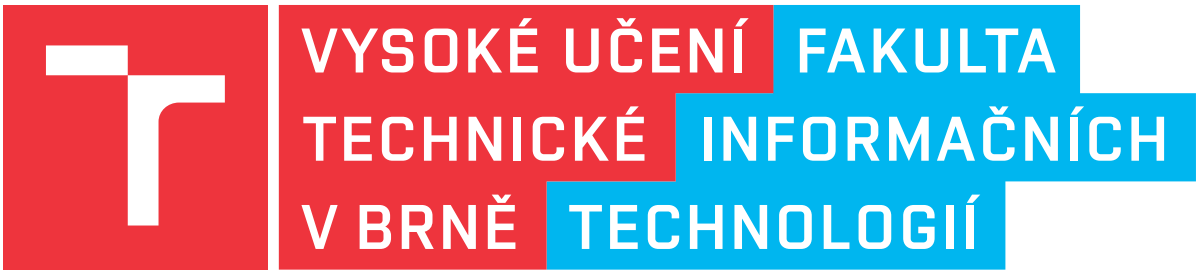


Grafické a multimediální procesory (GMU)

Příprava na zkoušku

https://github.com/VasaMM/GMU-otazky

9. ledna 2020

Obsah

1 Popište základní principy návrhu energeticky úsporného procesoru CPU/GPU, jak se vyhodnocuje energetická úspornost. 4

2 [TODO] Popište vývoj 2D/3D grafického řetězce s různým stupněm zřetězení. 4

3 Popište 1- až n-rozměrné multiprocesorové propojovací struktury (Origin/Onyx) a typy propojovacích sběrnic Silicon Graphics. 5

4 Popište principy výstavby, činnosti a použití GPGPU. 5

5 [TODO] Popište koncepci a vlastnosti různých typů Streaming Multiprocesoru 6

6 Popište principy komprese dat v systému Pascal (delta-komprese a další). 6

7 [TODO] Popište architekturu grafických multiprocesorů a nové principy činnosti, jako komprese dat, preempce a její typy. 6

8 Popište princip tensorového jádra Turing, k čemu slouží, jaké formáty dat se používají. 6

9 Popište principy a funkce preempce Pascal. 7

10 [TODO] Popište vývoj unifikovaného adresového prostoru, principy činnosti, a jeho hardwarovou podporu. 7

11 [TODO] Popište principy návrhu energeticky úsporného GPU pro mobilní zařízení. 7

12 [TODO] Popište hlavní vývojové etapy a principy grafických systémů Mali pro mobilní zařízení. 8

13 Vysvětlete a zdůvodněte koncepci kachliček (tiles) 16x16 pixelů v grafice Mali. 8

14 [TODO] Popište formáty dat podporované v grafice Mali, skalární i vektorové. 8

15 [TODO] Popište a zdůvodněte principy tří základních typů komprese textur - ztrátové, bezeztrátové a adaptivní. 8

16 Popište alespoň 3 rozdíly mezi architekturami GPU a CPU. 8

17 Co je to kernel? 9

18 Co je to vlákno? 9

19 Co je to divergence vláken a kdy vzniká? 9

20 Co je to warp/wavefront? 9

21 Co je to multiprocesor (streaming multiprocessor/compute unit) GPU a k čemu slouží. 9

22 Co je to fronta příkazů (command queue)? 10

23 [TODO] Jakým způsobem je možné synchronizovat úlohy ve frontě příkazů s vykonáváním mimo pořadí (out-of-order execution) nebo mezi frontami v OpenCL? 10

24 [TODO] Jakým způsobem se eliminuje/zmírňuje dopad aritmetických/paměťových latencí. 10

25 [TODO] Co je to konflikt banků v lokální paměti a kdy vzniká? 10

26 [TODO] Co je to zarovnaný přístup do paměti? 10

27 Popište konstantní/uniformní paměť. 10

28 Popište lokální/sdílenou paměť. 10

29 [TODO] K čemu slouží texturovací jednotky? 10

30 [TODO] Jaké jsou rozdíly mezi bufferem a texturou v OpenCL? 10

31 Jaký je rozdíl mezi normalizovanými a nenormalizovanými texturovacími koordináty? 11

32 Co je to register spilling? 11

33 Jaké jsou způsoby komunikace mezi vlákny ve skupině a mezi skupinami? 11

34 K čemu slouží atomické instrukce? Popište jejich vlastnosti a příklad algoritmu kde je možné je využít. 11

35 Jaké mohou být příčiny nízké výkonnosti kernelů? 11

36 [TODO] Co je to obsazenost multiprocesoru (occupancy), na čem je závislá? 12

37 [TODO] Co je to aritmetická a paměťová latence? 12

38 [TODO] Co je to separabilní filtr a jakým způsobem lze přistupovat k jeho paralelizaci? 12

