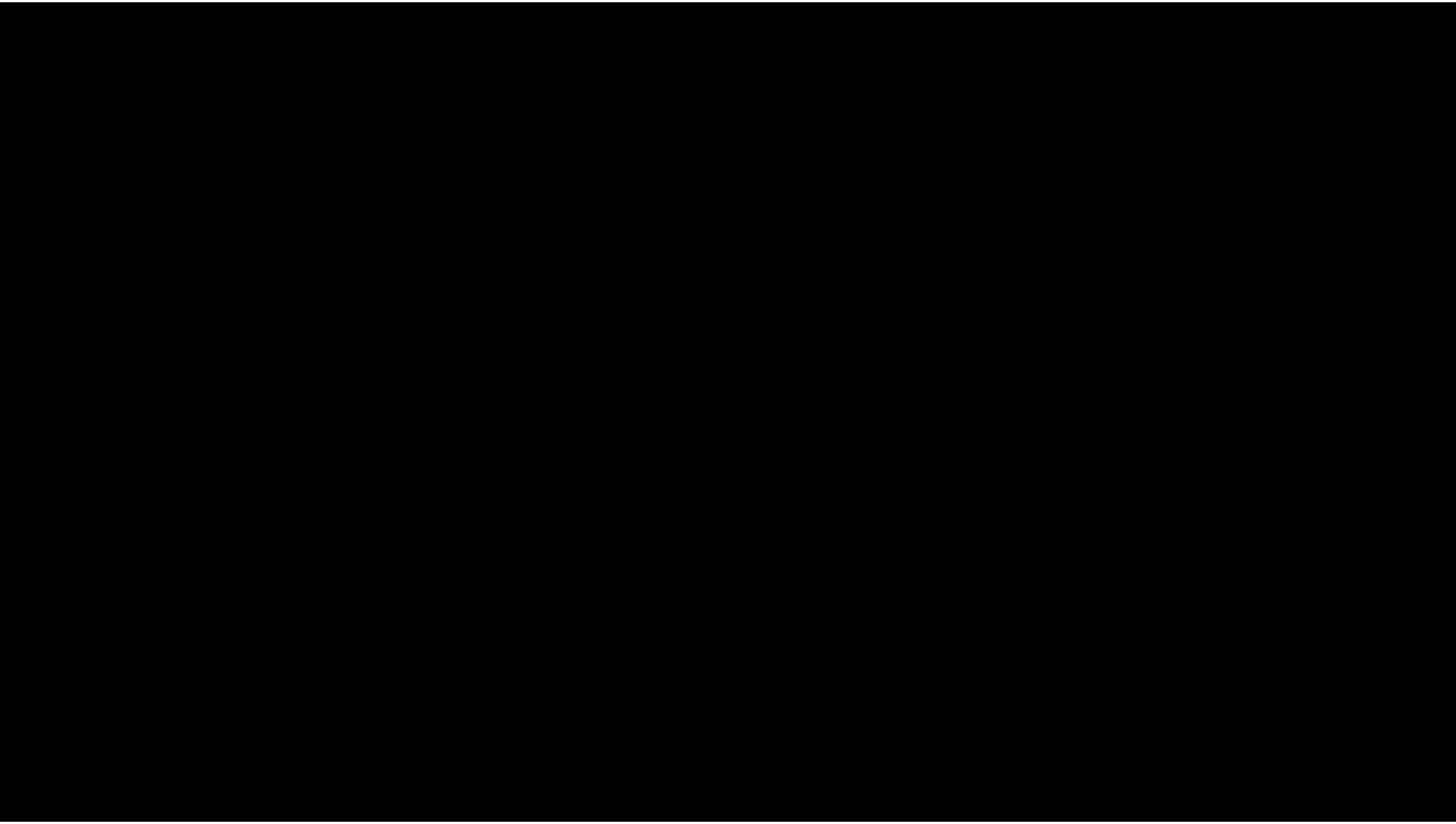
Paralelno programiranje za grafičke procesore



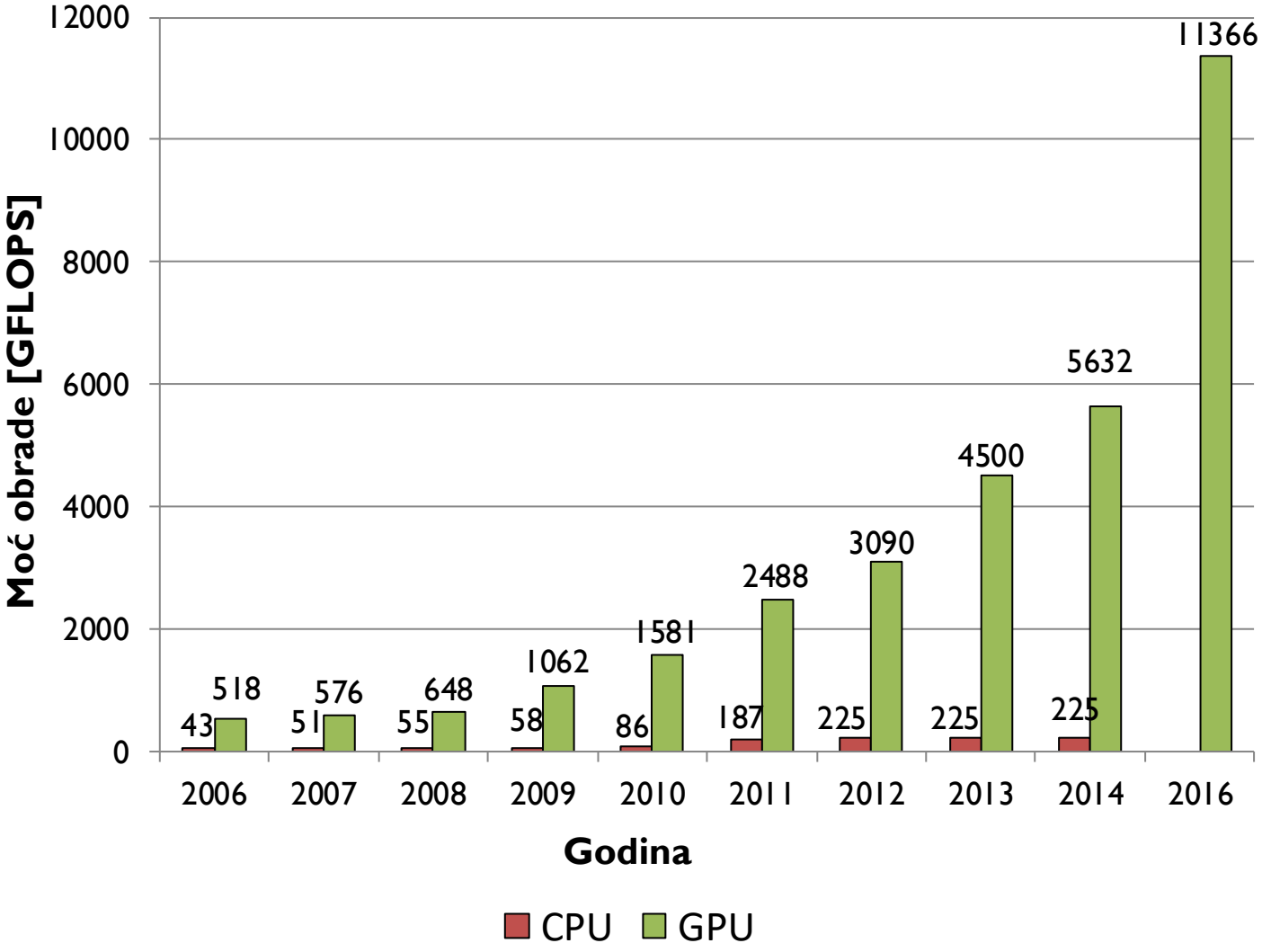
Izvor: NVidia NVision 2008, https://www.youtube.com/watch?v=WmW6SD-EHVY

