AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Module name:	Multicore programming				
Academic year:	2017/2018	Code:	JFT-1-007-s	ECTS credits:	4
Faculty of:	Physics and Applied Computer Science				
Field of study:	Technical Physics	Specialty:	—		
Study level:	First-cycle studies	Form and type of study:	Full-time studies		
Lecture language:	English	Profile of education:	Academic (A)	Semester:	0
Course homepage:	—				
Responsible teacher:	dr hab. inż, prof. AGH Szumlak Tomasz (szumlak@agh.edu.pl)				
Academic teachers:	dr hab. inż, prof. AGH Szumlak Tomasz (szumlak@agh.edu.pl)				

Module summary

Recent advances in processor technology clearly show a major push towards increase the number of processing cores (threads) on a single physical chip. Learning the basic programming skills pertaining to concurrent programming is vital for each software developer.

Description of learning outcomes for module

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Connections with FLO	Method of learning outcomes verification (form of completion)
Social competence			
M_K001	A student can comunicate his/her results and discuss them	FT1A_K03, FT1A_K01	Project, Participation in a discussion
Skills			
M_U001	A student understands the subtle interactions and interlinks between operating system, hardware, specialised drivers and software development tools.	FT1A_U05, FT1A_U01	Project, Execution of laboratory classes
M_U002	A student can work as a part of a team and can interact properly with his/her co-workers	FT1A_U09, FT1A_U08	Project, Participation in a discussion, Execution of laboratory classes
Knowledge			
M_W001	A student gains basic knowledge on modern CPU and GPU multicore architectures and the ways this technological progress can be used to speed up data processing.	FT1A_W01	Activity during classes, Examination, Project

FLO matrix in relation to forms of classes

MLO code	Student after module completion has the knowledge/ knows how to/is able to	Form of classes										
		Lectures	Auditorium classes	Laboratory classes	Project classes	Conversation seminar	Seminar classes	Practical classes	Fieldwork classes	Workshops	Others	E-learning
Social competence												
M_K001	A student can comunicate his/her results and discuss them	-	-	+	-	-	-	-	-	-	-	-
Skills												
M_U001	A student understands the subtle interactions and interlinks between operating system, hardware, specialised drivers and software development tools.	-	-	+	+	-	-	-	-	-	-	-
M_U002	A student can work as a part of a team and can interact properly with his/her co-workers	-	-	+	+	-	-	-	-	-	-	-
Knowledge												
M_W001	A student gains basic knowledge on modern CPU and GPU multicore architectures and the ways this technological progress can be used to speed up data processing.	+	-	-	-	-	-	-	-	-	-	-

Module content

Lectures

Introduction

Recent advances in microelectronics gave us a unique opportunity to operate CPU or GPU chips containing many processors – called cores. It is only natural that software should follow the same way and provide tools for development of applications that can properly use this potential. The first two lectures will be devoted to a generic discussion on how to harness the power of multicore architecture, we learn about Amdahl and Gustafson’s laws, look at different computing platforms and discuss modern processors design. Also processes and threads will be introduced along as well.

What language should I use if I want to go parallel?

Here we discuss writing parallel programs using commonly known programming languages such as C, C++ and Python. What is needed to turn an ordinary mundane sequential code into nice and shiny parallel applications.

OpenCL (Khronos) platform

OpenCL stands for ‘Open Computing Language’ and is a software platform that facilitates writing code that can be executed on hybrid hardware platforms consisting of both CPUs and GPUs. The main features of the framework will be discussed during this lecture.

CUDA (NVIDIA) platform

CUDA framework is similar in many ways to the OpenCL one. The main difference come from significant specialization that allows to run CUDA applications only on CUDA enabled hardware (such as NVIDIA GPUs). This results in less flexibility regarding the available hardware but on the other hand allows to write software that uses better the available hardware resources. Again the main features of this platform will be presented together with an example of drivers and SDK (Software Development Kit) installation.

Laboratory classes

Introduction

This part is devoted to show a student what is needed to prepare an parallel application. We discuss subtleties of quite complicated interaction between the CUDA enabled hardware and operating system running the appropriate CUDA drivers. Also, the NVIDIA SDK environment will be demonstrated.

CUDA toolkit: hands-on

How to write CUDA ‘Hello Word’. Code for a host (i.e., CPU and system memory) and that written for a device (i.e., GPU and its memory). Executing device code from a host. A device memory management.

Writing parallel code

Basic ways to use CUDA for writing parallel applications. Using threads and how to make them talking to each other. Threads synchronisation.

Memory management - essentials

CUDA and the constant memory (CM). CUDA events and application performance measurements.

Complex example

Using the skills and knowledge gained so far we try to write a more complex application – heat transfer simulation.

Project classes

Projects

A CUDA related project will be provided for a student. It is also possible to hand our a project that can reflect specific interest of an individual (i.e., related to his/her Master thesis)

Method of calculating the final grade

The final mark will be determined by observing the general rules set by the AGH University
The final mark (FM) E – exam, L – computer lab, P – project
FM = 0,5 x E + 0,2 x L + 0,3 x P

Prerequisites and additional requirements

- basic knowledge of Python is an asset but not compulsory to take this course

Recommended literature and teaching resources

- web resources: nvidia.com, http://www.khronos.org/opencl/
- books: NOTE! usually become obsolete quite fast

1) Professional Multicore Programming: Design and Implementation for C++ Developers
Cameron Hughes, Tracey Hughes, ISBN: 978-0-470-28962-4
code samples available for download!
2) Fundamentals of Multicore Software Development
Victor Pankratius, Ali-Reza Adl-Tabatabai, Walter Tichy
ISBN-10: 143981273X

Scientific publications of module course instructors related to the topic of the module

Additional scientific publications not specified

Additional information

Absences on lab classes and tutorials will have to be negotiated with the tutor.

Student workload (ECTS credits balance)

Student activity form	Student workload
Participation in lectures	0 h
Realization of independently performed tasks	31 h
Participation in laboratory classes	18 h
Preparation for classes	20 h
Participation in project classes	10 h
Contact hours	20 h
Examination or Final test	1 h
Summary student workload	100 h
Module ECTS credits	4 ECTS