39 K čemu slouží paralelní redukce? 12

40 Co je to dynamický paralelismus? 12

41 [TODO] Jaké jsou možné typy přenosů dat mezi CPU a GPU? 13

42 K čemu slouží P2P přístup a jakým způsobem ho lze využít. 13

43 [TODO] K čemu slouží operace shuffle? 13

44 Vyjmenujte alespoň 3 api pro GP-GPU. 13

45 Co to je SpirV? 13

46 [TODO] K čemu slouží operace ballot? 13

47 [TODO] K čemu lze využít atomicCompSwap? 13

48 [TODO] Jaký je vztah mezi NDRange/Grid/Dispatch, pracovní skupinou a vláknem/invokací? 13

49 Jaké paměti jsou na GPU a popište jejich vlastnosti (odkud je lze plnit, odkud je lze číst, relativní rychlost, relativní velikost). 13

50 Jaký je rozdíl mezi pracovní skupinou a warpem/wavefrontem. 15

51 K čemu slouží příkaz barrier? Jakou má souvislost s rozdělením vláken do skupin, warpů? Jaký to má vztah k větvení programu? 15

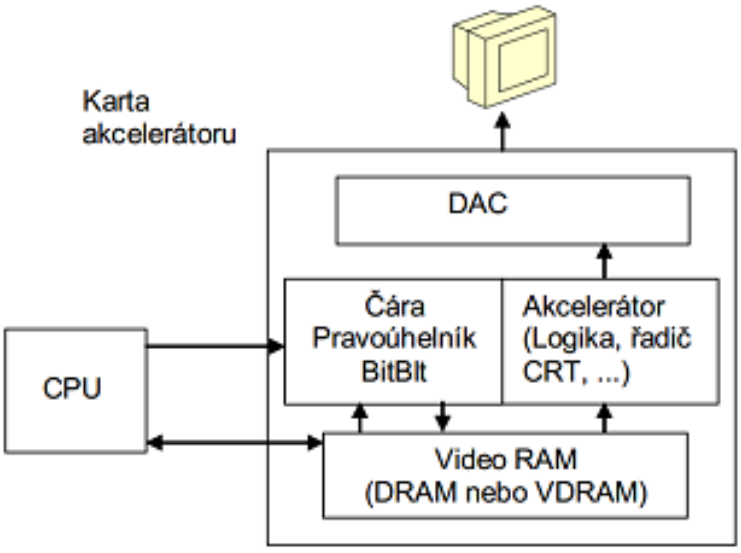
1 Popište základní principy návrhu energeticky úsporného procesoru CPU/GPU, jak se vyhodnocuje energetická úspornost.

Energetická úspornost: MFLOPS/W

- další snižovaní napájecího napětí není možné
- energie na operaci se snižuje lineárně s rozměry prvku
- současné technologie 3 W pro mobilní zařízení, 150 W stolní pc
- v klasické jednovláknové architektuře se většina energie spotřebuje na režii dodání dat (načtení operandů spotřebuje více E než operace)
- Řešení úpravou výpočetní aritmetiky
  - Snížení přesnosti kódu čísel, např. odseknutím n nejnižších bitů
  - přibližné počítání je samostatný specializovaný obor
- Řešení lokalitou dat
  - energeticky efektivní architektury musí snížit objem dat přenesených na instrukci
  - např. přístup do mimočipové paměti 250× dražší
  - požadavek ma zvýšení šířky pásma paměti mimo čip a snížení příkonu komunikace