GPU Computing

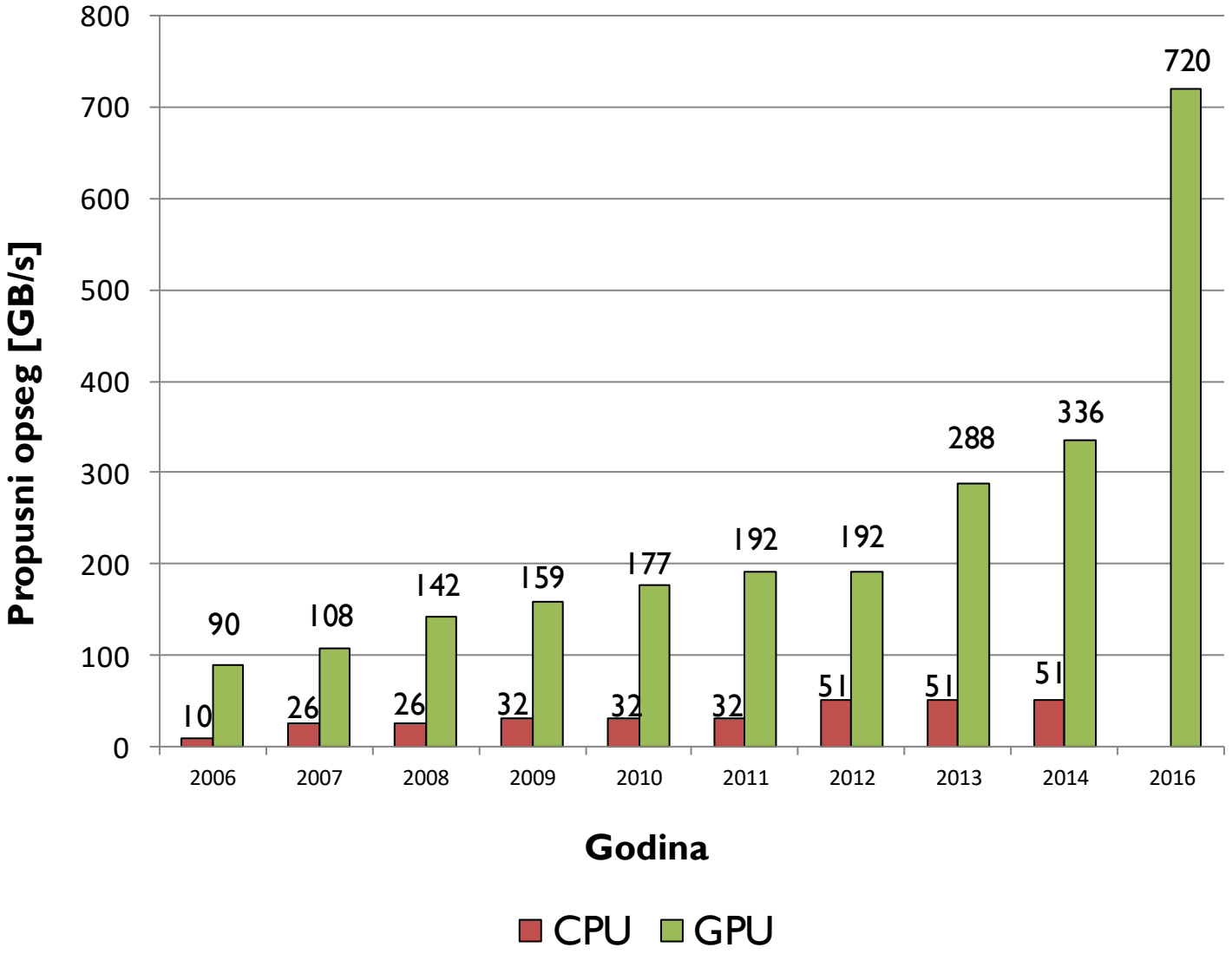
- **Grafičke procesne jedinice** (engl. *Graphics Processing Units – GPUs*) su postale standardni dodatak **centralnim procesnim jedinicama** (engl. *Central Processing Units – CPUs*) – oblast **GPU computing** ili **General-purpose computing on GPUs** (**GPGPU**)
- Evolucija arhitekture GPU od protočnog sistema sa fiksnim funkcionalnostima (engl. *fixed-function pipeline*) do **programabilne masivno-paralelne računarske platforme**

- Commodore Amiga – 1985.
- NVidia GeForce 256 – 1999.
- Industrija video igara dovela je do brze evolucije GPU
- NVidia CUDA se pojavila 2007.
- Deep learning – "big bang" se desio 2009.

