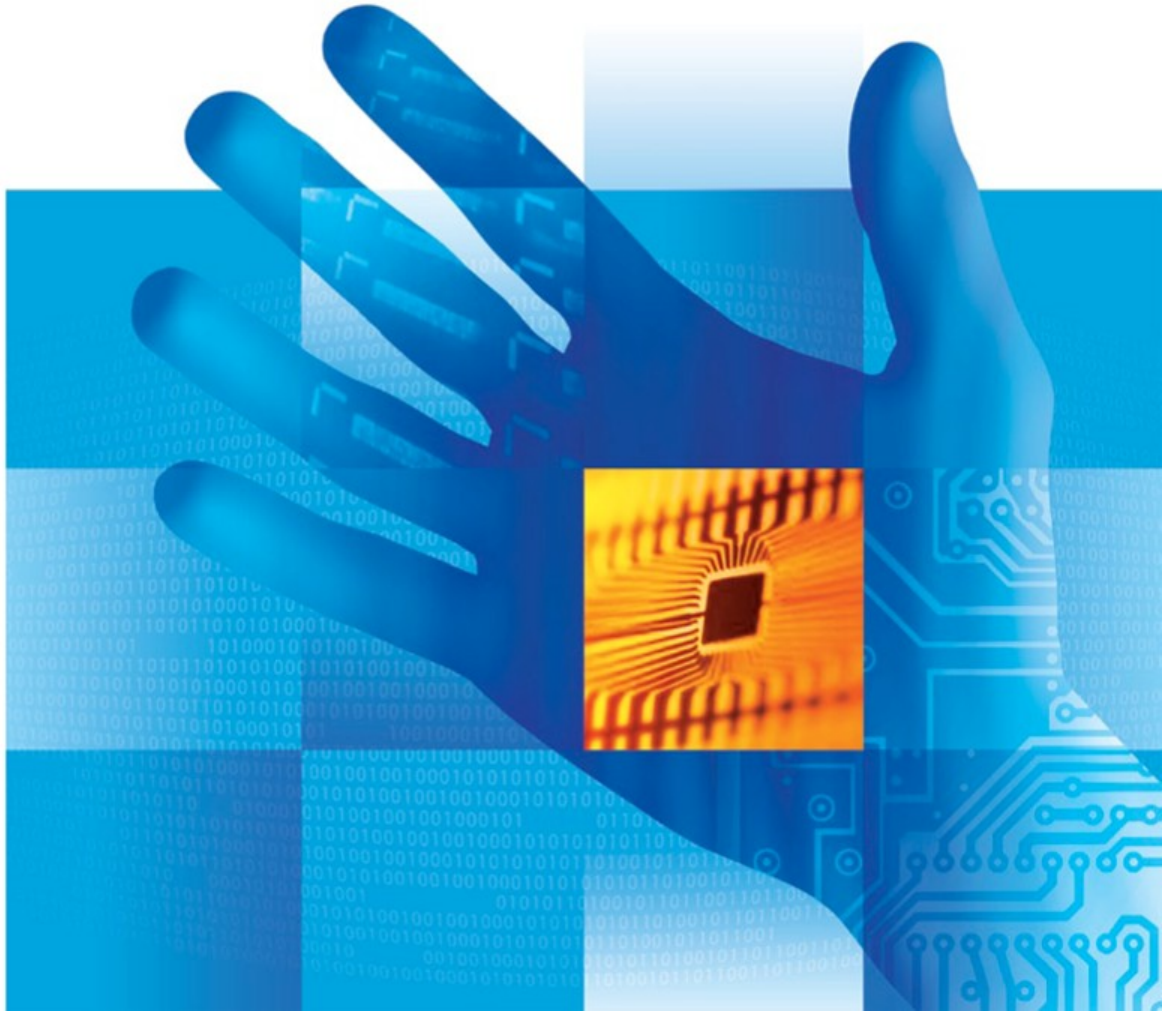
Předmět APPS
GPGPU a CUDA

Petr Olivka
katedra informatiky



Obsah

- Historie GPU
- CUDA
- Architektura
- Komunikace CPU - GPU

Historie GPU

- 1970 – ANTIC v 8 bitovém Atari
- 1980 – IBM 8514
- 1993 – založena Nvidia Co.
- 1994 – založeno 3dfx Interactive
- 1995 – čip NV1 od Nvidia
- 1996 - 3dfx vydalo Voodoo Graphics
- 1999 - GeForce 256 by Nvidia – podpora geom. transf.
- 2000 - Nvidia kupuje 3dfx Interactive
- 2002 - GeForce 4 vybaveno pixel a vertex shadery
- 2006 - GeForce 8 - unifikovaná architektura (nerozlišuje pixel a vertex shader) (Nvidia CUDA)