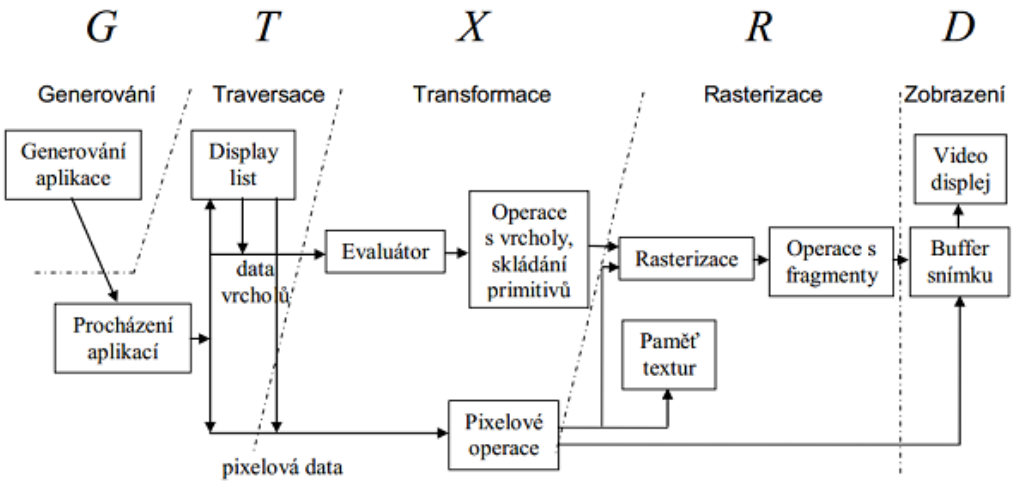
2 [TODO] Popište vývoj 2D/3D grafického řetězce s různým stupněm zřetězení.

GMU 02



Obrázek 1: Základní 2D zpracování (IBM 8514)

generovanie → traverzacia → transformacia → rasterizadia → zobrazenie (display) Implementovana napr. u i740, Savage3D



Obrázek 2: Akeleyho pipeline

3 Popište 1- až n-rozměrné multiprocesorové propojovací struktury (Origin/Onyx) a typy propojovacích sběrnic Silicon Graphics.

Origin

Propojovací struktura architektury koncepce Origin2000 umožnuje vytváret až 5D krychle. Díky její pružnosti a koncepcní pokrokovosti byla prevzata i do navazujících architektur SGI, jako napríklad Onyx2, a další.

Architektura Origin2000 je tvorena množstvím výpoctových uzlu, propojených navzájem propojovací sítí CrayLink. Každý výpoctový uzel obsahuje jeden nebo dva procesory s pametí cache, sdílenou pamet, adresár pro rízení koherence cache s hlavní pametí a dve ruzné jednotky interface – jeden, XIO, který pripojuje I/O jednotky, a druhý, CrayLink, propojující uzly pres jednotku Router.

Onyx2

Každý procesor je spolu se svou pamětí cache a částí operační paměti počítače (o kapacitě od 128 MB do 8 GB) součástí jednoho procesorového uzlu, který obsahuje čtyři procesory. Obousměrný přenos dat mezi procesory v jednom uzlu dosahuje hodnoty 1,6 GBs-1, resp. 800 MBs-1 v každém směru zvlášť. Jednotlivé uzly jsou mezi sebou propojeny pomocí propojovací sítě, jejíž topologie sice vychází z ideální hyperkrychle, ale z důvodu optimalizace datových přenosů jsou mezi uzly vytvořeny i další přídavné datové spoje (již z principu musí jít o diagonály), zejména mezi procesory a zobrazovacími subsystémy.

4 Popište principy výstavby, činnosti a použití GPGPU.

Jde o techniku využití GPU k obecným (negrafickým) výpočtům, které běžně probíhají na CPU. Zatímco dříve bylo zpracování dat na GPU pevně dané (static pipeline), dnes už je převážná část programovatelná, a tudíž nabízí možnost obecných výpočtů. K tomu slouží např. rozhraní OpenCL, CUDA či ATI Stream.

Z povahy grafických čipů vychází také způsob práce s daty, který je silně orientován na paralelní proudové zpracování dat. Data jsou často rozdělena do 2D mřížky, kde nad každou buňkou pracuje jedna výpočetní jednotka. Tyto jednotky sdílí kód programu, nazývaný také kernel. Možnost synchronizace či sdílení dat mezi jednotkami v průběhu výpočtu je omezená a značně snižuje výsledný výkon.