Moć obrade CPU i GPU

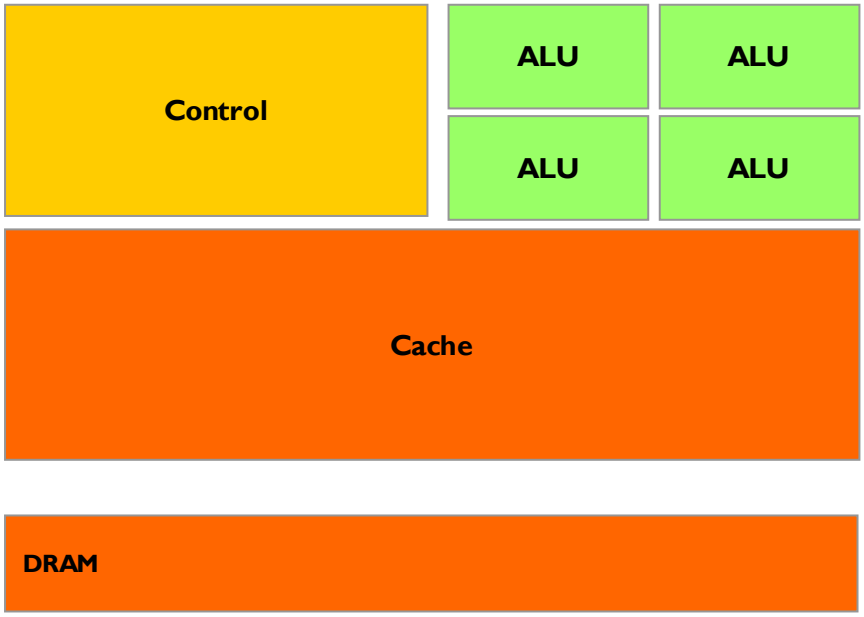


Propusni opseg CPU i GPU



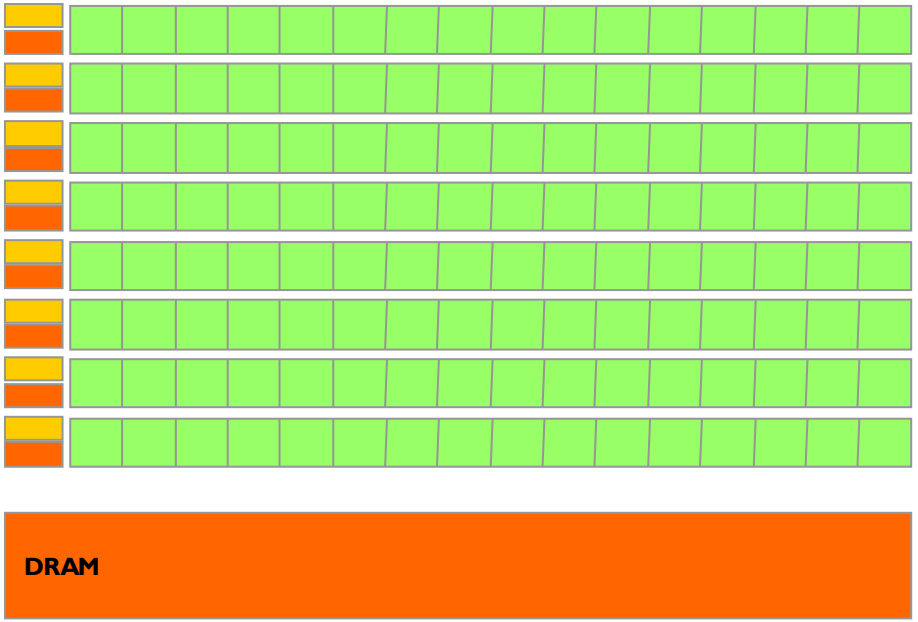
Arhitekture CPU i GPU

CPU



von Neumann, višejezgarna

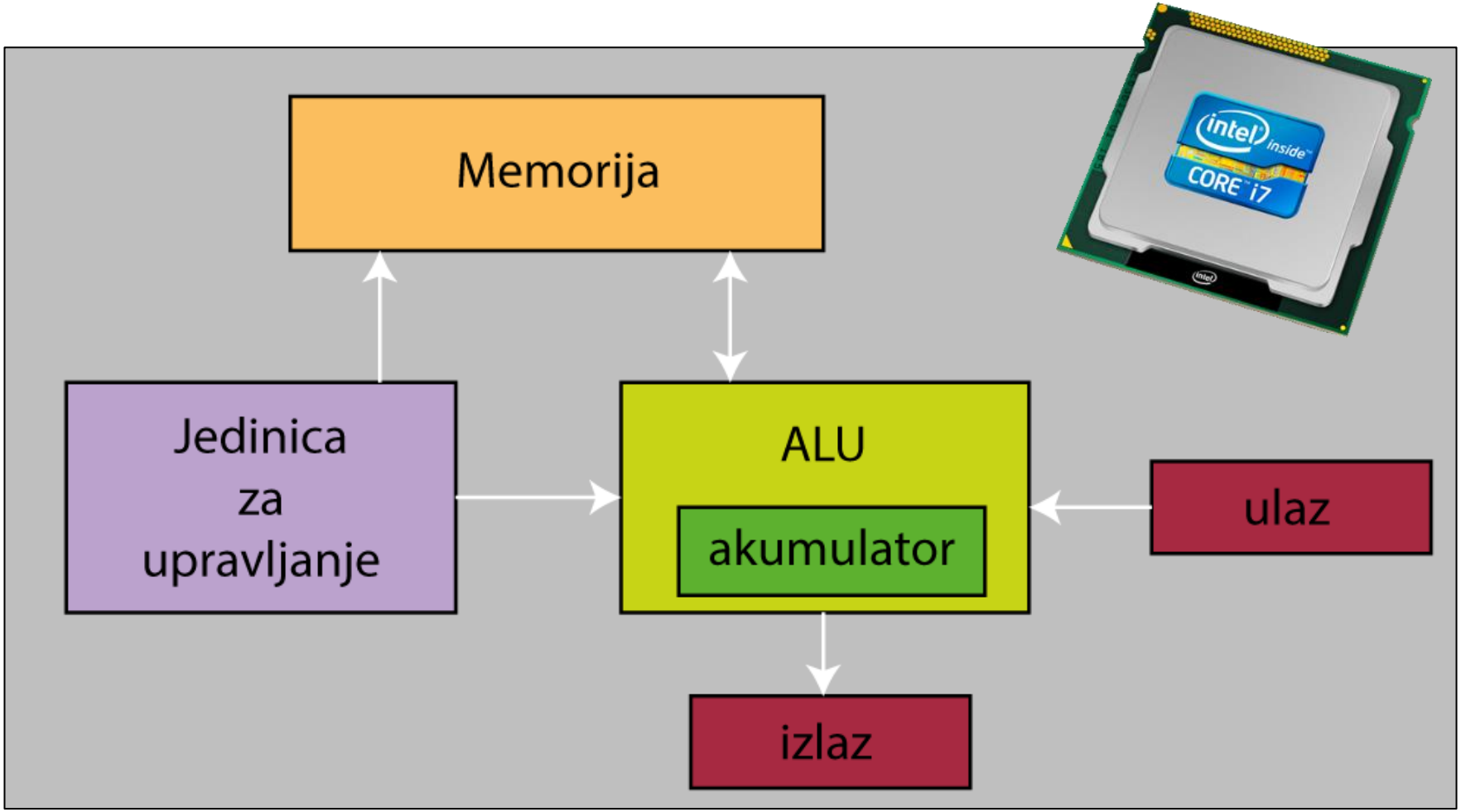
GPU



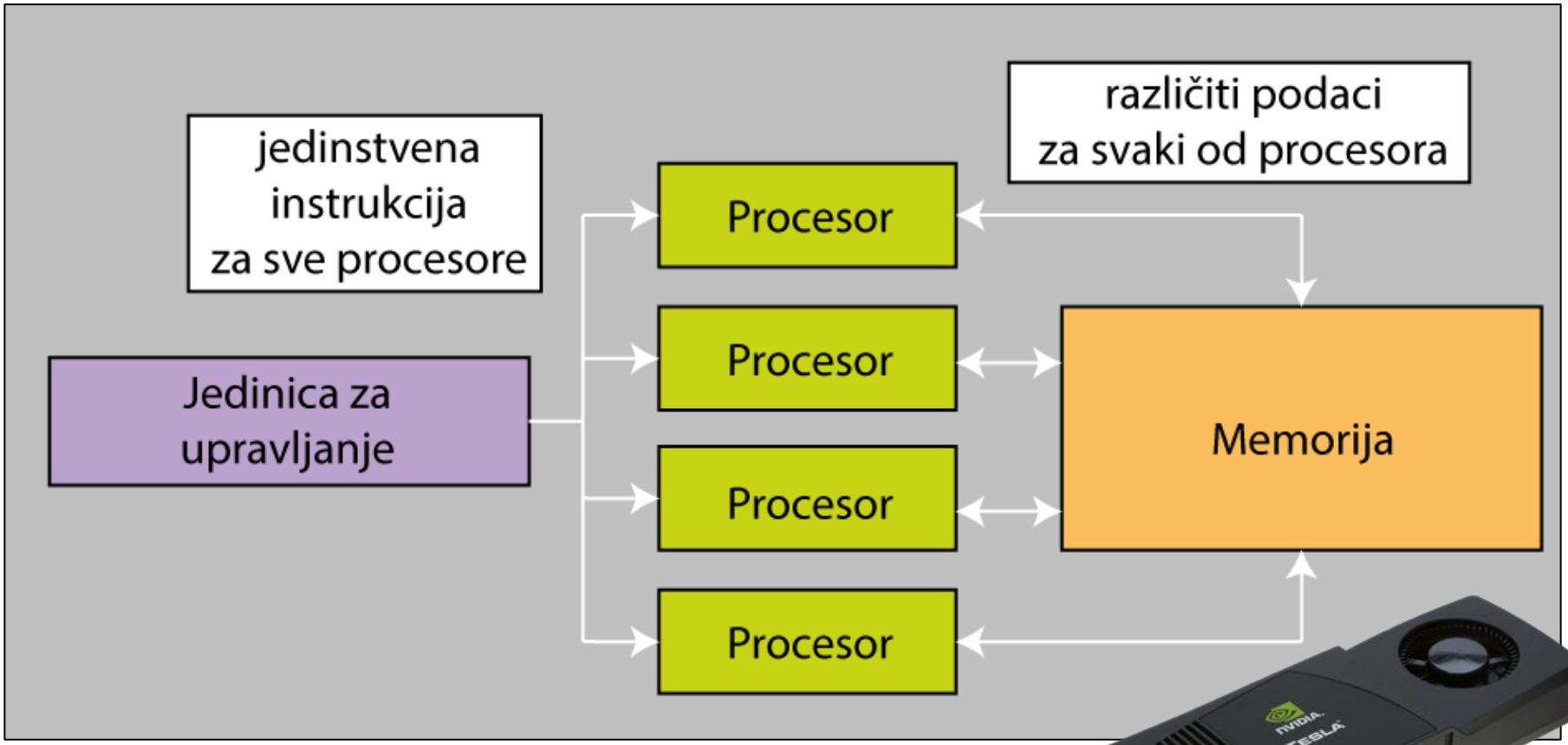
SIMD, mnogojezgarna

Izvor: https://commons.wikimedia.org/wiki/File:Cpu-gpu.svg

CPU – von Neumannova arhitektura



GPU – SIMD arhitektura



CPU: dizajn orijentisan ka latenciji

- **Von Neumannova arhitektura**, visok radni takt procesora
- **Velike keš memorije u više nivoa hijerarhije** (i do 90% tranzistora)
  - Pretvaranje dugačke latencije pristupa memoriji u kratku latenciju pristupa keš memoriji
- **Sofisticirana kontrola**
  - Predviđanje grananja (engl. *branch prediction*) za smanjenu latenciju prilikom grananja
