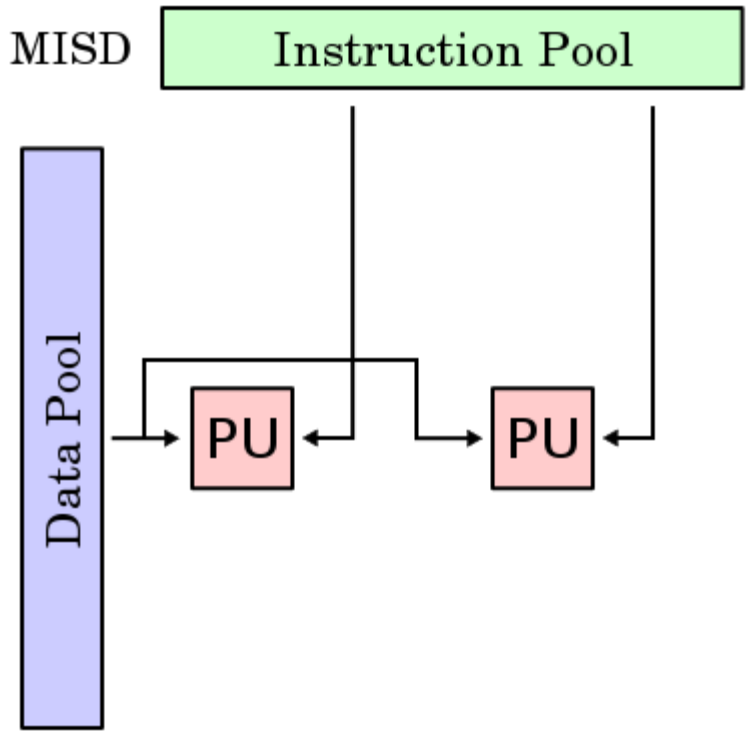
FLYNN'S TAXONOMY (SIMD)

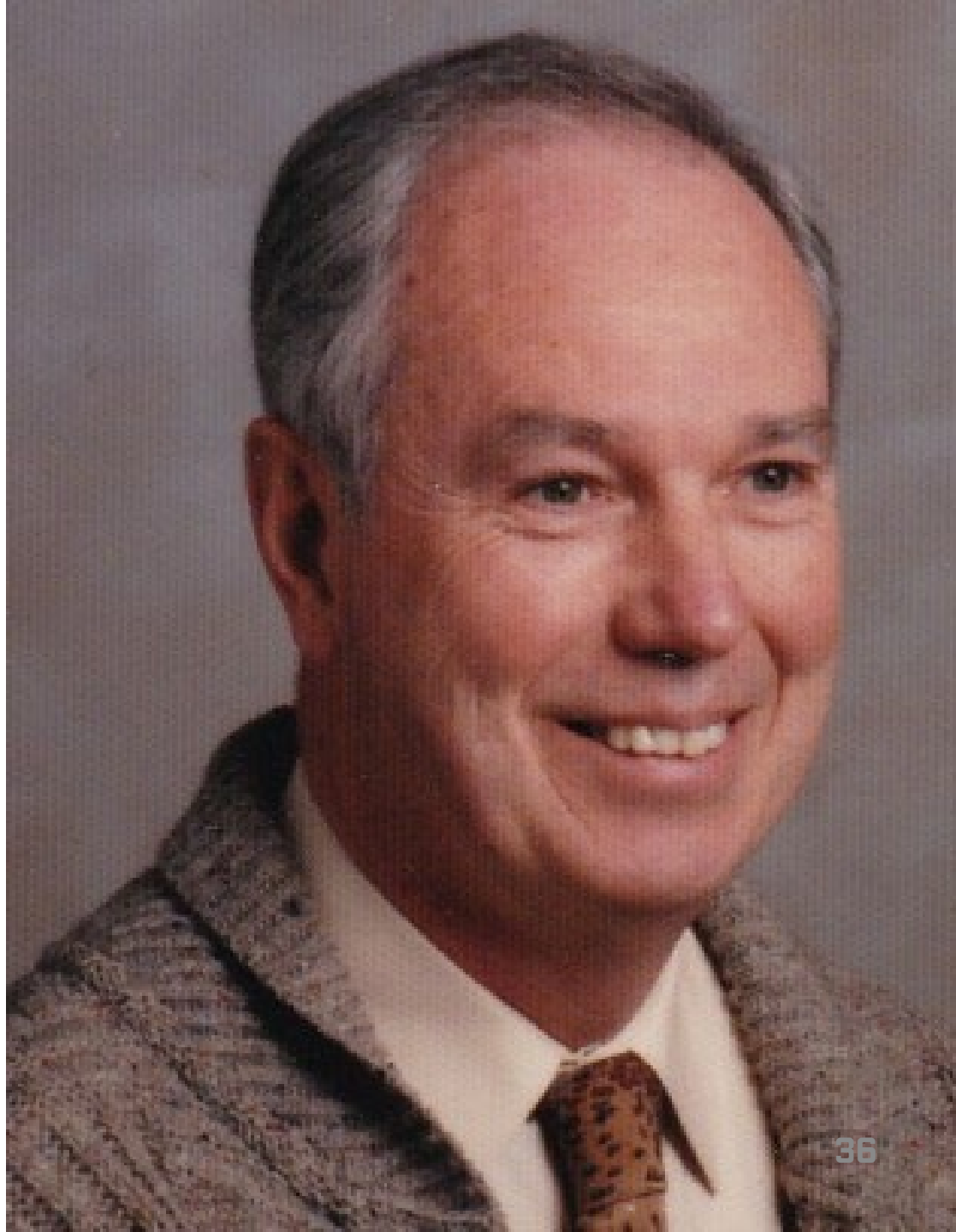




HARDWARE ARCHITECTURE

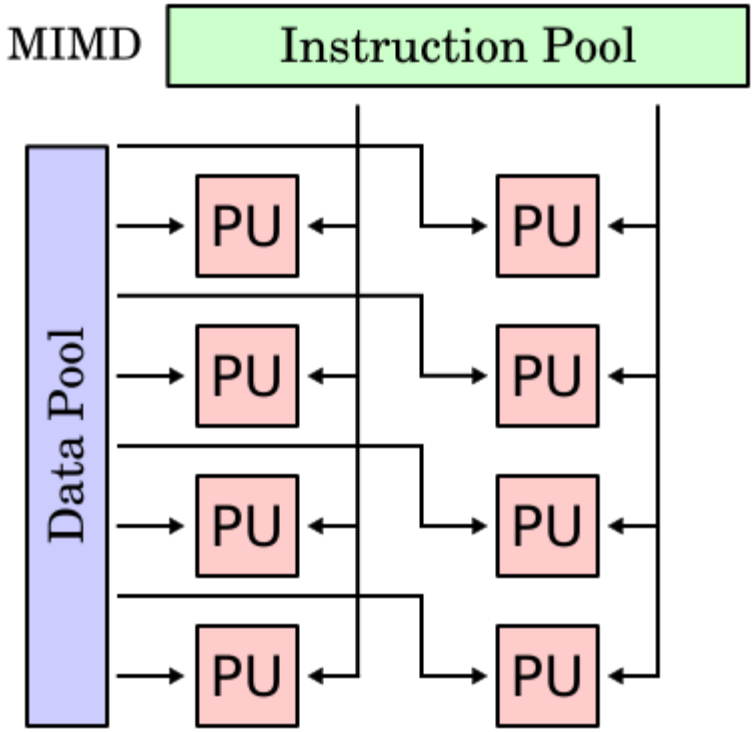
FLYNN'S TAXONOMY (MISD)