- 2008 - GeForce 280 - podpora dvojité přesnosti
- 2010 - GeForce 480 (Fermi) - první GPU postavené pro obecné výpočty - GPGPU

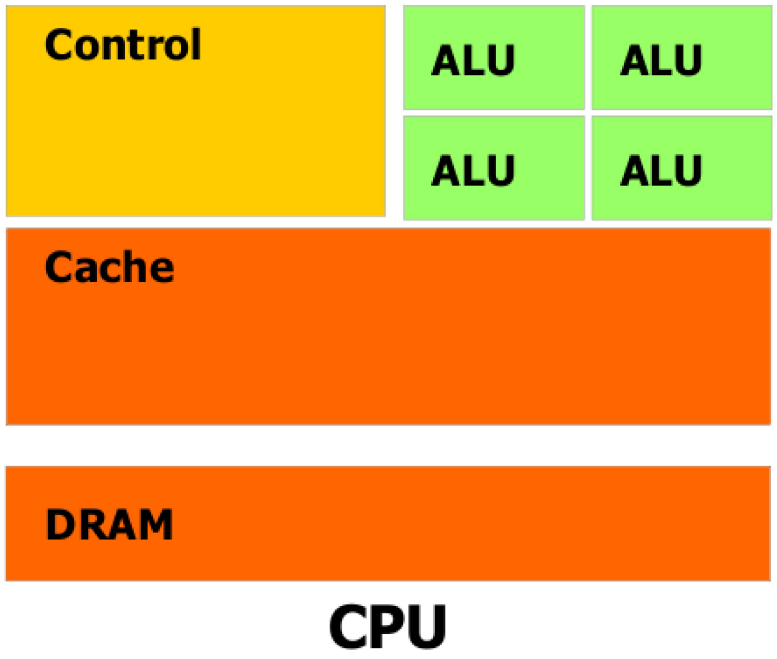
Výhody GPU

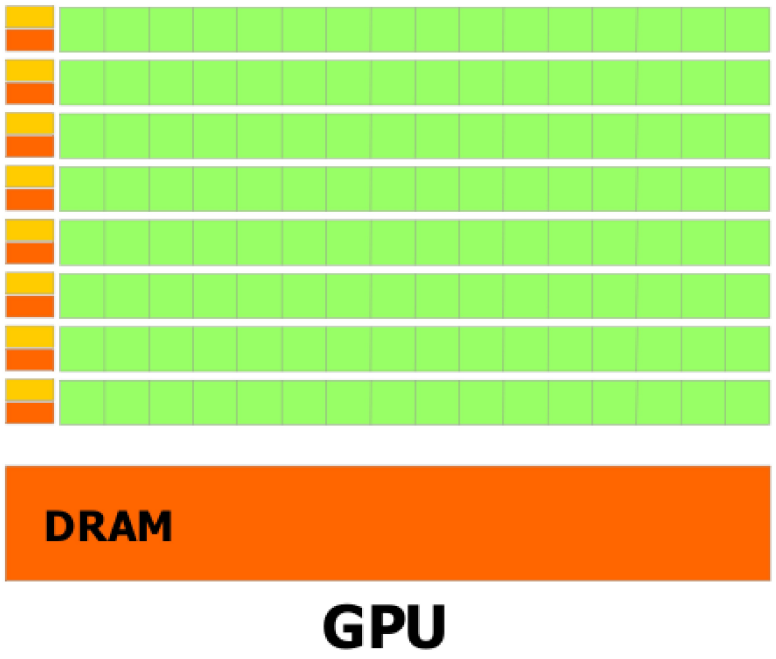
- GPU je navrženo pro současný běh stovek vláken - virtuálně až stovek tisíc vláken
- Vlákna musí být nezávislá, není zaručeno, v jakém pořadí budou provedena
- GPU je vhodné pro intenzivní výpočty s malým počtem podmínek
- Není zde podpora spekulativního zpracování
- GPU je optimalizována pro sekvenční přístup do paměti. Přenosové rychlosti stovky GB/s