(Alespoň v OpenCL možnost sdílení paměti existuje.) Z běžných součástí GPU lze použít například texturovací jednotky jako vstupní rozhraní a framebuffer jako výstupní.

Syntaxe programovacího jazyka může být podmnožinou jazyka C (v OpenCL), ovšem existují omezení plynoucí právě z paralelního proudového zpracování. Není možné používat rekurzi, volání vnořených funkcí má svá omezení, jsou zavedeny speciální funkce a datové typy pro práci s vektory, je žádoucí znovuvyužívat použité proměnné a obecně musí optimalizace probíhat jinak než u běžných (CPU) programů - s ohledem na povahu HW.

Vhodnými úlohami pro zpracování na GPU je například zpracování videa, zvuku či řeči, provádění fyzikálních simulací (kapaliny, osvětlení, počasí) anebo třeba úlohy z kryptografie.

5 [TODO] Popište koncepci a vlastnosti různých typů Streaming Multiprocesoru

GMU 08

6 Popište principy komprese dat v systému Pascal (delta-komprese a další).

Komprese v paměti cache L2
Jsou uplatněny tři principy delta komprese barev.
GPU vypočítává rozdíly mezi barvami pixelů v bloku a ukládá blok jako referenční pixely plus delta hodnoty – rozdíly od referenčních hodnot.
Pokud jsou delta hodnoty malé, pak je zapotřebí na jeden pixel jen malý počet bitů. Standardní kompresní metoda vede na kompresní poměr 2:1.
GeForce GTX 1080 Pascal zavedla tyto kompresní režimy:

- Zdokonalená komprese 2:1
- Komprese 4:1 pro případ velmi malých hodnot delta
- Kompresní režim 8:1 kombinuje kompresi konstantních barev 4:1 pro bloky 2x2 pixelů s kompresí 2:1 hodnot delta mezi těmito bloky

7 [TODO] Popište architekturu grafických multiprocesorů a nové principy činnosti, jako komprese dat, preempce a její typy.

8 Popište princip tensorového jádra Turing, k čemu slouží, jaké formáty dat se používají.

tensory – veličiny, dané např. třemi nebo čtyřmi vektory

Princip činnosti tensorového jádra

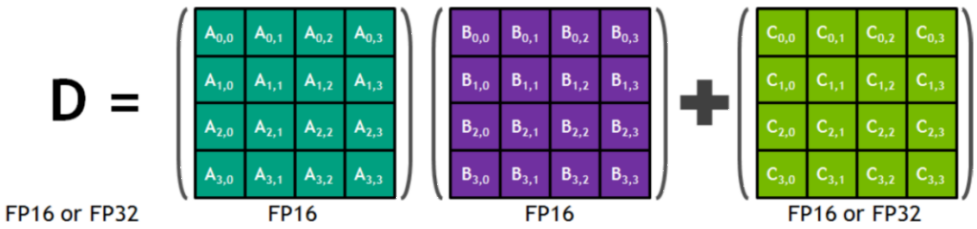
- Používané neuronové sítě mají tisíce vrstev a milióny neuronů.
- Pro jejich rychlé trénování a dedukování je zapotřebí vysoký výpočetní výkon.
- Zde jsou základními operacemi násobení matice – matice (Matrix-Matrix multiplication GEMM).
- Používají se k násobení velkých matic vstupních dat a vah v propojených vrstvách sítě.
- Násobení matic se provádí v některých aplikacích v jednoduché přesnosti.

- Pro trénování a dedukci se používá poloviční přesnosti.
- V jádře Volta se používá smíšené přesnosti, násobení matic se vstupy v FP16 a akumulace v FP32.
- Tensorová jádra jsou navržena pro činnost s vysokou energetickou efektivností.

Každé tensorové jádro pracuje s maticemi 4×4 a provádí operaci Matrix Multiply and Accumulate

$$D = A \times B + C$$

kde A, B, C a D jsou matice 4×4. Vstupy pro maticové násobení A, B jsou v FP16, akumulační matice C, D mohou být matice FP16 nebo FP32.



Obrázek 3: Matrix Multiply and Accumulate

9 Popište principy a funkce preempce Pascal.

Výpočtová preempce

Brání dlouho běžícím aplikacím ovládnout výlučným způsobem celý systém a znemožnit tak běh dalších aplikací, nebo časově překročit.

