Universidad Nacional Mayor de San Marcos
School of Computer Science
Syllabus of Course
Academic Period 2018-II

1. **Code and Name:** CS3402. Compilers (Mandatory)
2. **Credits:** 4
3. **Hours of theory and Lab:** 2 HT; 4 HL; (15 weeks)
4. **Professor(s)**

   Meetings after coordination with the professor

5. **Bibliography**

[Aho+08] Alfred Aho et al. *Compiladores. Principios, técnicas y herramientas.* 2nd. ISBN:10-970-26-1133-4. Addison Wesley, 2008.

[Aho90] Alfred Aho. *Compiladores Principios, técnicas y herramientas.* Addison Wesley, 1990.

[ALe96] Karen A.Lemone. *Fundamentos de Compiladores.* CECSA-Mexico, 1996.

[App02] A. W. Appel. *Modern compiler implementation in Java.* 2.a edición. Cambridge University Press, 2002.

[Lou04a] Kenneth C. Louden. *Construccion de Compiladores Principios y Practica.* Thomson, 2004.

[Lou04b] Kenneth C. Louden. *Lenguajes de Programacion.* Thomson, 2004.

[PV98] Terrence W. Pratt and Marvin V.Zelkowitz. *Lenguajes de Programacion Diseño e Implementacion.* Prentice-Hall Hispanoamericana S.A., 1998.

[TS98] Bernard Teufel and Stephanie Schmidt. *Fundamentos de Compiladores.* Addison Wesley Iberoamericana, 1998.

6. **Information about the course**

   (a) **Brief description about the course** That the student knows and understands the concepts and fundamental principles of the theory of compilation to realize the construction of a compiler

   (b) **Prerrequisites:** CS2101. Theory of Computation. (4th Sem)

   (c) **Type of Course:** Mandatory

   (d) **Modality:** Face to face

7. **Specific goals of the Course**

   - Know the basic techniques used during the process of intermediate generation, optimization and code generation.
   - Learning to implement small compilers.

8. **Contribution to Outcomes**

a) An ability to apply knowledge of mathematics, science. (**Assessment**)

b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Assessment**)

j) Apply the mathematical basis, principles of algorithms and the theory of Computer Science in the modeling and design of computational systems in such a way as to demonstrate understanding of the equilibrium points involved in the chosen option. (**Assessment**)

a) An ability to apply knowledge of mathematics, science. (**Assessment**)

b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Assessment**)

j) Apply the mathematical basis, principles of algorithms and the theory of Computer Science in the modeling and design of computational systems in such a way as to demonstrate understanding of the equilibrium points involved in the chosen option. (**Assessment**)

9. Competences (IEEE)

C8. Understanding of what current technologies can and cannot accomplish. ⇒ **Outcome a**

C9. Understanding of computing's limitations, including the difference between what computing is inherently incapable of doing vs. what may be accomplished via future science and technology.⇒ **Outcome b,j**

C8. Understanding of what current technologies can and cannot accomplish. ⇒ **Outcome a**

C9. Understanding of computing's limitations, including the difference between what computing is inherently incapable of doing vs. what may be accomplished via future science and technology.⇒ **Outcome b,j**

10. List of topics

1. Program Representation
2. Language Translation and Execution
3. Syntax Analysis
4. Compiler Semantic Analysis
5. Code Generation

11. Methodology and Evaluation

Methodology:

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of the UTEC Online virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

Evaluation System:

12. Content

Unit 1: Program Representation (5)	
Competences Expected: C9	
Learning Outcomes	**Topics**
• Explain how programs that process other programs treat the other programs as their input data [Familiarity] • Describe an abstract syntax tree for a small language [Familiarity] • Describe the benefits of having program representations other than strings of source code [Familiarity] • Write a program to process some representation of code for some purpose, such as an interpreter, an expression optimizer, or a documentation generator [Familiarity] • Explain the use of metadata in run-time representations of objects and activation records, such as class pointers, array lengths, return addresses, and frame pointers [Familiarity] • Discuss advantages, disadvantages, and difficulties of just-in-time and dynamic recompilation [Familiarity] • Identify the services provided by modern language run-time systems [Familiarity]	• Programs that take (other) programs as input such as interpreters, compilers, type-checkers, documentation generators • Abstract syntax trees; contrast with concrete syntax • Data structures to represent code for execution, translation, or transmission • Just-in-time compilation and dynamic recompilation • Other common features of virtual machines, such as class loading, threads, and security.
Readings : [Lou04b], [PV98]	

Unit 2: Language Translation and Execution (10)	
Competences Expected: C8	
Learning Outcomes	**Topics**
• Distinguish a language definition (what constructs mean) from a particular language implementation (compiler vs interpreter, run-time representation of data objects, etc) [Assessment] • Distinguish syntax and parsing from semantics and evaluation [Assessment] • Sketch a low-level run-time representation of core language constructs, such as objects or closures [Assessment] • Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Assessment] • Identify and fix memory leaks and dangling-pointer dereferences [Assessment] • Discuss the benefits and limitations of garbage collection, including the notion of reachability [Assessment]	• Interpretation vs. compilation to native code vs. compilation to portable intermediate representation • Language translation pipeline: parsing, optional type-checking, translation, linking, execution – Execution as native code or within a virtual machine – Alternatives like dynamic loading and dynamic (or "just-in-time") code generation • Run-time representation of core language constructs such as objects (method tables) and first-class functions (closures) • Run-time layout of memory: call-stack, heap, static data – Implementing loops, recursion, and tail calls • Memory management – Manual memory management: allocating, deallocating, and reusing heap memory – Automated memory management: garbage collection as an automated technique using the notion of reachability
Readings : [Aho+08], [Aho90], [Lou04a], [TS98], [ALe96], [App02]	

Unit 3: Syntax Analysis (10)	
Competences Expected: C8	
Learning Outcomes	**Topics**
• Use formal grammars to specify the syntax of languages [Assessment] • Use declarative tools to generate parsers and scanners [Assessment] • Identify key issues in syntax definitions: ambiguity, associativity, precedence [Assessment]	• Scanning (lexical analysis) using regular expressions • Parsing strategies including top-down (e.g., recursive descent, Earley parsing, or LL) and bottom-up (e.g., backtracking or LR) techniques; role of context-free grammars • Generating scanners and parsers from declarative specifications
Readings : [Aho+08], [Aho90], [Lou04a], [TS98], [ALe96], [App02]	

Unit 4: Compiler Semantic Analysis (15)	
Competences Expected: C8	
Learning Outcomes	**Topics**
• Implement context-sensitive, source-level static analyses such as type-checkers or resolving identifiers to identify their binding occurrences [Assessment] • Describe semantic analyses using an attribute grammar [Assessment]	• High-level program representations such as abstract syntax trees • Scope and binding resolution • Type checking • Declarative specifications such as attribute grammars
Readings : [Aho+08], [Aho90], [Lou04a], [TS98], [ALe96], [App02]	

Unit 5: Code Generation (20)	
Competences Expected: C8	
Learning Outcomes	**Topics**
• Identify all essential steps for automatically converting source code into assembly or other low-level languages [Assessment] • Generate the low-level code for calling functions/methods in modern languages [Assessment] • Discuss why separate compilation requires uniform calling conventions [Assessment] • Discuss why separate compilation limits optimization because of unknown effects of calls [Assessment] • Discuss opportunities for optimization introduced by naive translation and approaches for achieving optimization, such as instruction selection, instruction scheduling, register allocation, and peephole optimization [Assessment]	• Procedure calls and method dispatching • Separate compilation; linking • Instruction selection • Instruction scheduling • Register allocation • Peephole optimization
Readings : [Aho+08], [Aho90], [Lou04a], [TS98], [ALe96], [App02]	