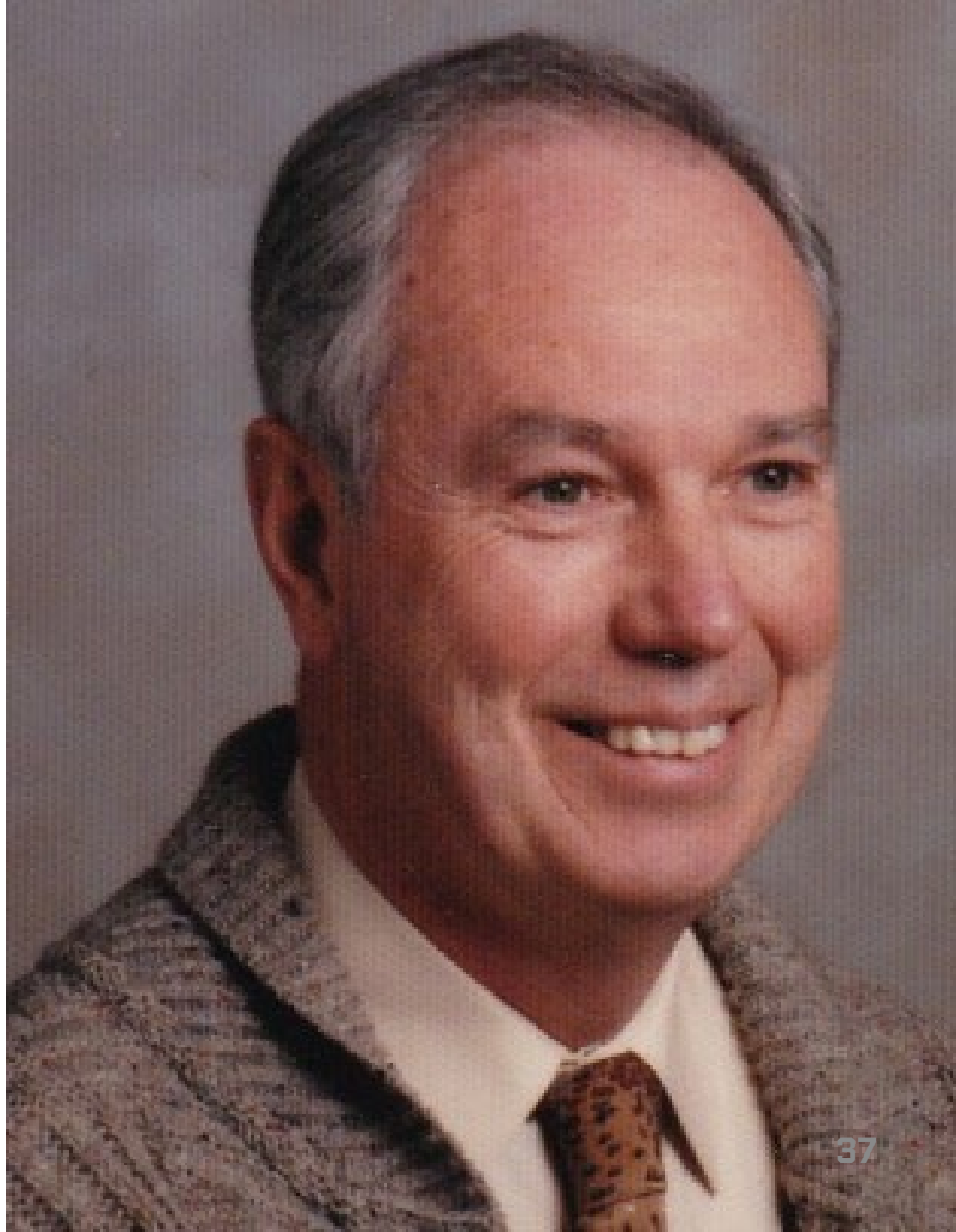




HARDWARE ARCHITECTURE

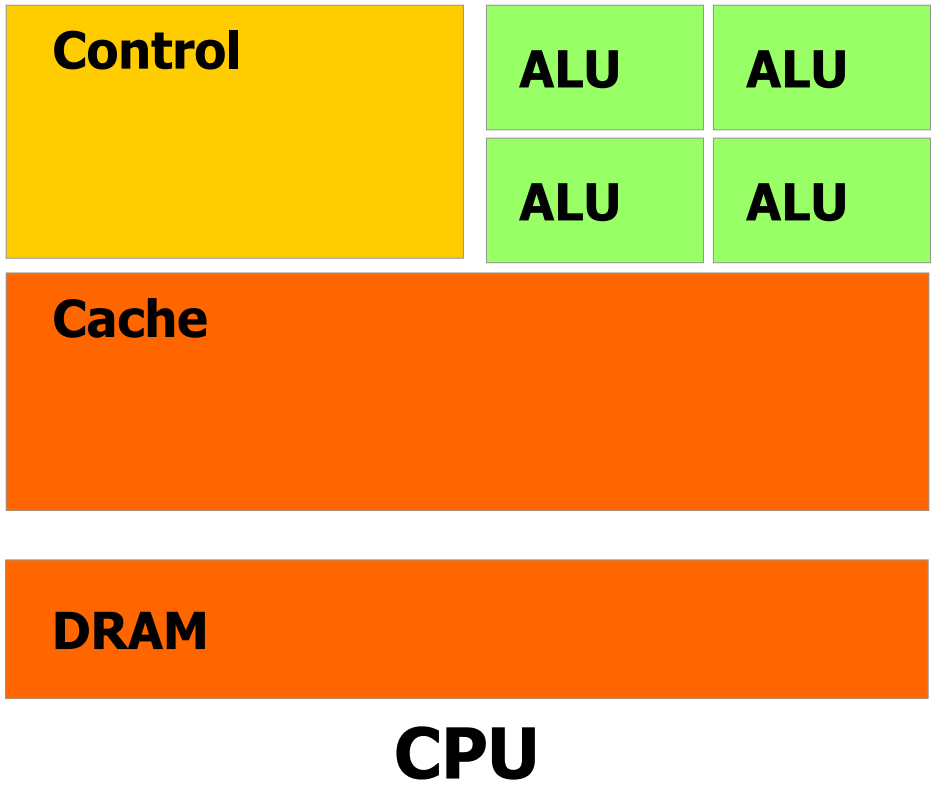
FLYNN'S TAXONOMY (MIMD)

