SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeş Bolyai University**
1.2 Faculty	**Faculty of Mathematics and Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study programme / Qualification	**Computer Science - english**

2. Information regarding the discipline

2.1 Name of the discipline		**Computer System Architecture**					
2.2 Course coordinator		**Lect. Dr. Vancea Alexandru-Ioan**					
2.3 Seminar coordinator		**Lect. Dr. Vancea Alexandru-Ioan**					
2.4. Year of study	**1**	2.5 Semester	**1**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Compulsory**

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	5	Of which: 3.2 course	2	3.3 seminar/laboratory	1 sem + 2 lab
3.4 Total hours in the curriculum	70	Of which: 3.5 course	28	3.6 seminar/laboratory	42
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					20
Additional documentation (in libraries, on electronic platforms, field documentation)					10
Preparation for seminars/labs, homework, papers, portfolios and essays					20
Tutorship					10
Evaluations					20
Other activities:					

3.7 Total individual study hours	80
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	•
4.2. competencies	•

5. Conditions (if necessary)

5.1. for the course	•
5.2. for the seminar /lab	• Laboratory with computers

activities	

6. Specific competencies acquired

Professional competencies	• Knowledge, understanding and use of basic concepts of theoretical Computer Science
Transversal competencies	• Abilities of a full control of a computing system's resources and reaching the skills of optimizing the programs developed in high level programming languages.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	• Knowledge of the computer architecture models, processor functioning, computer information representation usage
7.2 Specific objective of the discipline	• Understanding by the students of the computer architecture models, processor functioning, computer information representation usage • Initiation in assembler language programming, which will assure the comprehension of the microprocessor architecture and functioning • Understanding the basic functions of a computer's architectural components and its native low-level workflow. Awareness of the architectural impact on designing and implementing high level programming languages. • Initiation in interrupt systems architecture, with the 80x86 case study

8. Content

8.1 Course	Teaching methods	Remarks
Data representation: elementary data, binary representation and placement orders, data organizing and storing (W1), character coding, signed and unsigned representation, complementary code, conversions, the concept of overflow (W2); Computing systems (CS) architecture: organization of a CS, the central processing unit, the system clock, computer on n bits, the storage, peripheral devices (W3), CS performances, the 80x86 microprocessor's architecture – structure, registers, address computation, addressing modes, far addresses and near addresses (W4);	Exposure, description, explanation, examples, discussion of case studies	

Assembly language elements: the source line format, expressions, accessing the operands, operators (W5), directives for defining the segments, for defining data, LABEL, EQU, PROC, INCLUDE, repetitive blocks and macros (W6); Assembly language instructions: transfer instructions, conversions, signed and unsigned arithmetic operations, bitwise shifting and rotating, logical bitwise operations (W7), conditional and unconditional jump instructions, looping instructions, string instructions (W8); Interrupts: classification, specific instructions working with interrupts, the COM and EXE formats (W9) ; Interrupts redirection: TSR programs, installing and deinstalling TSR programs, debugging a TSR program, interrupts redirection under Windows OS (W10); Subprograms call implementation and multimodule programming: call code, entry code, exit code, the directives PUBLIC, EXTRN, GLOBAL, linking TASM modules with modules written in high-level programming languages (W11); Low-level programming in high level programming languages: inserting machine code, inline assemblers, assembler procedures and functions, accessing registers and calling interrupts, interrupt procedures and functions (W12); x86 extensions: protected mode, architectural extensions and new instructions added during the evolution of the 80x86 family of processors (W13); Assembly programming under Windows: system calls in protected mode, restrictions imposed on the interrupt system, MASM and NASM assemblers, the Visual C++ inline assembler (W14);		

Bibliography

1. Al. Vancea, F. Boian, D. Bufnea, A. Gog, A. Darabant, A. Sabau – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2005.

2. A. Gog, A. Sabau, D. Bufnea, A. Sterca, A. Darabant, Al. Vancea – Programarea în limbaj de asamblare 80x86. Exemple si aplicatii., Editura Risoprint, Cluj-Napoca, 2005.

