|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. Course title: Compilers and assemblers | | | | | |
|  | | | | |
| 2. Code: | | 3. Type (lecture, practice etc.): practice | | | |
|  | | | | |
| 4. Contact hours: 2 hoursper week | | 5. Number of credits (ECTS): 2 | | | |
|  | | | | |
| 6. Preliminary conditions (max. 3): | | | | | |
|  | | | | |
| 7. Announced:fall semester, spring semester, both | | | | | |
|  | | | | |
| 8. Limit for participants: 48 | | | | | |
|  | | | | |
| 10. Responsible teacher (faculty, institute and department):  Gimesi László PhD (Faculty of Science, Institute of Mathematics and Informatics) | | | | | |
|  | | | | |
| 11. Teacher(s) and percentage: | | Makkai Géza PhD | | 100 % | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | | | | |
| 12. Language:English | | | | | |
|  | | | | |
| 13. Course objectives and/or learning outcomes:  Students are introduced to the operation of compilers, different analytical methods. They will have knowledge of low-level programming of processors and microcontrollers. They will know the basics of Assembly programming language, and will be familiar with elementary (binary) algorithms. | | | | | |
|  | | | | |
| 14. Course outline   1. Annunciation of course requirements. Binary arithmetic, number representation. Memory addressing, segment and offset addressing. Processor types, registers, register operations. Structure of Assembly programs. 2. Assembly programming environment. Addressing methods: direct, register, memory, indirect etc. Instruction set: data moving, arithmetic, logic, jump etc. 3. Managing interrupts, subroutine call. Writing characters and strings to console, using keyboard. 4. Examples for using keyboard and console. INT 21h interrupt. 5. Theoretical summary of formal languages and automats, connection between automats and compilers. Summary of programming languages. Introduction to operation of compilers. 6. Declaring data types, variables, constants, and labels. Memory allocation (DATA and DATA?). Examples for using data segments. 7. Structure of compilers. Lexical, syntactic, and semantic analysis. 8. Error handling, code generating, code optimization. 9. Examples: analysis of code of high level language programs. 10. Embedded screen-memory management. 11. Mass storage management, writing its content to console. 12. Connection of Assembly to other programming languages. Compiling directives. 13. Summary, evaluation of course fulfilment. | | | | | |
|  | | | | |
| 15. Mid-semester works   1. Test. 2. Test. 3. Assessment: individual software development. | | | | | |
|  | | | | |
| 16. Course requirements and grading   * 2 tests: 50%, * 2 home works: 50%. | | | | | |
|  | | | | |
| 17. List of readings   1. Torben Ægidius Mogensen: Basics of Compiler Design, University Of Copenhagen, 2010. 2. Peter Norton and John Socha: Assembly language for the PC Prentice Hall Press, 1986. 3. Published tasks in [Neptun Meet Street](javascript:__doPostBack('upChooser$btnKollab','')) | | | | | |
|  | | | | |
| 18. Recommended texts, further readings   1. Grune, D., van Reeuwijk, K., Bal, H.E., Jacobs, C.J.H., Langendoen, K.: Modern Compiler Design, Springer, 2012. 2. Microsoft MASM Programmers’s Guide. | | | | | |
|  | | | | |
| **Date** | 24 April, 2017 | **Prepared by** |  | | |
| Dr. Gimesi László  Responsible teacher | | |
|  | | | | |
| **Endorsed by** | | |  | | |
| Dr. Koniorczyk Mátyás program supervisor | | |