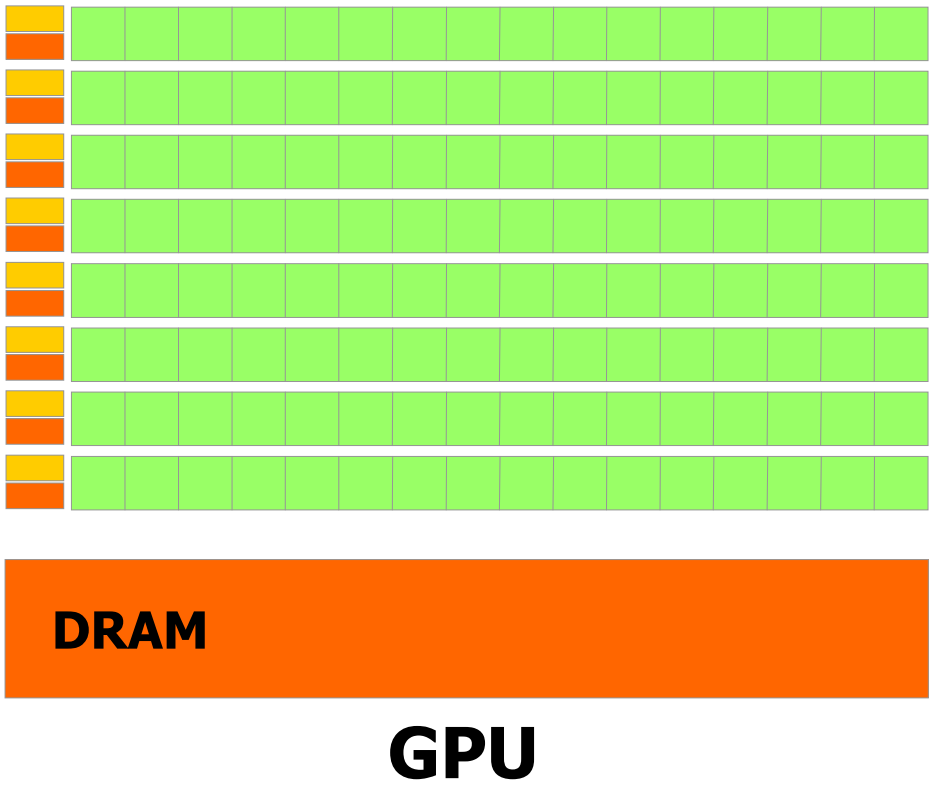




HARDWARE ARCHITECTURE

DIFFERENCES BETWEEN CPU AND GPU



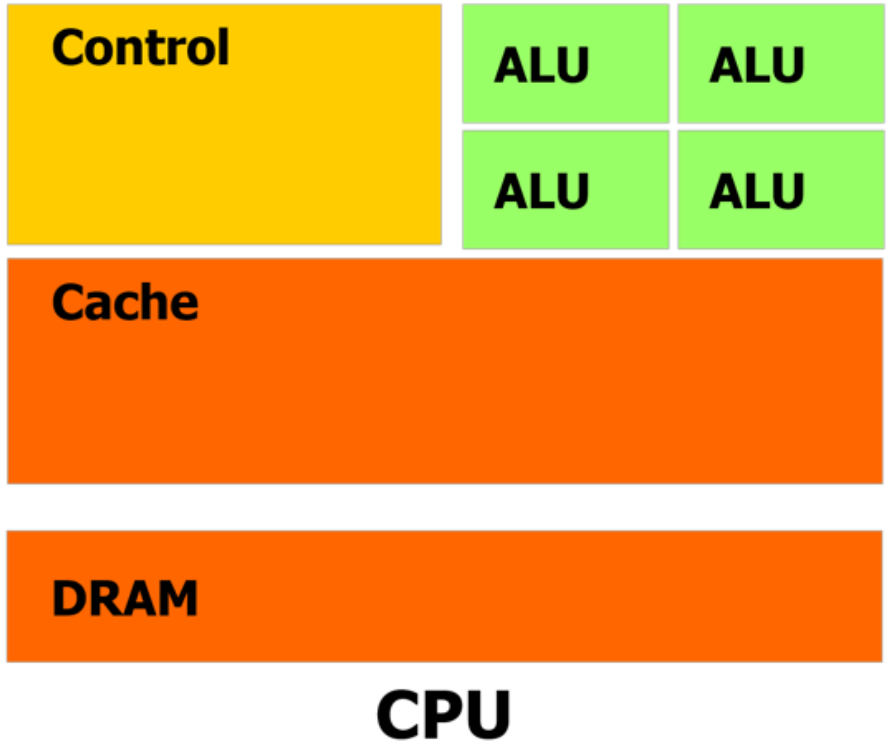


HARDWARE ARCHITECTURE

DIFFERENCES BETWEEN CPU AND GPU

CPU

- **Large caches**
  - Allow for reduced latency in memory access
- **Complex control unit**
  - Branch prediction
  - Data forwarding
- **Complex ALUs**
  - Reduce operation latency

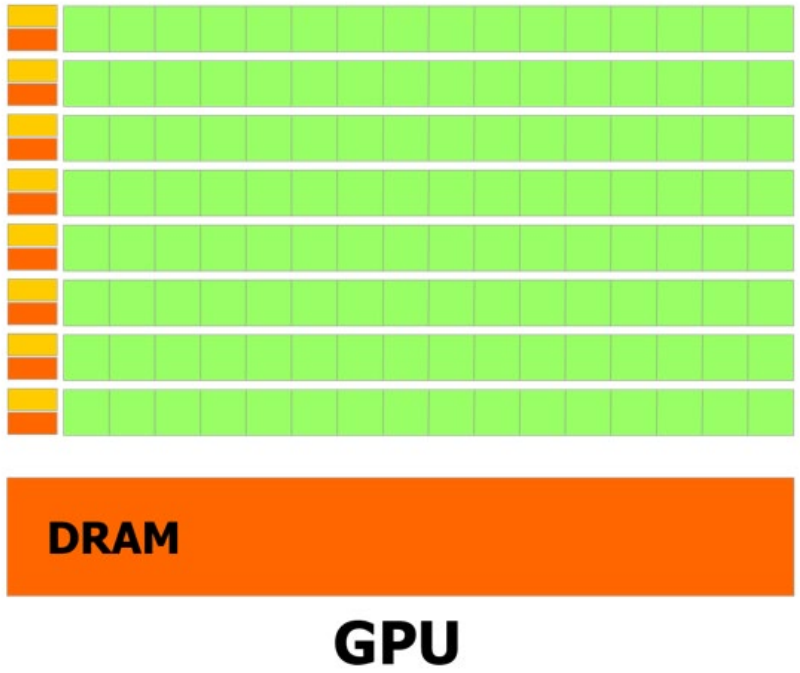


HARDWARE ARCHITECTURE

DIFFERENCES BETWEEN CPU AND GPU

GPU

- **Small caches**
  - Enhance memory bandwidth.
- **Simple control unit**
  - No Branch prediction
  - No Data forwarding
- **Many simple ALUs**
  - Highly segmented to increase memory bandwidth
- **Requires a very high number of threads to hide latencies**



HARDWARE ARCHITECTURE

DIFFERENCES BETWEEN CPU AND GPU

- **Threads on the GPU are very lightweight.**
  - Very little time is needed to create/destroy threads.
- **The GPU requires many threads to be efficient.**
  - A multi-core CPU only needs a few threads.
- **The memory access time on the GPU is high.**
  - On the CPU, the bandwidth is lower.

