1.3. Module/ course form

To be completed by Course Team							
	Module name : **PARALLEL AND DISTRIBUTED PROCESSING**					Module code: M7	
	Course name: **Parallel and Distributed Processing**					Course code: M7-5	
	Faculty: **Institute of Applied Informatics**						
	Field of study: **INFORMATICS**						
	Mode of study : **stationary**		Learning profile: **PRACTICAL**			Speciality: BSc	
	Year/ semester: **2/4**		Module/ course status: **MANDATORY**			Module/ course language: **POLISH/ENGLISH**	
	Type of classes	lecture	lessons	lab	project	tutorial	other (please specify)
	Course load	**15**		**30**			

Module/ course coordinator	**Zdzisław Szczerbiński, PhD Eng.**
Lecturer	**Zdzisław Szczerbiński, PhD Eng.**
Module/ course objectives	1. Teaching basic principles of parallel architectures and software. 2. Teaching main principles of parallel programming.
Entry requirements	1. Attestation of the course Basics of Programming I (semester I). 2. Capability of classical, sequential programming in a procedural language.

LEARNING OUTCOME		
No.	LEARNING OUTCOME DESCRIPTION	Learning outcome reference
01	knows how to construct and operate current parallel systems (multiprocessor/manycore computers, clusters)	K_W05, K_W08, K_W09
02	knows the basic programming constructs associated with parallel programming.	K_W07
03	is familiar with the obstacles to non-problem program parallelization.	K_W15
04	classifies existing parallel architectures and draws conclusions from theorems pertaining to effective use of these systems.	K_U06
05	writes, debugs and tests parallel programs in a shared-memory programming environment	K_U16
06	codes simple parallel algorithms	K_U17

07	is aware of the necessity of constantly updating own knowledge of the fastest developing directions in technology (informatics) and their novel branches (parallel and distributed processing)	K_U06

CURRICULUM CONTENTS

Lecture

Introduction to parallel computers, programming and computation.
Classification of computer architectures relating to the control mechanism and address space organization.
Shared memory multiprocessor systems. Types of processor-memory connections.
Distributed memory multiprocessor systems. Structure of the interprocessor network. Static and dynamic networks.
Computer clusters and network computing systems.
Efficiency measures for parallel computing. Amdahl's law.
SIMD algorithms. Illustration of SIMD operation in a system of serially connected processors.
MIMD algorithms. Illustration of MIMD operation in a system of processors connected as a binary tree.
Interprocessor communication and synchronization by means of shared variables.
The busy-waiting method. Semaphores.
Concurrent processes. Controlling process access to a common resource with semaphores.
Processes in a deadlock. Process starvation. The dining philosophers problem.
Applying semaphores to event synchronization.
Example of using general semaphores: the producer-consumer problem.
Example of using binary semaphores: the bear and bees problem.
Interprocessor communication and synchronization by message passing.

Lab

The lab's main objective is to teach the ideas and methods of parallel programming. The hardware platform is a dual-processor SunFire system, with SunBlade computers as the students' interactive terminals. The software platform is the C programming language augmented with parallelizing directives of the OpenMP standard. During the classes, consecutive programs are discussed which enable to understand the basics of programming shared-memory multiprocessor systems. After a whiteboard introduction and reading the program's text, the students individually compile and run the program. Afterwards, they analize the program's output and make relevant notes. The students have constant access, via the Internet, to the example programs which is beneficial to their mastering the practice of parallel programming.
In the concluding part of the semester, the students get acquainted with the principles of vector programming and the basics of parallel programming in the languages Python and Julia.

Basic literature	1. A. Karbowski, E. Niewiadomska-Szynkiewicz (eds.) – *Parallel and Distributed Programming*, Oficyna Wydawnicza Politechniki Warszawskiej, 2009 (in Polish). 2. Z. Weiss, T. Gruźlewski – *Concurrent and Distributed Programming: Examples and Problems*, WNT, 1993 (in Polish).
Additional literature	1. S. Kozielski, Z. Szczerbiński – *Parallel Computers: Architecture, Elements of Programming*, WNT, 1994 (in Polish). 2. B. E. Borowik – *Parallel Programming in Applications*, MIKOM, 2001 (in Polish).

Teaching methods	lecture, labs

Assessment method	Learning outcome number
written exam (60 minutes) spanning theoretical (4 problems) and practical (2 problems) issues	01, 02, 04
individual solution of lab problems in parallel programming	03, 05, 06
discussion on the previously elaborated topics, underlining the knowledge of the newest trends in computer architecture and parallel programming	07

Form and terms of an exam	Final exam, written, spanning the instruction material for lectures and labs. Percentage in the exam mark: lecture – 70%, lab – 30%. The final mark is additionally affected by high activity and infrequent presence at the lab classes.

STUDENT WORKLOAD		
	Number of hours	
	In all	including practical
Participation in lectures	15	
Independent study of lecture topics	10	
Participation in tutorials, labs, projects and seminars	30	21
Independent preparation for tutorials*	45	31
Preparation of projects/essays/etc.*		
Preparation/ independent study for exams	20	5
Participation during consultation hours	3	
Other - exam	2	
TOTAL student workload in hours	**125**	**57**
Number of ECTS credit per course unit	**5 ECTS**	
Number of ECTS points assigned to the scientific discipline	Technical informatics and telecommunications **5 ECTS**	
Number of ECTS credit associated with practical classes	**2,3 ECTS**	
Number of ECTS for classes that require direct participation of professors	**2 ECTS**	