  - Prosleđivanje podataka (engl. *data forwarding*) za redukovanu latenciju podataka
- **Moćna aritmetičko-logička jedinica** (engl. *Arithmetic Logic Unit – ALU*)
  - Kratka latencija operacija

GPU: dizajn orijentisan ka propusnog opsegu

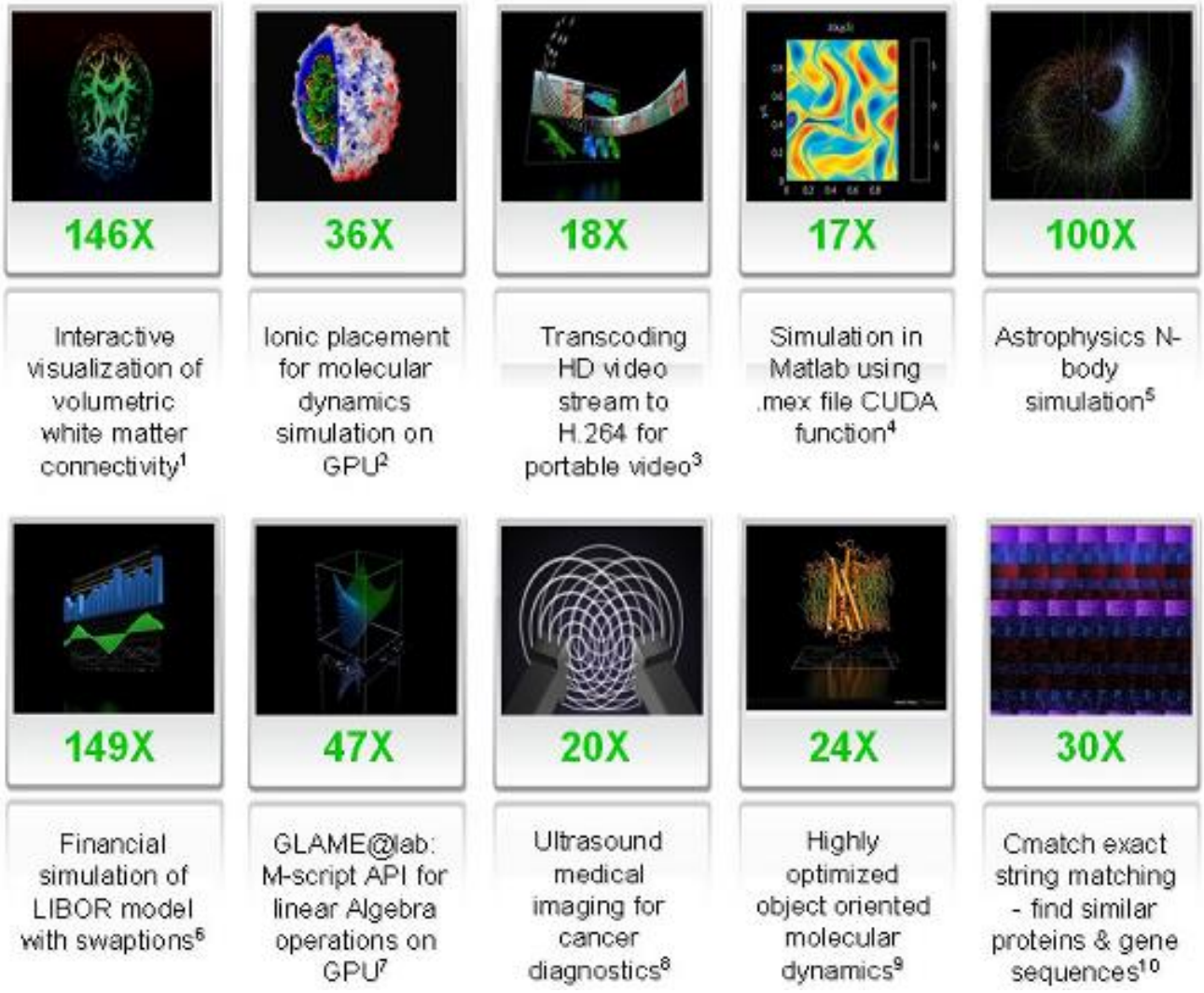
- **SIMD arhitektura**, umeren radni takt procesora
- **Male keš memorije**
  - Kako bi se pospešio propusni opseg memorije
- **Jednostavno upravljanje**
  - Nema predikcije grananja
  - Nema prosleđivanja podataka
- **Energetski efikasne ALU**
  - Mnogo jedinica sa dugačkom latencijom, dugačak protočni sistem (engl. pipeline) kako bi se ostvario visok propusni opseg
  - **Zahteva veoma veliki broj niti kako bi se "sakrila" latencija**