HARDWARE ARCHITECTURE

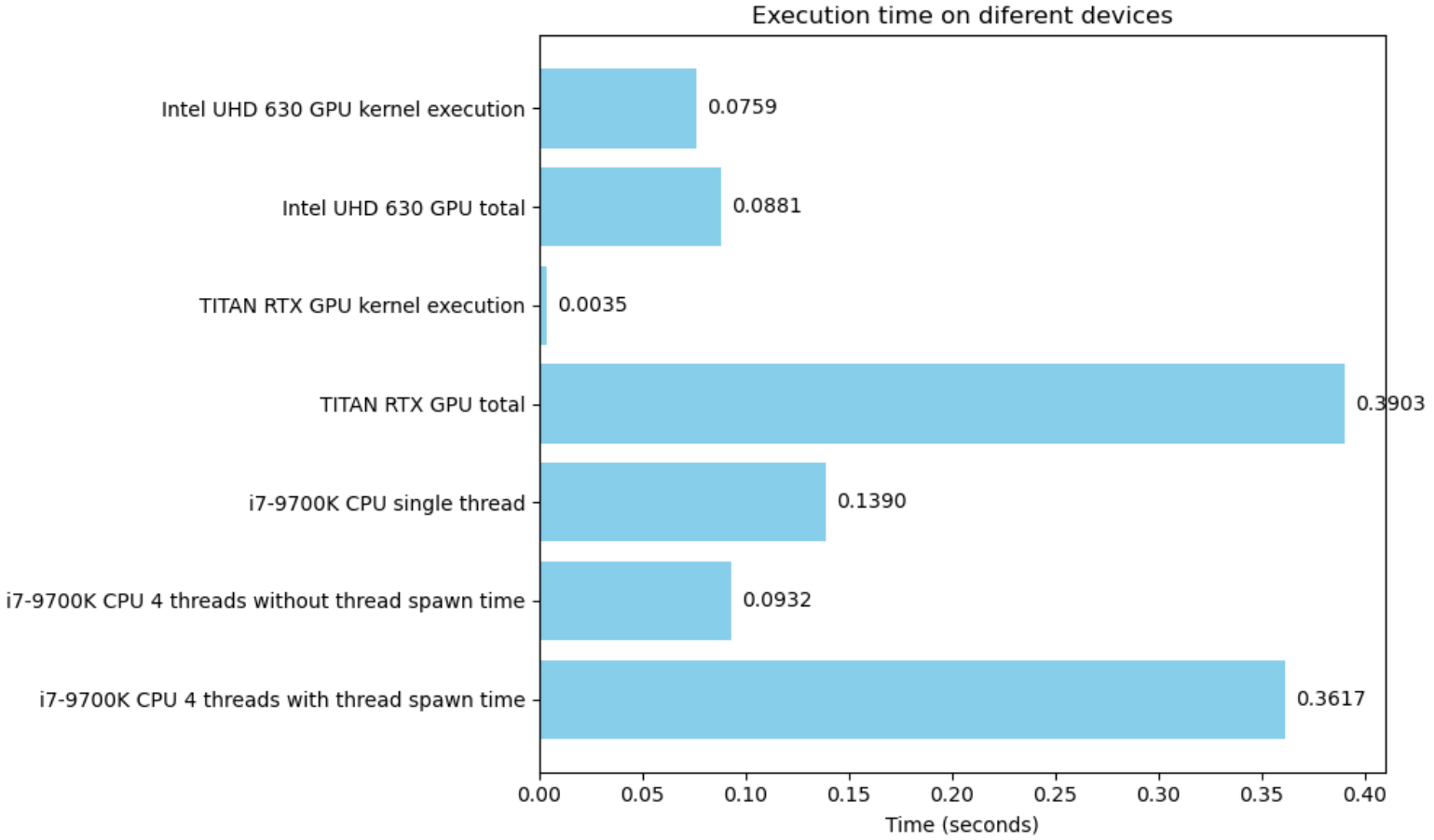
COMPARISON BETWEEN CPU AND GPU

- **Intel Core i7-12700**
  - 12 MIMD cores
  - Few registers, multi-level cache
  - 70 GB/s bandwidth to the main memory

- **NVIDIA GTX4080**
  - 9728 cores, organized into 76 SM units each with 128 cores.
  - Many registers, including level 1 and 2 caches.
  - 5 GB/s bandwidth to the HOST processor.
  - 700 GB/s graphics card memory bandwidth.

