KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Zaawansowane zastosowania kart graficznych	**1010512321010519522**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**ogólnoakademicki**	**1 / 2**
Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
Technologie przetwarzania danych	**polski**	**obieralny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
II stopień	**niestacjonarna**

Godziny	Liczba punktów
Wykłady: **16** Ćwiczenia: **-** Laboratoria: **16** Projekty/seminaria: **-**	**4**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
kierunkowy	**z danego kierunku**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**4 100%**
nauki techniczne	**4 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Witold Andrzejewski
email: Witold.Andrzejewski@cs.put.poznan.pl
tel. 61 6652965
Instytut Informatyki
ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę o architekturze komputerów, o programowaniu obiektowym i proceduralnym oraz o metodach oceny złożoności algorytmów.
2	**Umiejętności:**	Powinien posiadać umiejętność rozwiązywania podstawowych problemów algorytmicznych, programowania w jezyku C/C++, oraz umiejętność pozyskiwania informacji ze wskazanych źródeł.
3	**Kompetencje społeczne**	Powinien również rozumieć konieczność poszerzania swoich kompetencji i mieć gotowość do podjęcia współpracy w ramach zespołu. Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu:

1. Przekazanie studentom podstawowej wiedzy z programowania procesorów kart graficznych (GPU), w zakresie:
   a. Wybranych aspektów architektury sprzętowej popularnych kart graficznych.
   b. Logicznego modelu współbieżności kart graficznych w kontekście API CUDA i OpenCL.
   c. Optymalizacji wykonania programów na kartach graficznych.
   d. Wybranych podstawowych algorytmów równoległych, w tym: map, reduce, compact, sort, scan, search, generowania ntych wyrazów ciągów.
   e. Wybranych struktur danych, w tym: macierzy, tablic hashowych i drzew CSS i algorytmów ich przetwarzania.
   f. Oceny złożoności algorytmów równoległych (w tym model PRAM) i ich związku z faktyczną złożonością programów wykorzystujących karty graficzne.
   g. Zastosowania kart graficznych do rozwiązywania praktycznych problemów związanych z przetwarzaniem i wizualizacją danych.
2. Rozwijanie u studentów umiejętności:
   a. Rozwiązywania problemów algorytmicznych z ukierunkowaniem na zrównoleglenie operacji przetwarzania danych.
   b. Optymalizacji programów wykorzystujących procesory kart graficznych.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. ma zaawansowaną i pogłębioną wiedzę z zakresu architektury kart graficznych - [K2st_W1]

2. ma zaawansowaną wiedzę szczegółową dotyczącą tworzenia algorytmów równoległych na procesory kart graficznych - [K2st_W3]

3. ma zaawansowaną i szczegółową wiedzę o działaniu procesorów kart graficznych - [K2st_W5]

4. zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu złożonych zadań inżynierskich związanych z projektowaniem i optymalizacją niskopoziomową programów wykorzystujących procesory kart graficznych. - [K2st_W6]

Umiejętności:

1. potrafi pozyskiwać informacje z literatury, baz danych oraz innych źródeł (w języku polskim i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie - [K2st_U1]

2. potrafi ? przy formułowaniu i rozwiązywaniu zadań inżynierskich ? integrować wiedzę z różnych obszarów informatyki (a w razie potrzeby także wiedzę z innych dyscyplin naukowych) oraz zastosować podejście systemowe, uwzględniające także aspekty pozatechniczne - [K2st_U5]

3. potrafi ocenić przydatność i możliwość wykorzystania nowych osiągnięć (metod i narzędzi) oraz nowych produktów informatycznych - [K2st_U6]

4. potrafi - stosując m.in. koncepcyjnie nowe metody - rozwiązywać złożone zadania informatyczne, w tym zadania nietypowe oraz zadania zawierające komponent badawczy - [K2st_U10]

5. potrafi współdziałać w zespole, przyjmując w nim różne role - [K2st_U15]

6. potrafi określić kierunki dalszego uczenia się i zrealizować proces samokształcenia, w tym innych osób - [K2st_U16]

Kompetencje społeczne:

1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe - [K2st_K1]

2. zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych - [K2st_K2]

Sposoby sprawdzenia efektów kształcenia

Ocena formująca

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach

b) w zakresie laboratoriów / ćwiczeń weryfikowanie założonych efektów kształcenia realizowane jest przez: okresową ocenę przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych (sprawdzian wejściowy) oraz ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych

Ocena podsumowująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez ocenę wiedzy i umiejętności wykazanych na zaliczeniu pisemnym o formie testu jednokrotnego wyboru składającego się z ok. 30 pytań, łączna liczba punktów za prawidłowe odpowiedzi: 30, minimalna liczba punktów umożliwiających zaliczenie: 16

b) w zakresie laboratoriów / ćwiczeń weryfikowanie założonych efektów kształcenia realizowane jest przez ocenę i obronę przez studenta sprawozdania z realizacji projektu.

Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:

- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,

- uwagi związane z udoskonaleniem materiałów dydaktycznych,

- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.

Treści programowe

Program wykładu obejmuje następujące zagadnienia:

1. Motywacja stojącą za wykorzystaniem kart graficznych do obliczeń. Dodatkowo: przedstawienie różnych rozwiązań technologicznych umożliwiających przetwarzanie równoległe, wprowadzenie podstawowych pojęć stosowanych w kolejnych wykładach. Omówienie podstaw modelu programistycznego wykorzystywanego w programach wykorzystujących procesory kart graficznych.