Preempce pracovní zátěže

V zásadě jde o pozastavení vykreslování, uložení kontextu, kde a jak později pokračovat, a rychle přejít na žádost o preempci.

10 [TODO] Popište vývoj unifikovaného adresového prostoru, principy činnosti, a jeho hardwarovou podporu.

11 [TODO] Popište principy návrhu energeticky úsporného GPU pro mobilní zařízení.

GMU 10

12 [TODO] Popište hlavní vývojové etapy a principy grafických systémů Mali pro mobilní zařízení.

13 Vysvětlete a zdůvodněte koncepci kachliček (tiles) 16x16 pixelů v grafice Mali.

Tento princip je srovnáván s tradičním bezprostředním režimem (immediate mode), používaným typicky v počítačích PC a v terminálech. Vykreslování s použitím kachliček znamená vykreslovat buffer snímku z menších úseků – z kachliček 16×16 pixelů. Když je kachlička dokončená, vypíše se do paměti.

Bezprostřední režim

Vykreslování v bezprostředním režimu se provádí jako přesná posloupnost příkazů pro každý primitiv v každém volání kreslení, nepoužívá se paralelní zpracování a pipelining.

14 [TODO] Popište formáty dat podporované v grafice Mali, skalární i vektorové.

15 [TODO] Popište a zdůvodněte principy tří základních typů komprese textur - ztrátové, bezeztrátové a adaptivní.

Komprese ASTC

Adaptive Scalable Texture Compression je kompresní metoda ARM. Je to ztrátová metoda pro kompresi a dekompresi textur, ukládaných v bufferech v pamětech DDR – Double Data Rate.

Ukládání geometrických dat vyžaduje běžně přes 64 bytů na vrchol, ve srovnání se 4 bity na texel pro Ericsson Texture Compression 2 (ETC2).

Komprese AFBC

ARM Frame Buffer Compression je bezeztrátová kompresní metoda pro kompresi a dekompresi toku videa např. ve formátu H.264. Umožňuje přístup k datům v libovolném místě toku videa.

Metoda AFBC je nyní dostupná pro všechny video stroje Mali, budoucí GPU Mali a jako licencovaný IP pro displejové procesory.

16 Popište alespoň 3 rozdíly mezi architekturami GPU a CPU.

CPU

- programovaní pomocí běžých jazyků (C, C++, ...)
- menší počet komplexních výkonných výpočetních jednotek
- predikce skoků, spekulativní vykonávaní, přejmenovaní registrů, vykonývaní instrukcí mimo pořadí, ...

GPU

- programování/konfigurace jednotek pomocí speciálního API (OpenCL, Vulcan)
- velké množství jednoduchých skalárních procesorů (SIMD jednotek) organizovaných do desítek multiprocesorů

- cca 20× vyšší výkon a 10× paměťová propustnost
- na každý multiprocesor lze naalokovat řádově stovky až tisíce vláken
- přepínání vláken bez penalty
- různé druhy pamětí sloužící různým účelům
- in-order zpracování

17 Co je to kernel?

- program pro GPU, spouští se paralelně na více vláknech zároveň
- sdružuje se do pracovních skupin
- typicky definován v externím kódu (OpenGL, OpenCL), spouští se pomocí API
- lze jej řadit do front

18 Co je to vlákno?

- jedna instance kernelu (viz 17)
- jsou organizovány do pracovních skupin
- každé vlákno má svůj index v rámci skupiny i v rámci kernelu – slouží k indexaci dat, nad kterými má pracovat (až 3D index)

19 Co je to divergence vláken a kdy vzniká?

Když se vlákna jednoho warpu dostanou do různých větví kódu – toto zpomaluje vykonávání programu, protože jednotlivé větve nelze vykonávat najednou (SIMD ne MIMD) a musí se provádět postupně.

Ideálně potřebujeme aby vlákna dělala vždy tu samou instrukci, pokud ne, dochází k divergenci.