HARDWARE ARCHITECTURE

COMPARISON BETWEEN CPU AND GPU



Times required to perform a simple calculation

HARDWARE ARCHITECTURE

STREAMING MULTIPROCESSORS

INSTRUCTION CACHE/DECODER

SCHEDULER

CUDA CORES

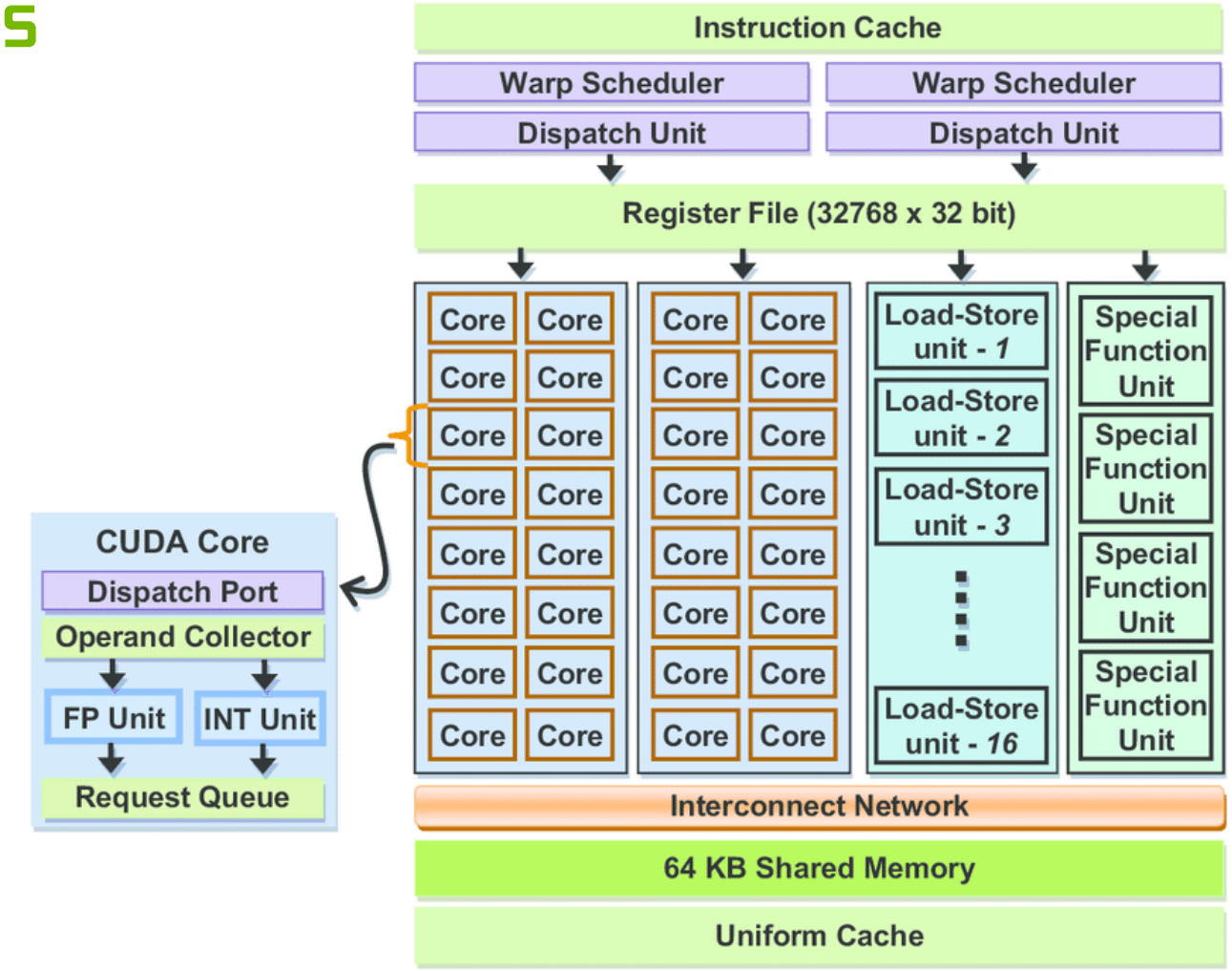
LOAD-STORE UNITS

SFU UNITS

SHARED MEMORY

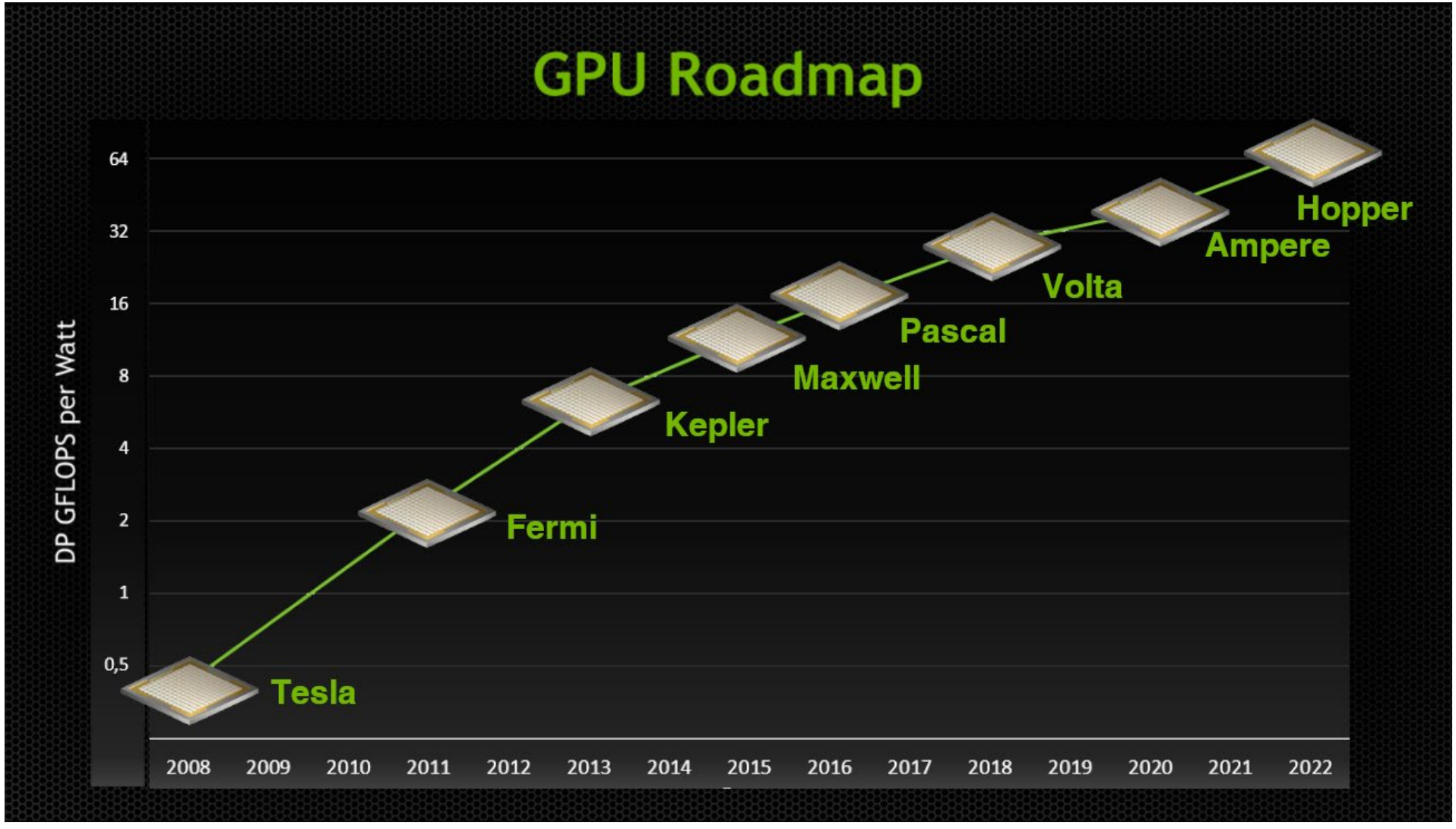
TEXTURE CACHE

REGISTERS



HARDWARE ARCHITECTURE

EVOLUTION OF MICROARCHITECTURES



HARDWARE ARCHITECTURE

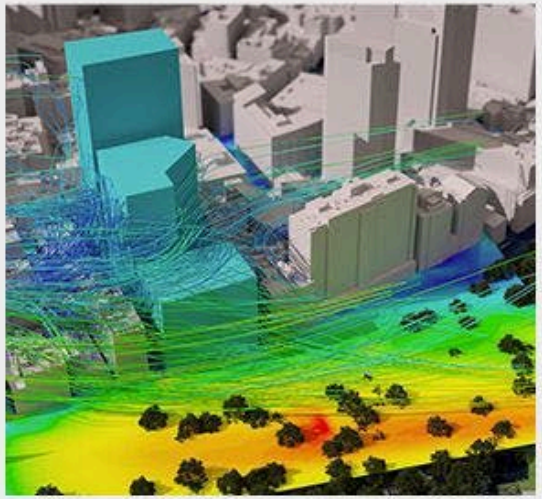
GPU TECHNOLOGY CONFERENCE 2024

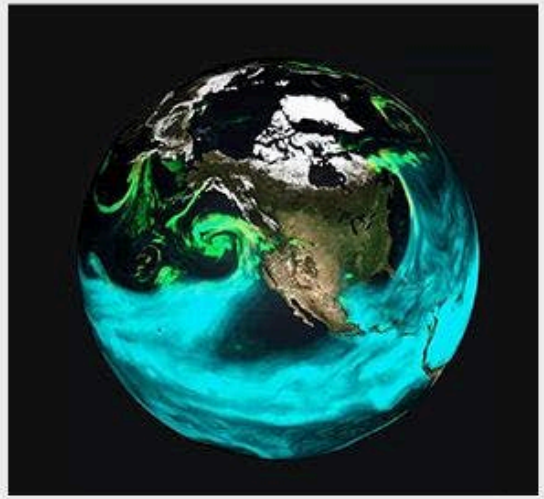
March 18, 2024

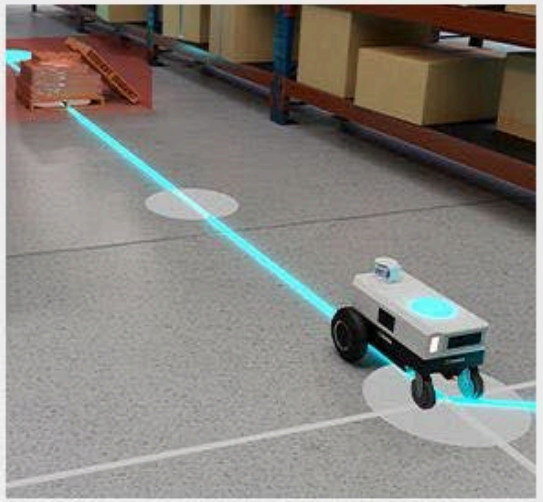
March 20-23

NVIDIA GTC

The Conference for the Era of AI and the Metaverse







HARDWARE ARCHITECTURE

CUDA

- **CUDA (Compute Unified Device Architecture)**
  - Refers to both a compiler and a set of development tools created by NVIDIA.