Porovnání CPU - GPU

	Nvidia GeForce 580	Intel i7-960 6xCore
Tranzistory	3000 * 10^6	1170 * 10^6
Frekvence	1.5 GHz	3.5 GHz
Počet vláken	512	12
Max. výkon	1.77 Tflops	~200 GFLops
Propustnost	194 GB/s	26 GB/s
RAM	1.5 GB	~48GB
Výkon	244W	130W

Výhody GPU





Maximální možný počet tranzistorů je použit na výpočetní jednotky, ne na cache a řízení toku dat.

Podstata GPGPU

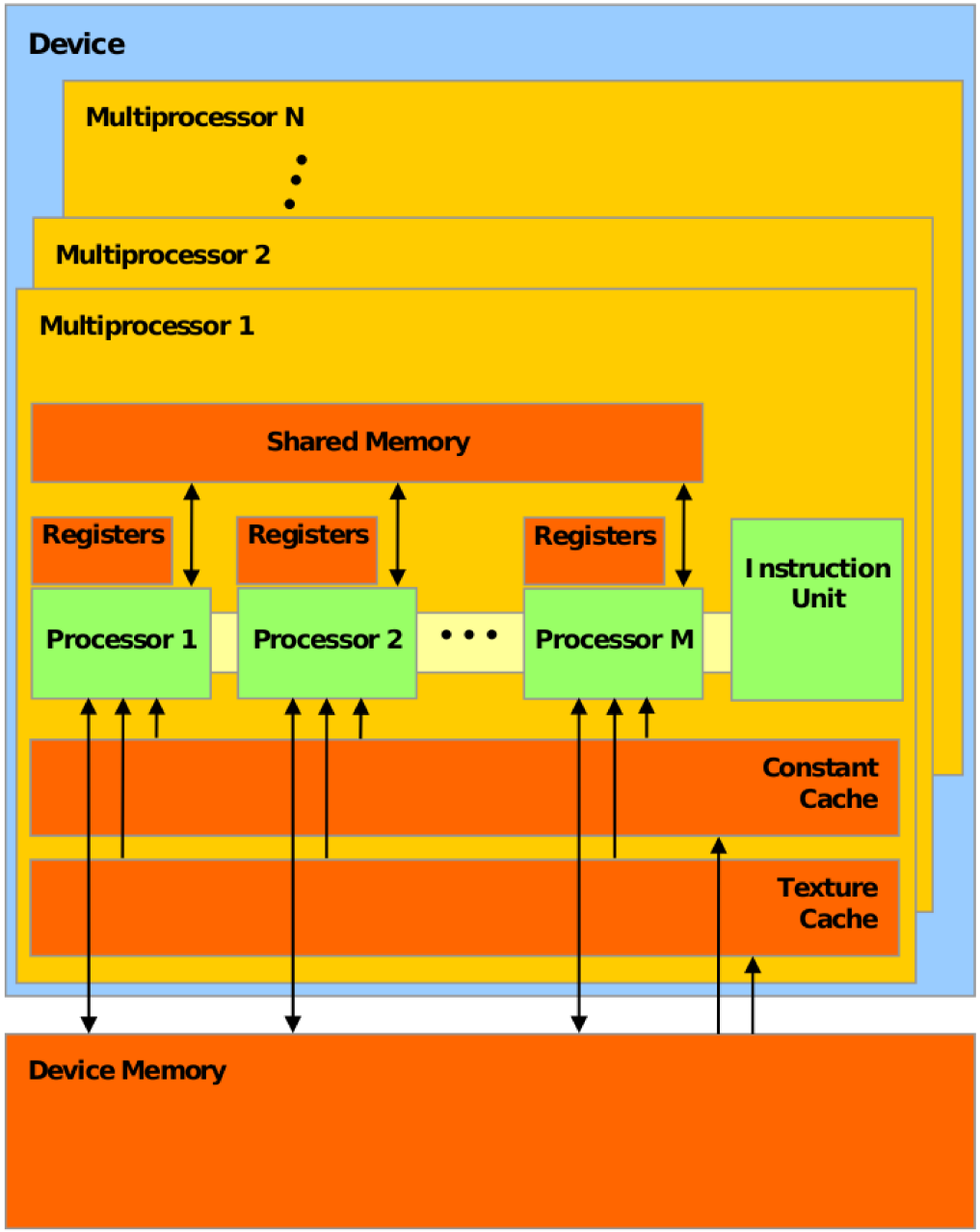
- Na počátku bylo nutno pro GPGPU výpočty používat rozhraní OpenGL
- Úlohy byly formulovány pomocí textur a operací s body
- Vývojáři her potřebovali univerzálnější hardware – vznikly pixel shadery
  - Jde o jednoduchý programovatelný procesor pro operace s body
  - Má podporu pro výpočty s plovoucí desetinnou tečkou jednoduché přesnosti
  - Kód je omezen na několi desítek instrukcí