20 Co je to warp/wavefront?

- Skupina vláken (podskupina pracovní skupiny) která patří do jedné pracovní skupiny a vykonávají se společně ve stejnou dobu na jednom multiprocesoru.
- Nvidia – *warp*, AMD – *wavefront*
- Typická velikost 32 nebo 64

21 Co je to multiprocesor (streaming multiprocessor/compute unit) GPU a k čemu slouží.

- Jsou to hlavni výpočetní jednotky na GPU které vykonávají kód warpu
- Nvidia – *streaming multiprocessor*, AMD – *compute unit*
- akce běžící na různých MP jsou na sobě nezávislé
- na jednom MP může běžet víc warpů, pokud na to stačí zdroje
- MP má plánovač, který řeší vyhodnocování warpů
- na každý MP lze naalokovat až tisíce vláken a typicky obsahuje až desetitisíce 4 B registrů

22 Co je to fronta příkazů (command queue)?

- fronta, ve které jsou řazeny jednotlivá volání kernelů pro GPU
- OpneGL nemá přímí přístup ke frontě – vytváří se automaticky při vytvoření kontextu
- OpenCL může mít i více front, které manipulují s buffery, spouští kernelů
- Implicitně *in-order* fronta (lze přenastavit)

23 [TODO] Jakým způsobem je možné synchronizovat úlohy ve frontě příkazů s vykonáváním mimo pořadí (out-of-order execution) nebo mezi frontami v OpenCL?

24 [TODO] Jakým způsobem se eliminuje/zmírňuje dopad aritmetických/paměťových latencí.

Pokud nějaké vlákno čeká vlivem latence → je místo něj spuštěno jiné vlákno, pokud máme přístup k dostatečnému množství vláken které lze spustit můžeme kompletně vyblokovat efekty latence.

25 [TODO] Co je to konflikt banků v lokální paměti a kdy vzniká?

26 [TODO] Co je to zarovnaný přístup do paměti?

Přístup k datům postupně jak jsou v paměti – je třeba si je takto přichystat aby se vždy přistupovalo k smysluplným datům -– toto umožňuje lépe pracovat s pamětí — zlepšuje to práci cache a plně se využije načítání větších bloků paměti najednou

27 Popište konstantní/uniformní paměť.

viz 49

28 Popište lokální/sdílenou paměť.

viz 49

29 [TODO] K čemu slouží texturovací jednotky?

Urychleni prace s texturami, obsahuji operace specificke texturam – asi hlavne interpolaci a mozna i nejakou specifickou praci s pameti Řekl bych, že mapují textury na vygenerované polygony.

30 [TODO] Jaké jsou rozdíly mezi bufferem a texturou v OpenCL?

Textury budou nejspise vyuzivat texturovaci jednotky

31 Jaký je rozdíl mezi normalizovanými a nenormalizovanými texturovacími koordináty?

Nenormalizované jsou v rozsahu $[0, x]$, kde x je velikost obrazu.
U normalizovaných $[0, 1]$.

32 Co je to register spilling?

Když existuje vice proměnných než je možné udržet v registrech musí se některé proměnné ukládat do paměti.

33 Jaké jsou způsoby komunikace mezi vlákny ve skupině a mezi skupinami?

V rámci skupiny

1. zápis do lokální nebo globální paměti
2. synchronizace
3. čtení z lokální nebo globální paměti

Omezení při současném zápisu do stejné proměnné z více vláken – potřeba atomických instrukcí

Jednosměrná komunikace mezi skupinami

- pomocí globální paměti
- není zaručeno pořadí vyhodnocení pracovních skupin
- při současném zápisu do stejné proměnné z více skupin – potřeba atomických instrukcí

34 K čemu slouží atomické instrukce? Popište jejich vlastnosti a příklad algoritmu kde je možné je využít.

K výlučnému přístupu k daným datům a zabráněním WAR, WAW a podobných paměťových chyb.

Použití třeba u paralelní redukce (suma, min, max, ...) kdy všechny vlákna chtějí přičíst svou hodnotu k celkovému výsledku, ale bez výlučného přístupu jeden udělá `sum +=` y_1 a než zapíše, tak jiný udělá své `sum +=` y_2 a změní původní hodnotu a opožděný první mu ji přepíše a tím se znehodnotí výsledek.

35 Jaké mohou být příčiny nízké výkonnosti kernelů?

- slabá obsazenost (occupancy)
- aritmetické latence - RAW konflikty
- divergence vláken - větvení, cykly
- paměťové latence
- paměťová propustnosti globální/lokální paměti

36 [TODO] Co je to obsazenost multiprocesoru (occupancy), na čem je závislá?