CPU i GPU su komplementarni

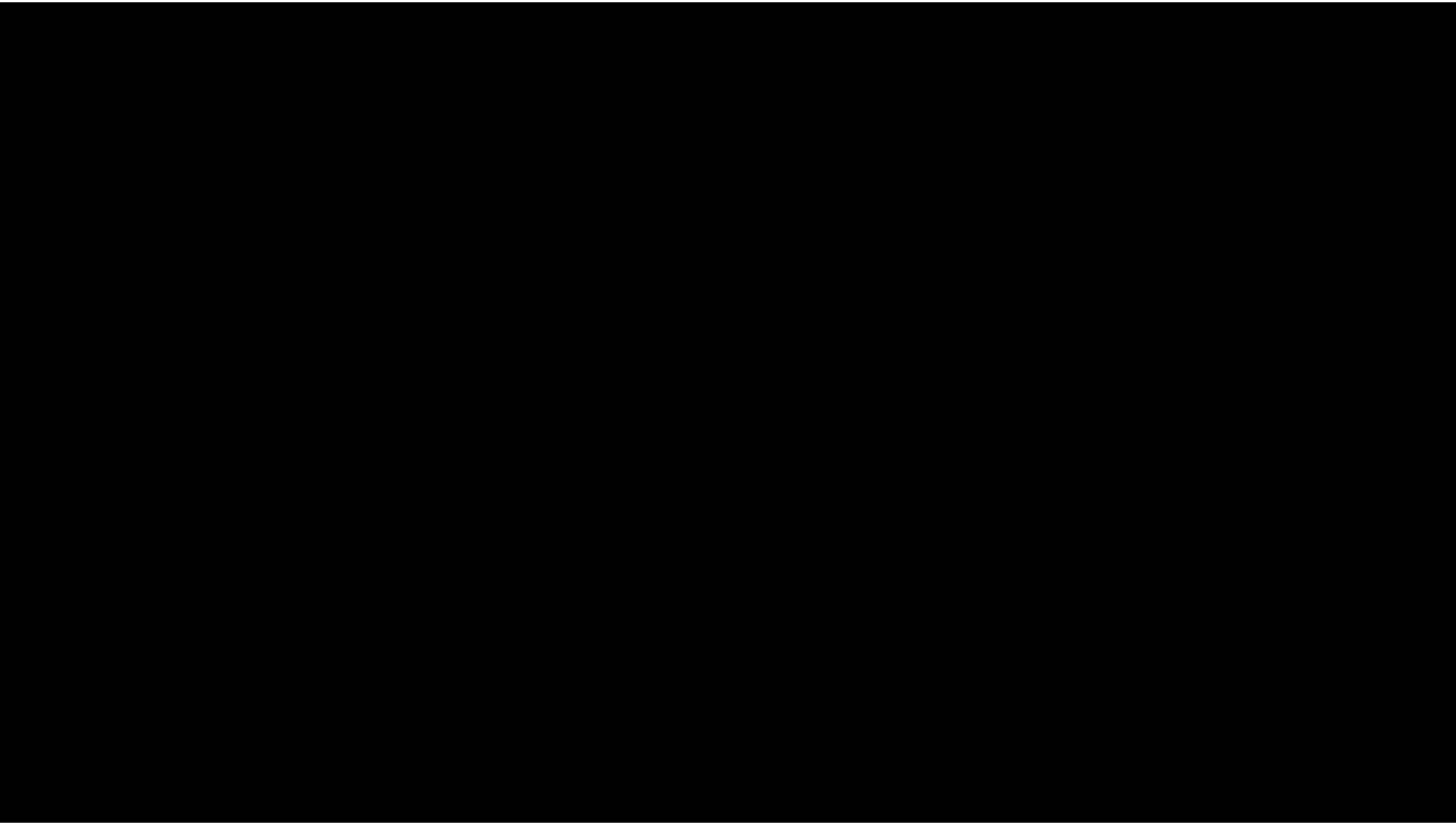
- **CPU za sekvencijalne delove gde je latencija važna**
  - CPU mogu biti za više redova veličina (engl. orders of magnitude) brži od GPU prilikom izvršavanja sekvencijalnog programskog koda
- **GPU za paralelne delove gde propusni opseg pobeđuje**
  - GPU mogu biti za više redova veličina brži od CPU prilikom izvršavanja paralelnog programskog koda

GPGPU – Izvršavanje paralelnih algoritama opšte namene na grafičkim procesorima

Primene GPU izračunavanja



Izvor: Nvidia



Izvor: Nvidia GTC 2012 Kepler GPU Demo, https://www.youtube.com/watch?v=MNbmpVVhfJw