CUDA – Compute Unified Device Architecture (Nvidia 2/2007)

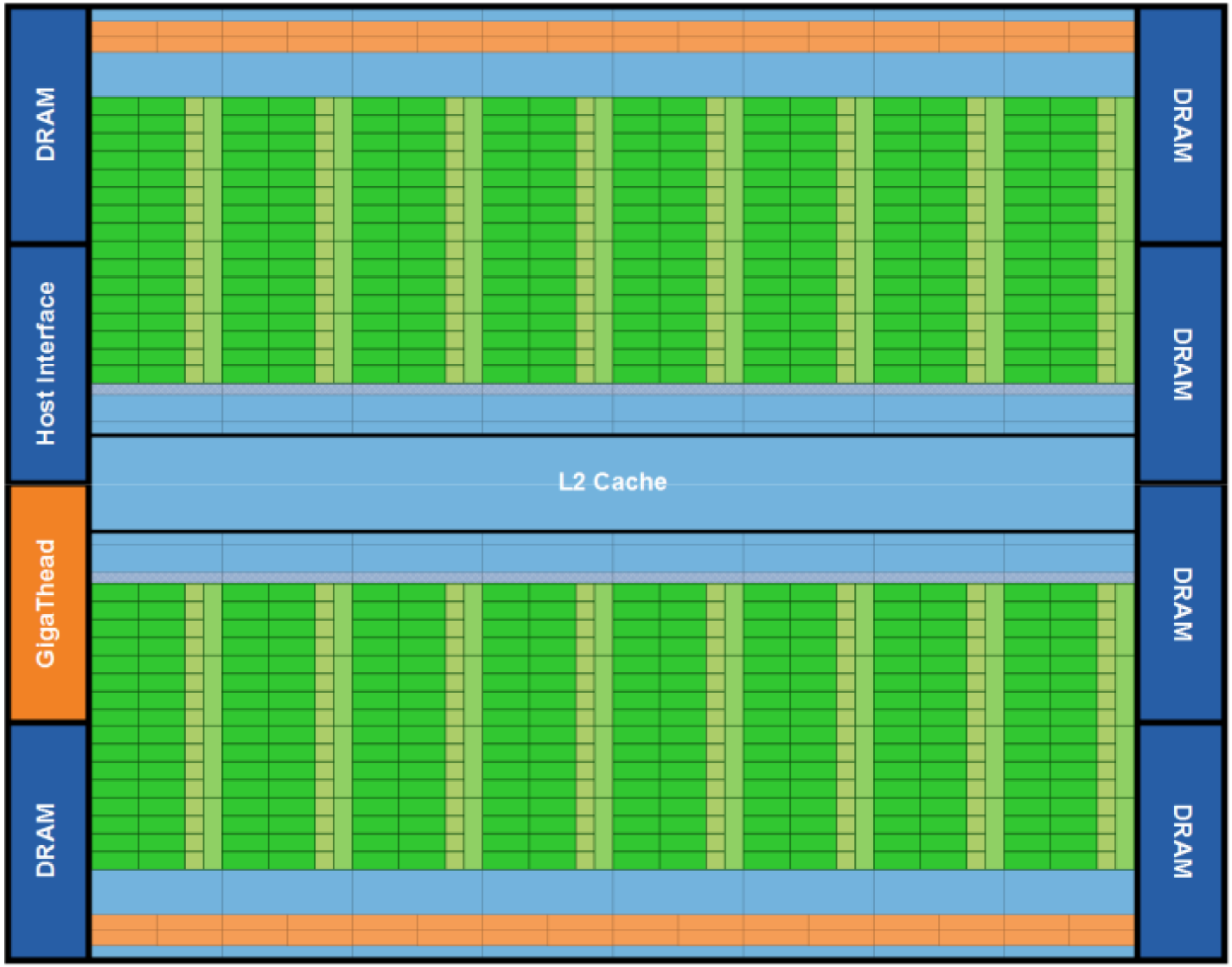
- Výrazně jednodušší programování GPGPU
- Zcela odstraněna nutnost pracovat s OpenGL a definování úloh pomocí textur
- Je založena na jednoduchém rozšíření jazyka C/C++
- Funguje jen s kartami Nvidia