Vytíženost multiprocessoru – čím větší tím lepší, závisí na divergenci vláken a na správném rozložení pracovních skupin – Pro maximální rychlost výpočtů má smysl co největší obsazenost, ale když vlákna nemakají (není pro ně práce) je vhodné uvolnit místo pro jiné, ať se nebrzdí výpočet jiných workgroup a podobně. Vhodné je taky uvažovat nad rozmístěním vláken v rámci multiprocesoru. Něco víc k tomu? Asi jo možná? -> GMU 06 -> 4 Obsazenost multiprocesoru (occupancy) % naalokovanych vláken z maximálního počtu alokovatelnych vláken na multiprocesor čím větší % tím větší množství aktivních warpu / multiprocesor -> lepší pokrytí aritmetických / paměťových latencí vlákna se alokují po celých skupinách Vliv na obsazenost má: spotřeba lokální pameti/skupinu -> maximální velikost na nových GPU typicky 64kB – 96kb/multiprocesor spotřeba registru/vlakno -> maximální velikost na nových GPU typicky 64k/multiprocesor velikost skupiny -> na nových GPU lze naalokovat maximálně 16 - 32 skupin příliš malé množství vlaken/skupinu -> limitace maximálním množstvím skupin (výjimkou je 64 vlaken/skupinu u AMD GPU -> možnost 100% zaplnění) příliš velké skupiny -> velký dopad synchronizačních funkcí, slabá granularita alokace skupin Výpočet spotřeby: NVIDIA occupancy calculator

37 [TODO] Co je to aritmetická a paměťová latence?

Doba čekání na dokončení paměťových operací a aritmetických operací – načítání dat zpravidla trvá více než jeden takt, podobně tomu bývá u dělení a jiných složitých aritmetických operací

Aritmetická latence

- způsobena čtením právě zapsané hodnoty (RAW - read after write konflikt)
- u moderních GPU typicky 10 – 20 cyklů
- řešení přeskupením instrukcí – automaticky pomocí kompilátoru
- řešení zpracováním více elementů/thread – stoupá spotřeba registrů, klesá potřeba aktivních warpů na multiprocesor

38 [TODO] Co je to separabilní filtr a jakým způsobem lze přistupovat k jeho paralelizaci?

GMU 06, p. 22

39 K čemu slouží paralelní redukce?

Paralelizace cyklu se závislostí na předešlé iteraci při operaci která je asociativní – jde ze složitosti $O(n)$ na $O(log(n))$

40 Co je to dynamický paralelismus?

Počet aktivních vláken se mění během výpočtu (došla práce pro některé vlákna, tak je zruším).

41 [TODO] Jaké jsou možné typy přenosů dat mezi CPU a GPU?

42 K čemu slouží P2P přístup a jakým způsobem ho lze využít.

Přímý přístup k paměti, používá se hlavně ke zvýšení propustnosti paměti.

43 [TODO] K čemu slouží operace shuffle?

44 Vyjmenujte alespoň 3 api pro GP-GPU.

VulkanRT, CUDA, OpenGL, DirectCompute, OpenCL

45 Co to je SpirV?

Softwarový nástroj překládající zdrojové kódy různých platforem (OpenGL, CUDA, ...) na jednotnou nízkoúrovňovou API. Cílem je zjednodušit vývoj a umožnit použití všeho na všem.

46 [TODO] K čemu slouží operace ballot?

47 [TODO] K čemu lze využít atomicCompSwap?

48 [TODO] Jaký je vztah mezi NDRange/Grid/Dispatch, pracovní skupinou a vláknem/invokací?

49 Jaké paměti jsou na GPU a popište jejich vlastnosti (odkud je lze plnit, odkud je lze číst, relativní rychlost, relativní velikost).