- **Based on C/C++** with some extensions

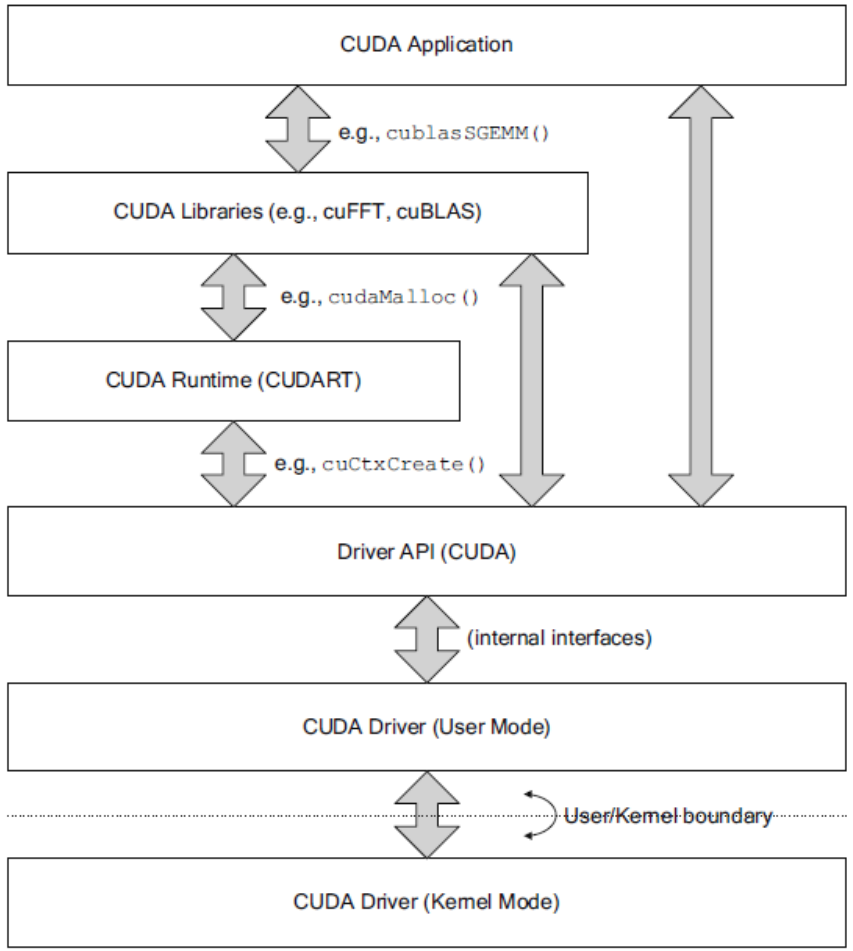
- Support for C++, Fortran. Wrappers for other languages: .NET, Python, Java...

- Large amount of **examples** and **documentation**, which reduces the learning curve for those with experience in languages like OpenMPI and MPI.

- Big user community on **NVIDIA forums**.