2. Architekturę sprzętowa kart graficznych. Omówienie związków pomiędzy architekturą sprzętową a modelem programistycznym. Przedstawienie trików bitowych.

3. Omówienie hierarchii pamięci i metod wydajnego dostępu dla różnych architektur sprzętowych. Przedstawienie przykładowych metod optymalizacji wykorzystujących różne poziomy pamięci operacyjnej. Omówienie rozwiązań pozwalających na zapewnienie równoległego przesyłu danych pomiędzy komputerem a kartą graficzną i wykonywania obliczeń.

4. Wprowadzenie podstaw teoretycznych oceny złożoności algorytmów równoległych. Omówienie obliczeń w modelu maszyny PRAM. Podstawowe algorytmy wykorzystywane podczas konstrukcji bardziej złożonych rozwiązań, w tym: map, gather, scatter, reduce, compact, search, scan. Szczegółowe omówienie algorytmu scan. Ocena złożoności wprowadzonych algorytmów w ich wersji sekwencyjnej i równoległej.

5. Szczegółowe omówienie algorytmów : compact i reduce oraz algorytmów przeszukiwania i sortowania danych. Ocena złożoności przedstawionych algorytmów.

6. Algorytmy generowania elementów w sekwencji kombinacji, permutacji bądź w wyniku iloczynu kartezjańskiego wraz z ich przykładowymi zastosowaniami.

7. Problemy związane z wykorzystywaniem standardowych struktur danych. Omówienie wydajnych metod przetwarzania macierzy (w tym macierzy rzadkich) oraz tablic haszowych. Oszacowanie złożoności algorytmów przetwarzania tych struktur.

8. Praktyczne zastosowania GPU do przetwarzania, eksploracji i wizualizacji danych.

Ćwiczenia realizowane są przez studentów samodzielnie Program laboratorium obejmuje następujące zagadnienia:

1. Zapoznanie się z CUDA.

2. Ćwiczenia związane z prawidłową konstrukcją siatki obliczeń.

3. Implementacja prostych algorytmów równoległych.

4. Omówienie sposobów debugowania programów wykorzystujących karty graficzne.

5. Realizacja ćwiczeń w których należy uwzględnić komunikację i synchronizację wątków.

6. Testowanie wydajności dostępu do różnych dostępnych rodzajów pamięci operacyjnej na karcie graficznej.

7. Zapoznanie się z biblioteką thrust. Omówienie podstawowych koncepcji stojących za API tej biblioteki. Przedstawienie podstawowych algorytmów w niej zaimplementowanych.

8. Omówienie biblioteki CURAND. Ćwiczenia wykorzystujące bibliotekę thrust. Budowa złożonych algorytmów z podstawowych algorytmów takich jak: map, reduce, gather, scatter, search, scan.

Cześć wymienionych wyżej treści programowych realizowana jest w ramach pracy własnej studenta.

Metody dydaktyczne:

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań.

2. ćwiczenia laboratoryjne: rozwiązywanie zadań, ćwiczenia praktyczne, pokaz multimedialny, demonstracja.

Literatura podstawowa:

1. Cuda w przykładach : wprowadzenie do ogólnego programowania procesorów GPU / Jason Sanders, Edward Kandrot

2. OpenCL : akceleracja GPU w praktyce / Marek Sawerwain.

3. NVIDIA: CUDA C Programming Guide: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

4. NVIDIA: CUDA C Best Practices Guide: http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html

5. NVIDIA: OpenCL Jumpstart Guide: http://www.cs.cmu.edu/afs/cs/academic/class/15668-s11/www/cuda-doc/OpenCL_Jumpstart_Guide.pdf

6. NVIDIA: OpenCL Best Practices Guide: http://www.nvidia.com/content/cudazone/CUDABrowser/downloads/papers/NVIDIA_OpenCL_BestPracticesGuide.pdf

Literatura uzupełniająca:

1. Jianlong Zhong* and Bingsheng He. Medusa: Simplified Graph Processing on GPUs. Accepted by TPDS 2013: IEEE Transactions on Parallel and Distributed System

2. Bingsheng He and Jeffrey Xu Yu. High-Throughput Transaction Executions on Graphics Processors. Proceedings of Very Large Data Bases (VLDB) 2011

3. Andrzejewski, Witold; Boinski, Pawel Efficient spatial co-location pattern mining on multiple GPUs Journal Article Expert Systems with Applications, 93 (Supplement C), pp. 465?483, 2018, ISBN: 0957-4174.

4. Andrzejewski, Witold; Boinski, Pawel Parallel GPU-based Plane-sweep Algorithm for Construction of iCPI-trees Journal Article Journal of Database Management, 26 (3), pp. 1-20, 2015, ISSN: 1063-8016.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. udział w zajęciach laboratoryjnych	16
2. przygotowanie do ćwiczeń laboratoryjnych	16
3. udział w konsultacjach związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych / projektu	4
4. przygotowanie do sprawdzianów / kolokwium	16
5. udział w wykładach	16
6. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi	16
7. przygotowanie do zaliczenia wykładów	

Obciążenie pracą studenta		
forma aktywności	**godzin**	**ECTS**
Łączny nakład pracy	100	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	36	2
Zajęcia o charakterze praktycznym	32	2