Je velmi snadné napsat kód pro CUDA, ale je potřeba mít hluboké znalosti o GPU, aby výsledný kód byl efektivní.

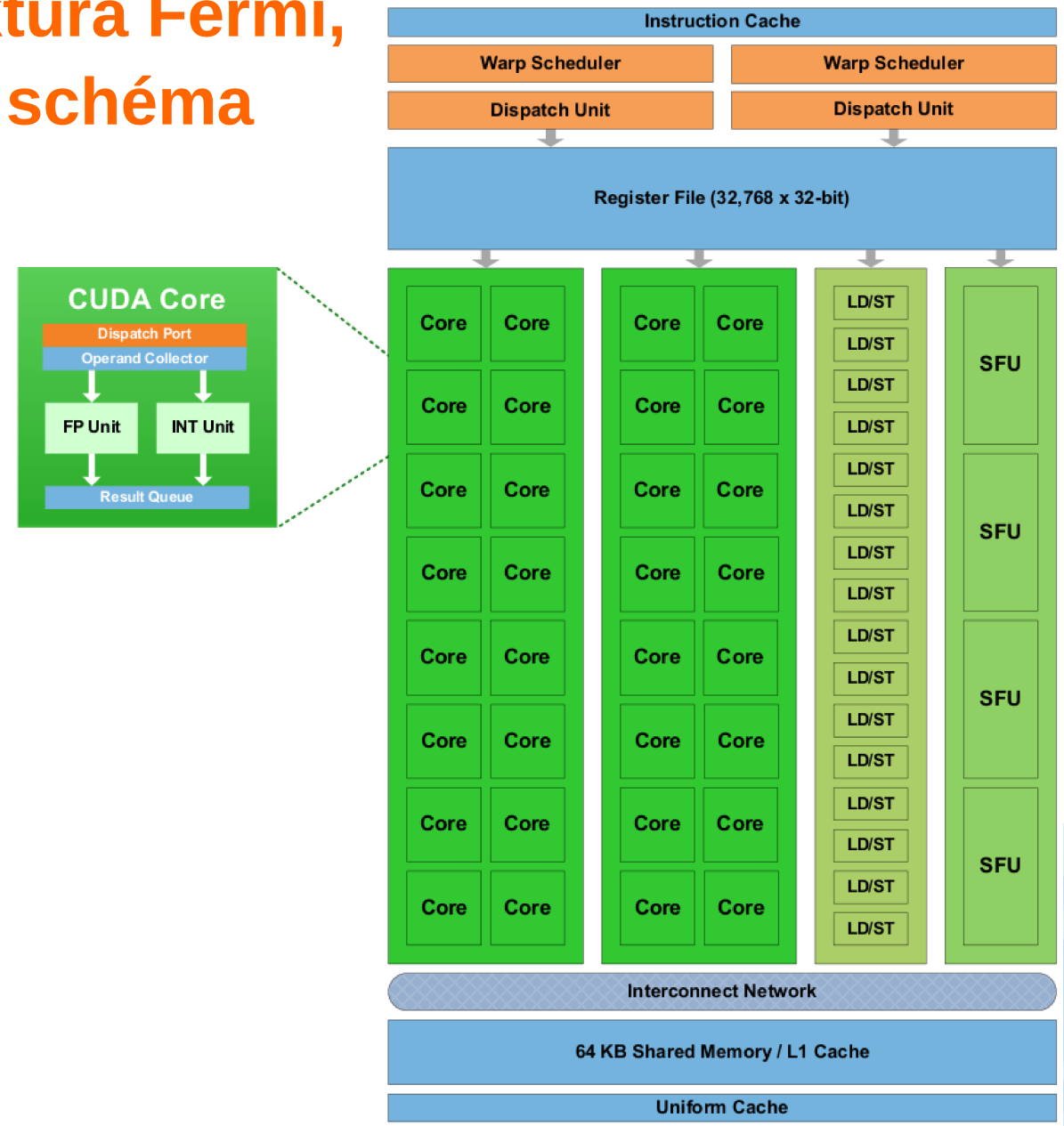
Architektura



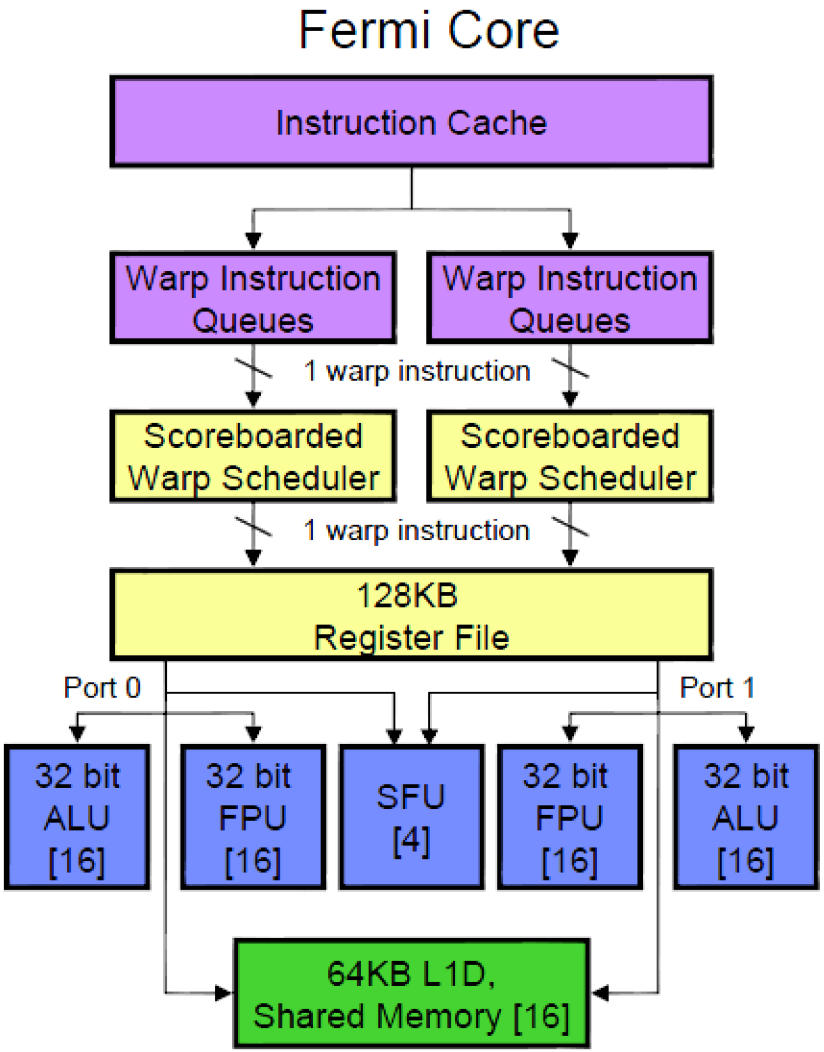
Architektura Fermi - ????