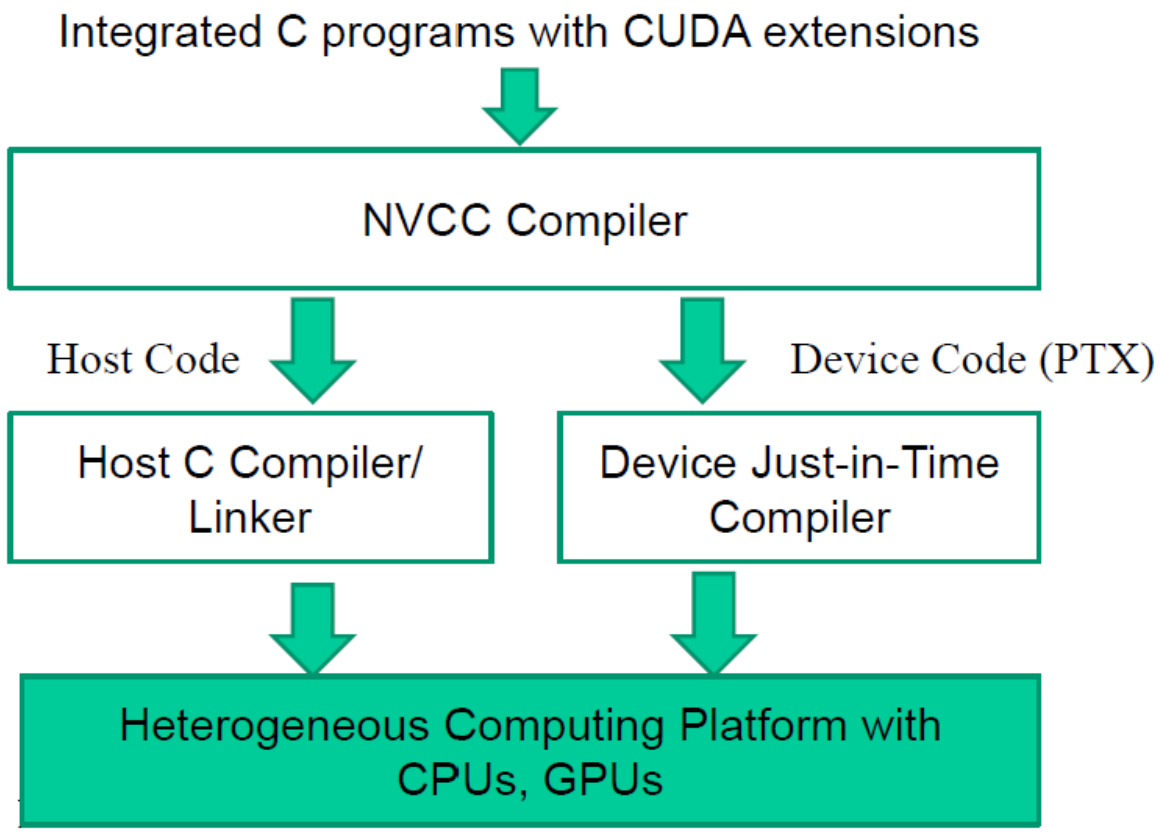
SOFTWARE ARCHITECTURE

STACK CUDA



SOFTWARE ARCHITECTURE

COMPILATION



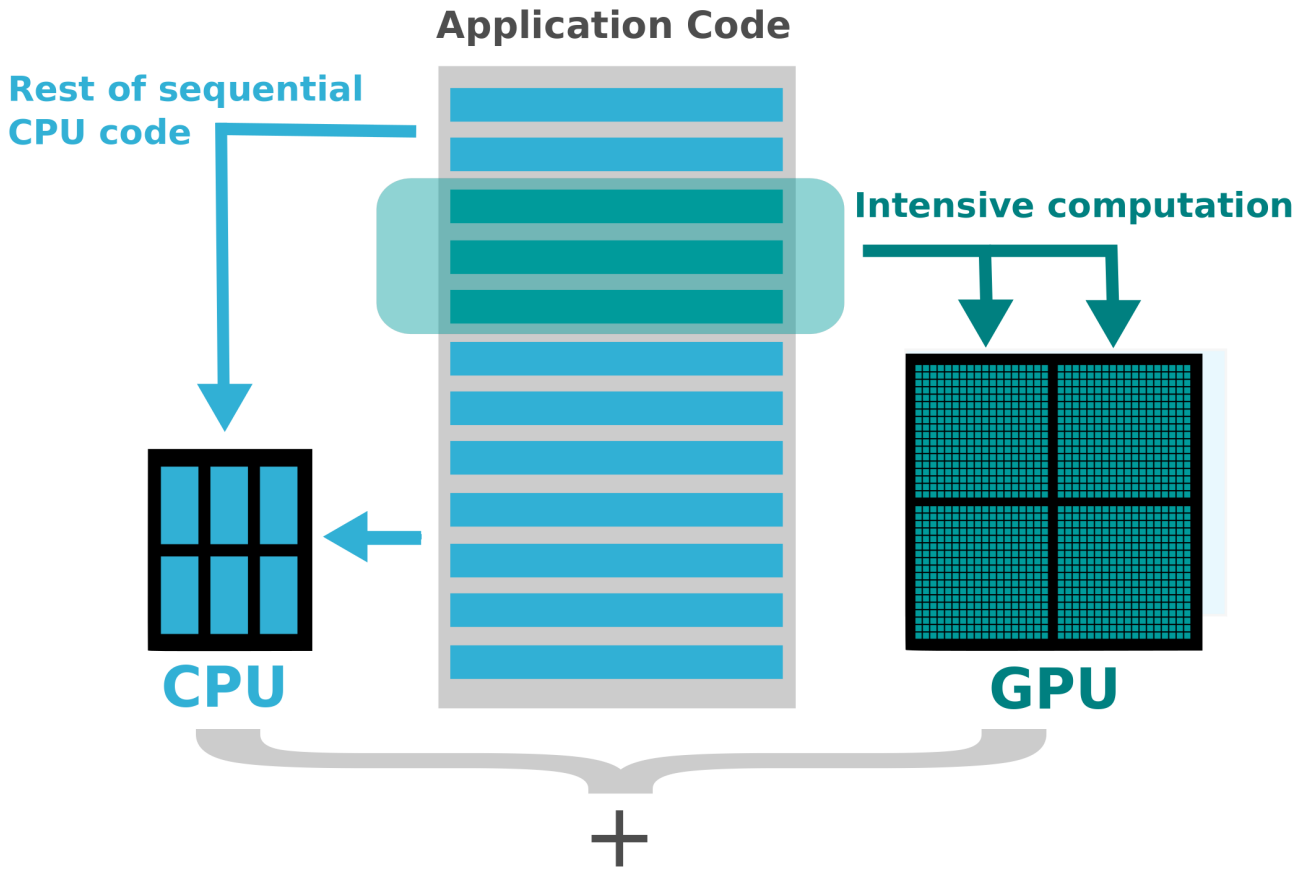
SOFTWARE ARCHITECTURE

HETEROGENEOUS COMPUTING

An approach that uses **multiple types of specialised processors**, including GPUs, for computing tasks.

Allows for the **assignment of specific tasks to the most suitable processors** for their execution, which improves performance and efficiency.

The GPU is key, as it is highly efficient in solving problems involving intensive calculations.



SOFTWARE ARCHITECTURE

BASIC CONCEPTS

THREAD: is the unit of execution on the GPU, which can perform calculations and access memory.

KERNEL: is a function that, when executed, will do so on a large number of threads and on the GPU.

BLOCK: is a grouping of threads in 1D, 2D, or 3D. Each block is executed on a single SM, but an SM can have several blocks assigned for execution.

GRID: is a way of structuring blocks, either in 1D, 2D, or 3D.

SOFTWARE ARCHITECTURE

BASIC CONCEPTS

KERNEL INVOCATION:

```
kernel_routine<<<grid_dim, block_dim>>> (args...);
```

BLOCK AND GRID SIZES DEFINED WITH DIM3:

```
dim3 block_dim (32, 32, 1); // 1024 threads in blocks of 32 x 32 x 1 (2D)
dim3 grid_dim (4, 4, 1); // 16 blocks in grid of 4 x 4 x 1 (2D)
```

SOFTWARE ARCHITECTURE

BASIC CONCEPTS

EACH THREAD EXECUTES A COPY OF THE KERNEL, AND HAS ACCESS TO THE FOLLOWING INFORMATION:

VARIABLES PASSED AS ARGUMENTS

- POINTERS TO GPU MEMORY
- SIMPLE VALUE VARIABLES (INT, FLOAT, CHAR, SIZE LIMIT)

GLOBAL CONSTANTS IN GPU MEMORY

SPECIAL VARIABLES (DIM3) TO IDENTIFY THE THREAD:

gridDim (grid size)
blockDim (block size)
blockIdx (block id) LOCAL FOR EACH BLOCK
threadIdx (thread id) LOCAL FOR EACH BLOCK

SOFTWARE ARCHITECTURE

EXECUTION FLOW

At the highest level, we find a process on the CPU (Host) that carries out the following steps:

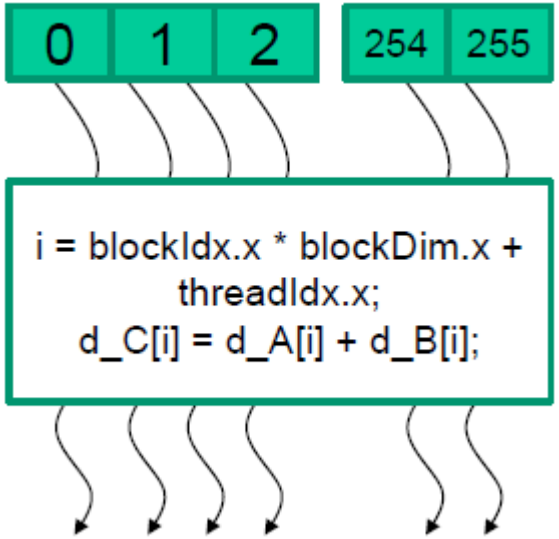
1. Initializes GPU
2. Allocates memory on the host and device parts
3. Copies data from the **host** to the **device** memory
4. Launches the execution of multiple copies of the **kernel**
5. Copies data from the **device** memory back to the **host**
6. Repeats steps 3-5 as many times as necessary
7. Releases memory and ends the execution of the master process