Problemi pogodni za rešavanje na GPU

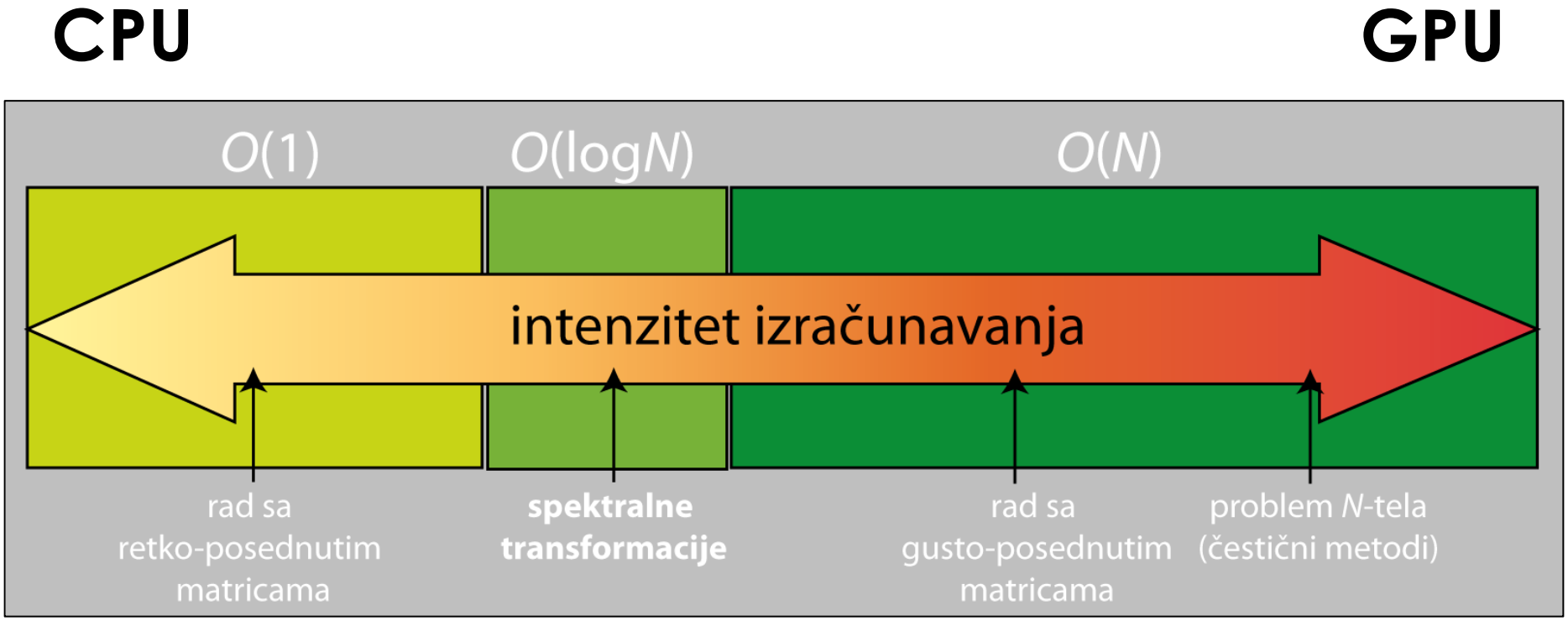
1. Zahtevaju **složena izračunavanja nad velikim skupovima podataka**

2. Karakterišu se **visokim aritmetičkim intenzitetom izračunavanja** (odnos broja ALU i memorijskih operacija)

3. Omogućavaju **značajan paralelizam – prijatno paralelni problemi** (engl. *pleasingly parallel problems*) su poželjni

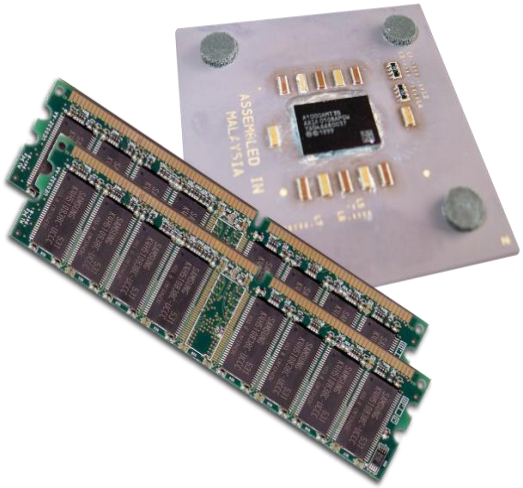
4. Značajno više zavise od **ukupnog propusnog opsega** nego od latencije svake pojedinačne operacije

Intenzitet izračunavanja



Heterogeno računarstvo

- Terminologija:
  - **Domaćin** (engl. *host*) je CPU i njegova memorija (memorija domaćina – *host memory*)
  - **Uređaj** (engl. *device*) je GPU i njegova memorija (memorija uređaja – *device memory*)



Domaćin (Host)

Uređaj (Device)

Heterogeno računarstvo

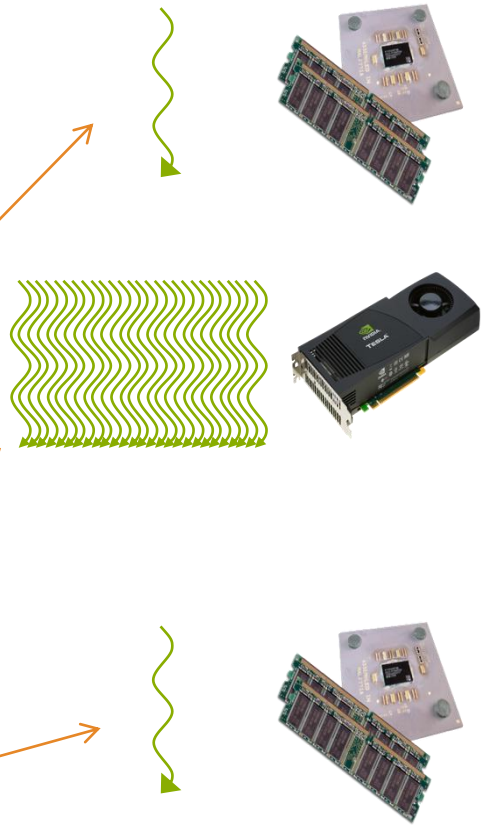
```
#include <iostream>
#include <algorithm>

using namespace std;

#define N          1024
#define RADIUS     3
#define BLOCK_SIZE 16

__global__ void stencil_1d(int *in, int *out) {
        __shared__ int temp[BLOCK_SIZE + 2 * RADIUS];
        int gindex = threadIdx.x + blockIdx.x * blockDim.x;
        int lindex = threadIdx.x + RADIUS;

        // Read input elements into shared memory
        temp[lindex] = in[gindex];
        if (threadIdx.x < RADIUS) {
                temp[lindex - RADIUS] = in[gindex - RADIUS];
                temp[lindex + BLOCK_SIZE] = in[gindex + BLOCK_SIZE];
        }

        // Synchronize (ensure all the data is available)
        __syncthreads();

        // Apply the stencil
        int result = 0;
        for (int offset = -RADIUS ; offset <= RADIUS ; offset++)
                result += temp[lindex + offset];

        // Store the result
        out[gindex] = result;
}

void fill_ints(int *x, int n) {
        fill_n(x, n, 1);
}

int main(void) {
        int *in, *out;          // host copies of a, b, c
        int *d_in, *d_out;      // device copies of a, b, c
        int size = (N + 2*RADIUS) * sizeof(int);

        // Alloc space for host copies and setup values
        in  = (int *)malloc(size); fill_ints(in,  N + 2*RADIUS);
        out = (int *)malloc(size); fill_ints(out, N + 2*RADIUS);

        // Alloc space for device copies
        cudaMalloc((void **)&d_in,  size);
        cudaMalloc((void **)&d_out, size);

        // Copy to device
        cudaMemcpy(d_in,  in,  size, cudaMemcpyHostToDevice);
        cudaMemcpy(d_out, out, size, cudaMemcpyHostToDevice);

        // Launch stencil_1d() kernel on GPU
        stencil_1d<<<N/BLOCK_SIZE,BLOCK_SIZE>>>(d_in + RADIUS,
d_out + RADIUS);

        // Copy result back to host
        cudaMemcpy(out, d_out, size, cudaMemcpyDeviceToHost);

        // Cleanup
        free(in); free(out);
        cudaFree(d_in); cudaFree(d_out);
        return 0;
}
```