3. Randal Hyde – The Art of Assembly Programming, No Starch Press, 2003. (http://homepage.mac.com/randyhyde/webster.cs.ucr.edu/www.artofasm.com/DOS/index.html)

4. Boian F. M. Sisteme de operare interactive. Ed. Libris, Cluj, 1994

5. Boian F. M. De la aritmetica la calculatoare. Ed. Presa Universitara Clujeana, Cluj, 1996

6. Boian F. M., Vancea A., Iurian S., Iurian M. Programare avansata de sistem si aplicatii IBM-PC, lito. Universitatea "Babes-Bolyai", 1996

7. Boian F.M. Vancea A. Arhitectura calculatoarelor, suport de curs. Facultatea de Matematica si Informatica, Centrul de Formare Continua si Invatamânt la Distanta,. Ed. Centrului de Formare Continua si Invatamânt la Distanta, Cluj, 2002,

8. Knuth D.E. Tratat de programarea calculatoarelor; vol 3: Algoritmi seminumerici. Ed. Tehnica, Bucuresti, 1985

8.2 Seminar and laboratory	Teaching methods	Remarks
Data representation: elementary data, binary representation and placement orders, data organizing and storing (W1), character coding, signed and unsigned representation, complementary code, conversions, the concept of overflow (W2); (seminar weeks W1/W2); Computing systems (CS) architecture: organization of a CS, the central processing unit, the system clock, computer on n bits, the storage, peripheral devices (W3), CS performances, the 80x86 microprocessor's architecture – structure, registers, address computation, addressing modes, far addresses and near addresses (W4); (seminar weeks W3/W4); Assembly language elements: the source line format, expressions, accessing the operands, operators (W5), directives for defining the segments, for defining data, LABEL, EQU, PROC, INCLUDE, repetitive blocks and macros (W6); (seminar weeks W5/W6); Assembly language instructions: transfer instructions, conversions, signed and unsigned arithmetic operations, bitwise shifting and rotating, logical bitwise operations (W7), conditional and unconditional jump instructions, looping instructions, string instructions (W8); (seminar weeks W7/W8); Interrupts: classification, specific instructions working with interrupts, the COM and EXE formats (W9) ; Interrupts redirection: TSR programs, installing and deinstalling TSR programs, debugging a TSR program, interrupts redirection under Windows OS (W10); (seminar weeks W9/W10); Subprograms call implementation and multimodule programming: call code, entry code, exit code, the directives PUBLIC, EXTRN, GLOBAL, linking TASM modules with modules written in high-level programming languages (W11);	Exposure, description, explanation, examples, discussion of case studies Practical projects	

Low-level programming in high level programming languages: inserting machine code, inline assemblers, assembler procedures and functions, accessing registers and calling interrupts, interrupt procedures and functions (W12); Topics 6 and 7 will be approached in (seminar weeks W11/W12); x86 extensions: protected mode, architectural extensions and new instructions added during the evolution of the 80x86 family of processors (W13); Assembly programming under Windows: system calls in protected mode, restrictions imposed on the interrupt system, MASM and NASM assemblers, the Visual C++ inline assembler (W14); topics 8 and 9 will be approached in (seminar weeks W13/W14);		

Bibliography

1. Al. Vancea, F. Boian, D. Bufnea, A. Gog, A. Darabant, A. Sabau – Arhitectura calculatoarelor. Limbajul de asamblare 80x86., Editura Risoprint, Cluj-Napoca, 2005.

2. A. Gog, A. Sabau, D. Bufnea, A. Sterca, A. Darabant, Al. Vancea – Programarea în limbaj de asamblare 80x86. Exemple si aplicatii., Editura Risoprint, Cluj-Napoca, 2005.

3. Randal Hyde – The Art of Assembly Programming, No Starch Press, 2003. (http://homepage.mac.com/randyhyde/webster.cs.ucr.edu/www.artofasm.com/DOS/index.html)

4. Boian F. M. Sisteme de operare interactive. Ed. Libris, Cluj, 1994

5. Boian F. M. De la aritmetica la calculatoare. Ed. Presa Universitara Clujeana, Cluj, 1996

6. Boian F. M., Vancea A., Iurian S., Iurian M. Programare avansata de sistem si aplicatii IBM-PC, lito. Universitatea "Babes-Bolyai", 1996

7. Boian F.M. Vancea A. Arhitectura calculatoarelor, suport de curs. Facultatea de Matematica si Informatica, Centrul de Formare Continua si Invatamânt la Distanta,. Ed. Centrului de Formare Continua si Invatamânt la Distanta, Cluj, 2002,

8. Knuth D.E. Tratat de programarea calculatoarelor; vol 3: Algoritmi seminumerici. Ed. Tehnica, Bucuresti, 1985

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

- The course exists in the studying program of all major universities in Romania and abroad;
- The content of the course is considered by the software companies as important for average programming skills

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	- know the basic principle of the domain;	Written exam	60%
	- application of these principles for problem solving		
10.5 Lab/Seminar activities	- implementation in assembly language	Laboratory work	20%
		Practical exam	20%
10.6 Minimum performance standards			
➢ At least grade 5 at written exam, laboratory work and pratical exam.			

Date	Signature of course coordinator	Signature of seminar coordinator
24.04.2013	Lect. Dr. Vancea Alexandru-Ioan	Lect. Dr. Vancea Alexandru-Ioan

Date of approval ..... Signature of the head of department .....….....