SOFTWARE ARCHITECTURE

EXECUTION OF A KERNEL

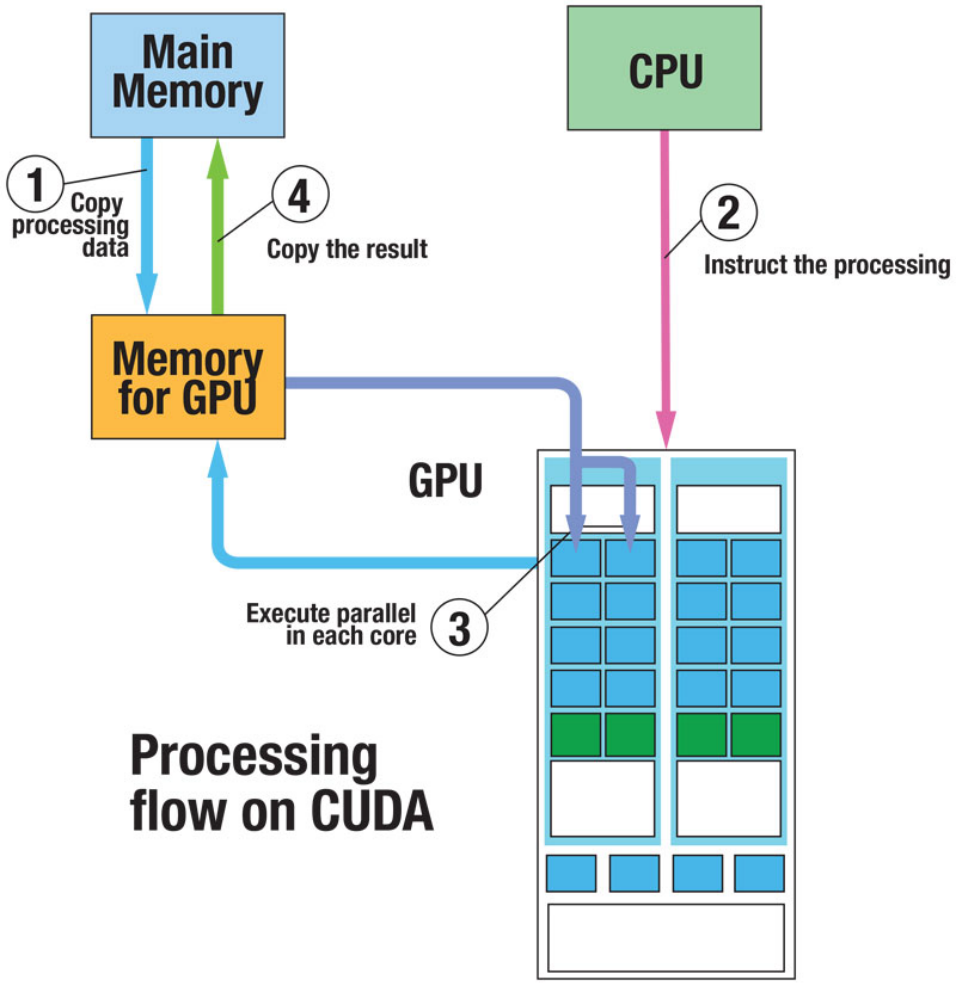
A CUDA kernel is executed over a grid of threads:

- All threads of the grid execute the same code
- Each thread has its indices and uses them to address the memory and perform its logic.



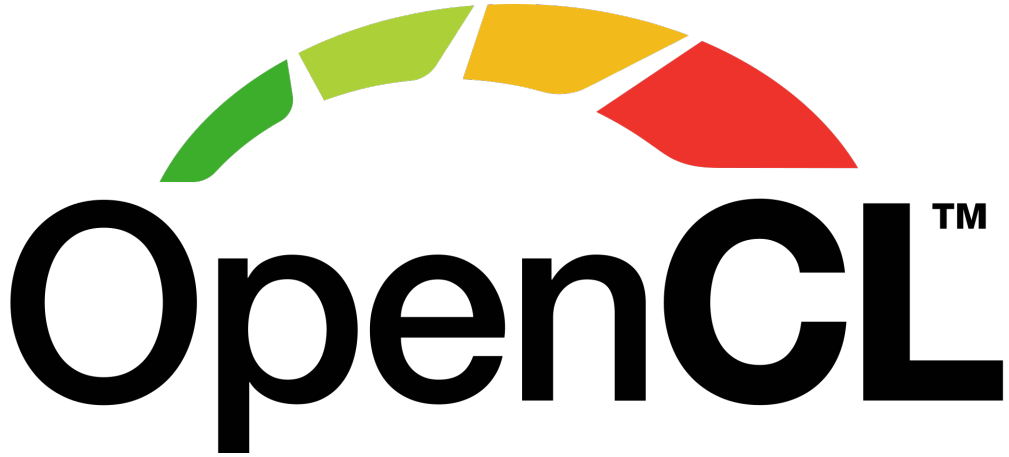
SOFTWARE ARCHITECTURE

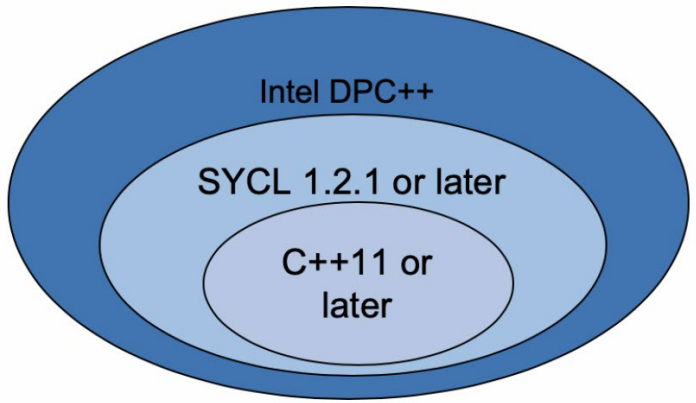
EXECUTION FLOW



ALTERNATIVES

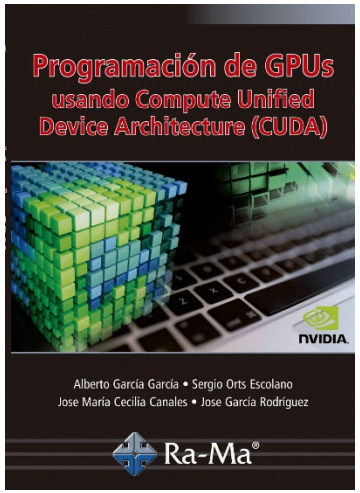
OTHER MODELS OF HETEROGENEOUS COMPUTING





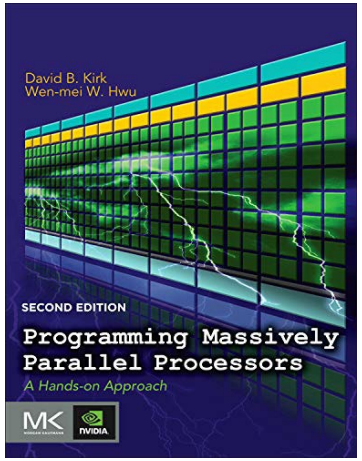
RECOMMENDED BIBLIOGRAPHY

TO LEARN A LITTLE BIT MORE



Programación de GPUs Usando Compute Unified Device Architecture (CUDA)

- Alberto García García, Sergio Orts Escolano, José Celilia Canales, José García Rodríguez

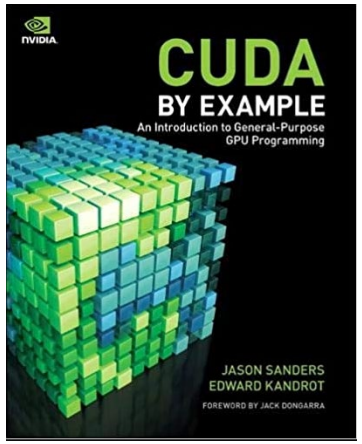


Programming Massively Parallel Processors (A Hands-on Approach)

- David B. Kirk, Wen-mei W. Hwu

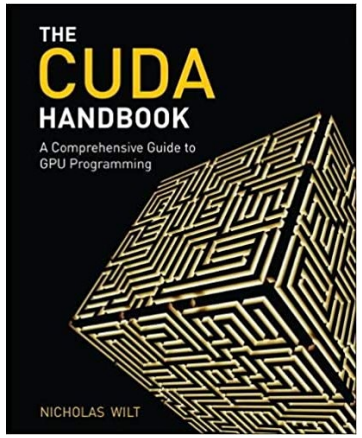
RECOMMENDED BIBLIOGRAPHY

TO LEARN A LITTLE BIT MORE



CUDA BY EXAMPLE (An Introduction to General-Purpose GPU Programming)

- Jason Sanders, Edward Kandrot



THE CUDA HANDBOOK (A Comprehensive Guide to GPU Programming)

- Nicholas Wilt