paralelne funkcije

serijski kod

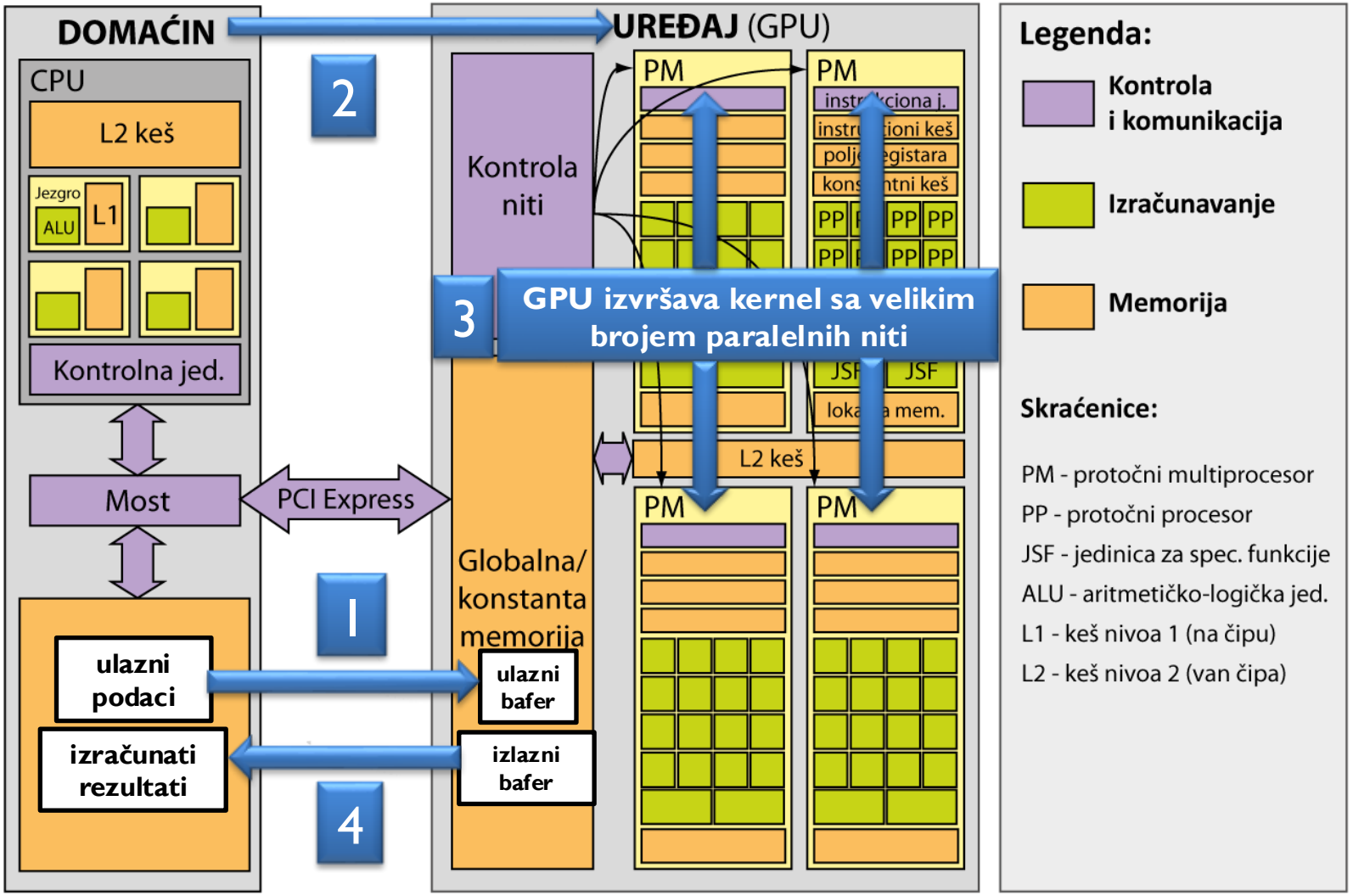
paralelni kod

serijski kod



Izvor: Nvidia

Rad GPGPU programa



Trendovi u arhitekturi GPU

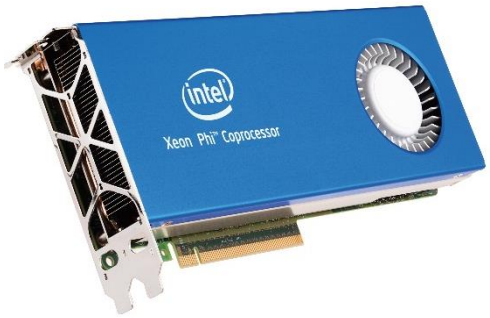
Arhitektura	Fermi	Kepler	Maxwell	Pascal	Turing
CUDA jezgra	512	1536	2048	3840	4352
Frekvencija (GHz)	1.5	1.0	1.1	1.5	1.35
PP po PM[1]	32	192	128	96	88
Propusni opseg (GBs/s)	192	288	336	548	616
TDP (W)	244	195	165	250	250