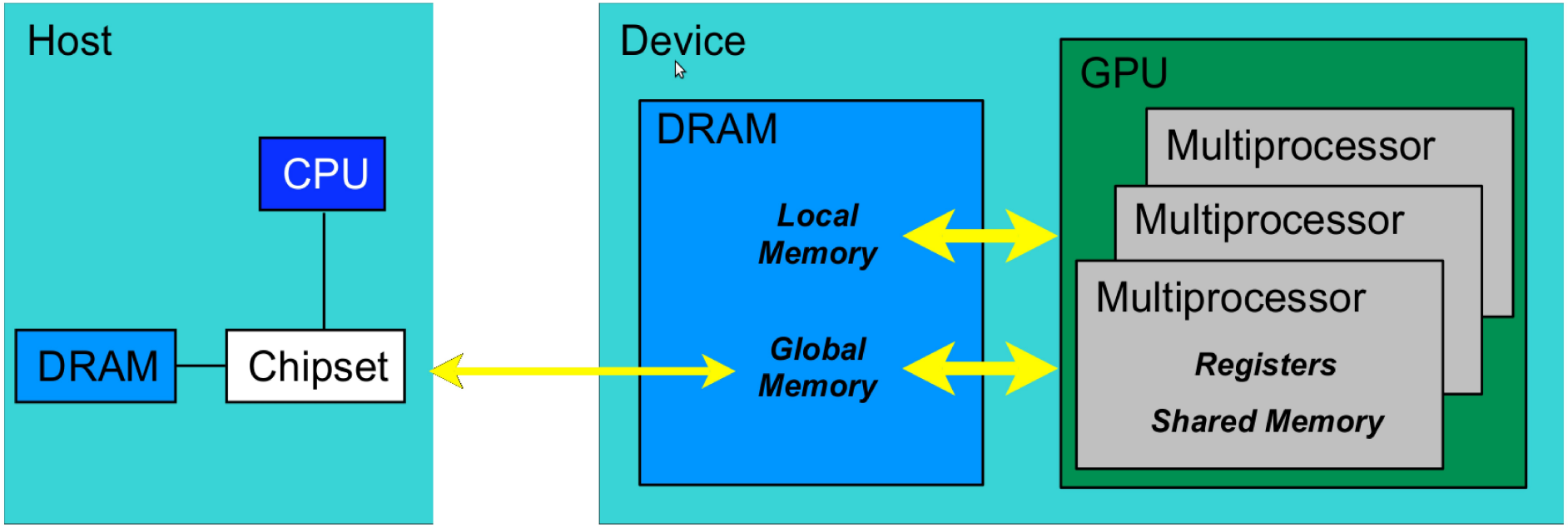
Architektura Fermi, blokové schéma



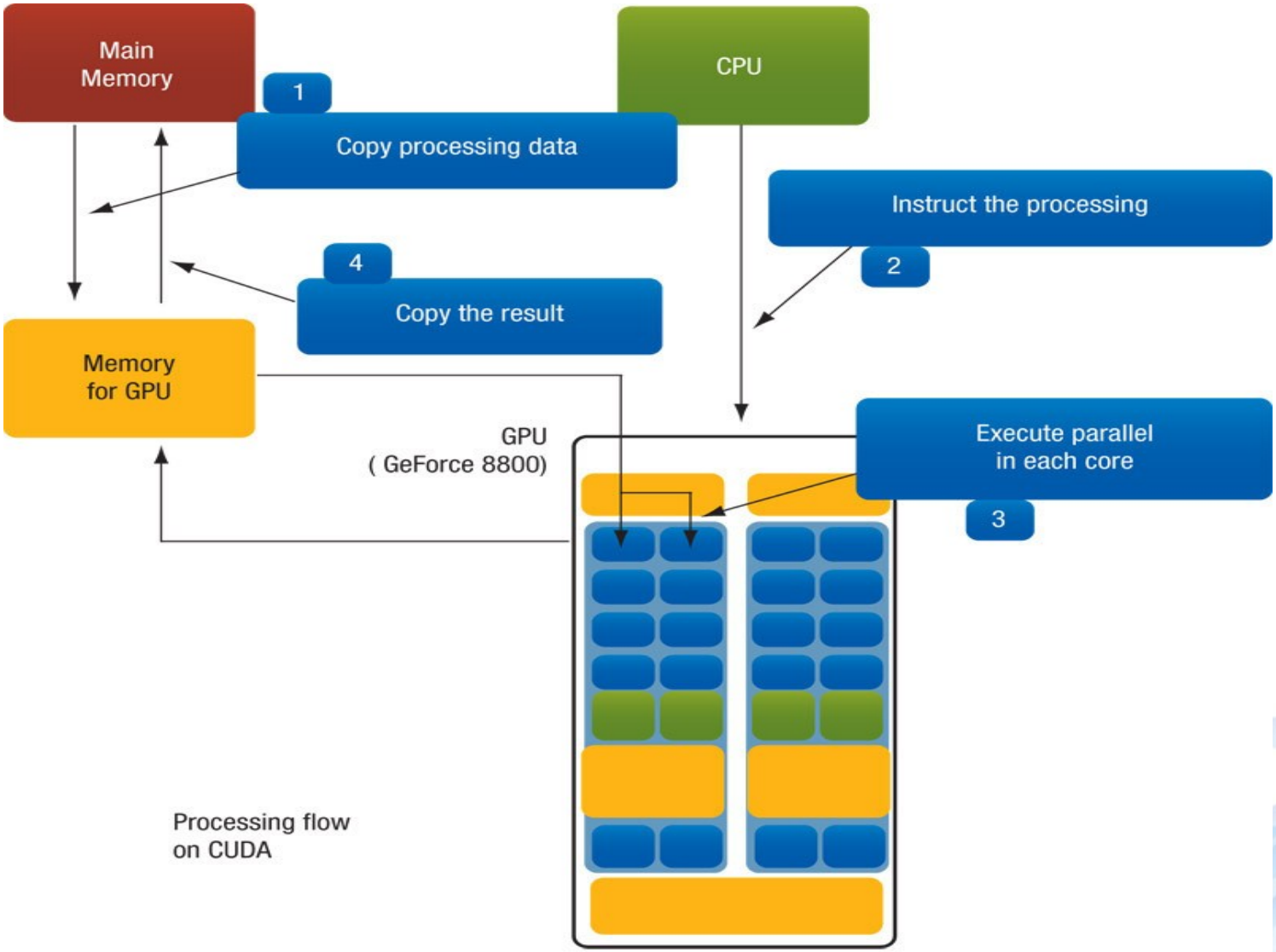
Architektura Fermi, blokové schéma



Paměťový model v PC



Postup výpočtu



Processing flow on CUDA

Komunikace mezi CPU a GPU

- Komunikace přes PCI-Express je velmi pomalá, méně než 5GB/s
- Je nutno tuto komunikaci minimalizovat, ideální jen přesun dat na začátku a na konci výpočtu.
- GPU se nevyplatí pro úlohy s nízkou výpočetní intenzitou
- Výhodnější mohou být GPU on-board se sdílenou pamětí
- Pokud se provádí častá komunikace CPU-GPU, je výhodné použití pipeliningu
- Je možné současně provádět: výpočet na CPU, výpočet na GPU, kopírovat data do GPU, kopírovat data z GPU

Vývoj efektivního kódu

Pro vývoj efektivního kódu je potřeba dodržet následující pravidla:

- Redukovat přenos dat mezi CPU a GPU
- Optimalizovat přístup do globální paměti GPU
- Omezit divergentní vlákna
- Zvolit správnou velikost bloků vláken