Typ	Umístšní	Rychlost	Cache	Velikost	Dostupnost
global	off-chip	0,11 B/tick	1	3 072 MB	vše
local	on-chip	1,33 B/tick	0	0,72 MB	skupina
constant	off-chip	4 B/tick	1	64 kB	vše
register	on-chip	16 B/tick	0	4 MB	vlákno

Globální paměť

- off-chip paměť operační paměť grafického adaptéru
- velikost v řádech GB
- latence v řádu stovek cyklů → nutnost běhu mnoha warpůpro překrytí latence (multithreading)
- rychlost o řád nižší než on-chip paměti (lokální paměti, registry)
- u novějších architektur cachováno pomocí L2 cache, případně i L1 cache → zmírnění dopadu opakovaného načítání stejných dat
- paměť rozdělena po segmentech
- pokud je na jednu adresu ze segmentu přistoupeno z warpu přečte se celý segment
- paměťové segmenty se pohybují mezi 32 B a 128 B
- u starších architektur nezarovnaný přístup - serializace

Konstantní paměť

- lze ji plnit a číst z aplikace
- maximální velikost bufferu dle specifikace 64 kb
- optimalizace na broadcast → přístup do jiných lokací v rámci warpu → serializace
- cachované pomocí konstatní cache
- omezený počet bufferů na kernel – minimálně 8 dle specifikace
- cachování ve speciální cache multiprocesoru
- použití: read-only konstanty používané všemi vlákny

Lokální paměť

- Sdílená v rámci work-group
- Umístěná přímo na čipu – větší propustnost a menší latence než globální paměť (často se před výpočtem plní z globální paměti)
- minimální velikost je pro OpenCL 1.0 16 kB, u novějšího 32 kB
- velikost na předchozích generacích GPU typicky 32 – 48 kB, na aktuální generaci GPU typicky 64 – 96 kB
- u CPU a některých starších GPU emulovaná pomocí globální paměti – pomalé, nepoužívat
- rozdělena do banků s velikostí buňky typicky 4 B
- přístupu do stejného banku více vlákny z warpu – serializace
- novější GPU umějí broadcast a multicast – přístup z více vláken z warpu na stejnou adresu nezpůsobuje bank konflikty
- použití: kooperace mezi vlákny, uživatelské cachování

Texturovací paměť

- data umístěna v globální paměti
- 1D až 3D textury
- data prochází přes texturovací jednotky → interpolace, data mimo rozsah, různé adresování
- adresování
  - normalizované → koordináty staženy do rozsahu $< 0{,}0, 1{,}0 >$
  - nenormalizované → koordináty v rozsahu $< 0{,}0, size >$, první texel na pozici 0,5 0,5
- interpolace
  - nejbližší soused → zaokrouhlení na nejbližší texel
  - lineární → lineární přechod mezi sousedními texely
- řešení okrajů
  - opakování obrazu
  - okrajová hodnota
  - zrcadlení
- optimalizace na načítání bloků
- cachování i na starých architekturách, na nových i do L1 cache multiprocesoru

Registry

- nejrychlejší paměť
- přístup do registrů bez latence
- typicky desetitisíce 4 B registrů na multiprocesor
- pro každé vlákno je alokován potřebný počet registrů – při příliš velké spotřebě omezuje obsazenost
- neblahý vliv na spotřebu registrů může mít rozbalování smyček
- příliš mnoho registrů na vlákno → register spilling

50 Jaký je rozdíl mezi pracovní skupinou a warpem/wavefrontem.

Oboje jsou skupiny vláken, které spolu mohou komunikovat – běží na stejné výpočetní jednotce/streaming multiprocessoru

Pracovní skupina

- uživatelem definovaná skupina vláken (work-group)
- může obsahovat vice wavefrontů — je jim nadřazená
- různé wavefronty mohou vykonávat různé instrukce – synchronizovat je nutné explicitně

warp/wavefront

- pro všechna vlákna se vykonává stejná instrukce (při divergency některé ale nemusí pracovat)

51 K čemu slouží příkaz barrier? Jakou má souvislost s rozdělením vláken do skupin, warpů? Jaký to má vztah k větvení programu?

- Slouží k synchronizaci vláken
- Vlákna můžeme synchronizovat pouze v rámci skupiny (work-groupy)
- Synchronizuje se pomocí příkazu `barrier`
- Všechna vlákna ve skupině jej musí zavolat, než kterékoliv může pokračovat
- Pokud je `barrier` v podmínce, musí ji navštívit všechny vlákna ve skupině nebo žádné
- V případě, kdyby jedno vlákno vykonávalo jinou *control flow* cestu, ostatní vlákna by zůstaly čekat na `barrier` donekonečna