[1] PP - protočni procesor, PM – protočni multiprocesor

Više izračunavanja za **manje energije**

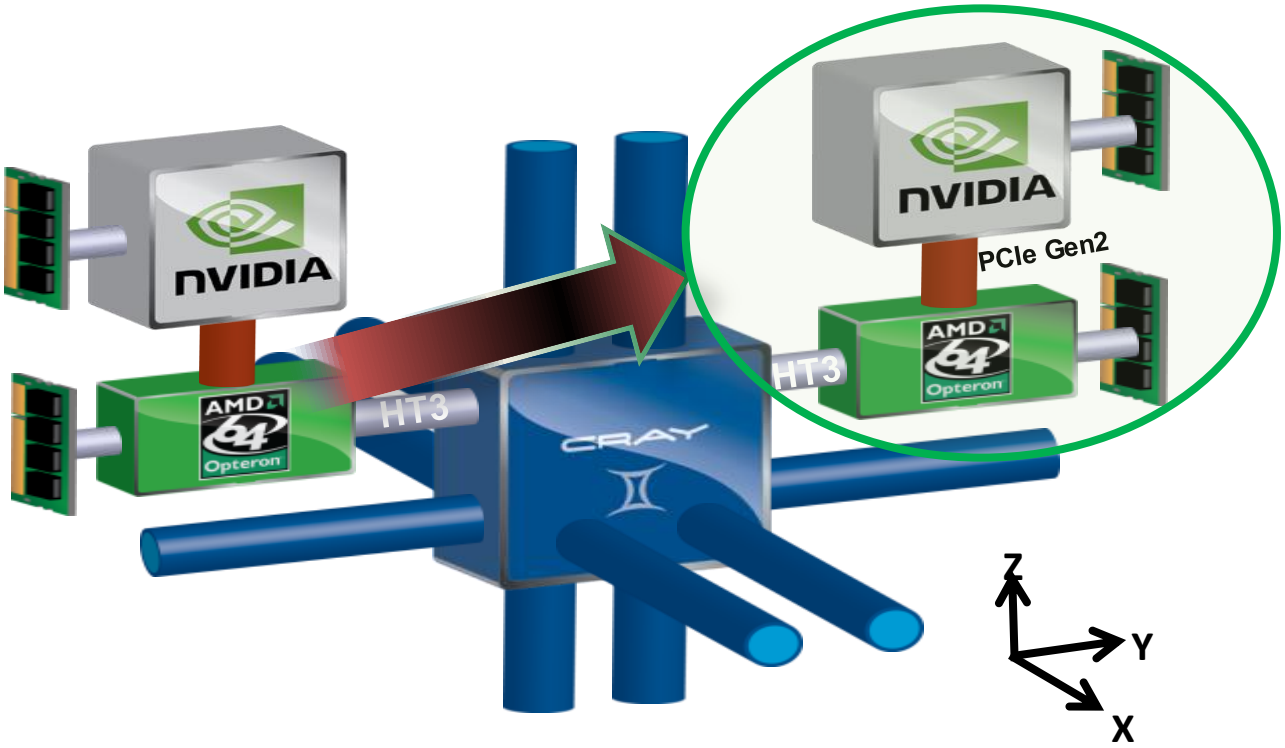
Integracija CPU i GPU kako bi se izbeglo PCIe usko grlo

Pojava Intel Xeon Phi i sličnih akceleratora



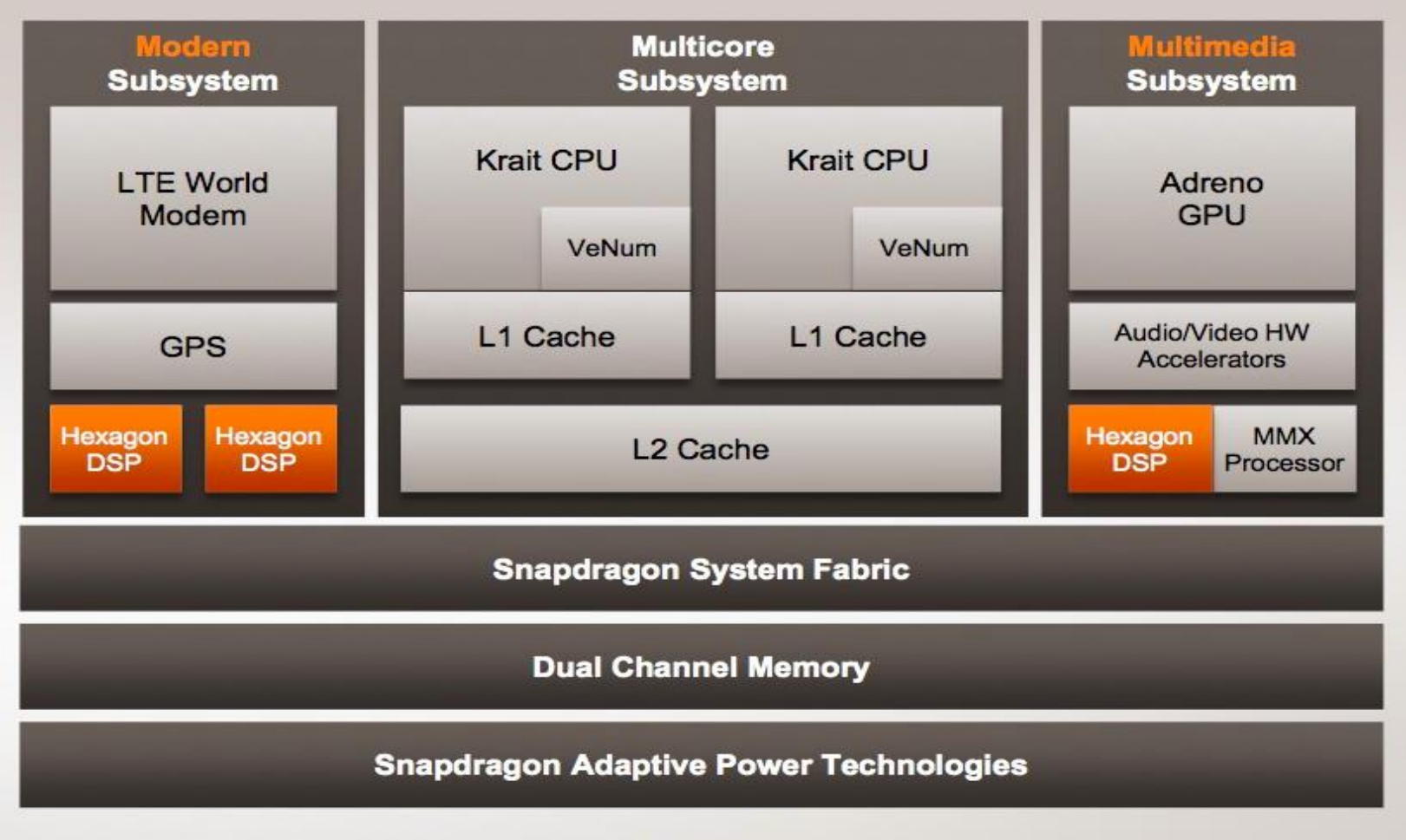
Cray Compute Node

Karakterstike XK7 čvora za izračunavanja
AMD Series 6200 (Interlagos)
NVIDIA Kepler
Memorija domaćina 32GB 1600 MT/s DDR3
NVIDIA Tesla X2090 Memorija 6GB GDDR5
Gemini High Speed Interconnect
Kepler u finalnoj instalaciji



Izvor: Nvidia

SoC arhitektura pametnih telefona



Izvor: Nvidia