Term/Year: Spring 2022

Subject Code and Course Number: CS 336

Course Title: Compiler Design

Number of Credits: 3

Instructor Name: Norayr Chilingarian, Rouben Khachatrian (TA)

Email Address: nchilingaryan@aua.am

Telephone Number: none. XMPP: norayr [at] spyurk [dot] am.

Office Location: 310W.

Office Hours: MWF after the class till 18:00, also by appointment.

Class schedule: MWF 12:30 AM - 13:20 PM.

Course Description: An introduction to the basic phases of modern compilers and their design principles. Topics covered include CPU instruction, finite state machines, lexical scanning, parsing schemes, code generation and translation, comparison of modern programming languages. As part of the course, students build a working compiler for an object-oriented language. Three hours of instructor-led class time per week including discussions and problem sets.

Prerequisites: Computer Organization class

Co-Requisites:

Required Materials: please note, the list below contains HTTP links:

- Niklaus Wirth: Compiler Construction (new version)part 1; part2; old version here
- Jack W. Crenshaw: Let's build a compiler.
- Per Brinch Hansen: Brinch Hansen on Pascal Compilers (1985, ISBN 0-13-083098-4)
- Jonathan Bartlett: Programming from the ground up.
- Niklaus Wirth, Jürg Gutknecht: Project Oberon: The Design of an Operating System and Compiler. Old edition here.
- Sergey Sverdlov: Programming Languages and Translation Techniques.
- (optional) Compilers: Principles, Techniques, and Tools

Schedule & Topics: [Note: Review and Q&A in preparation for exams should be scheduled and noted on syllabus.]

Course Syllabus is subject to change to address student needs.

Week	Topic
0	High level programming languages, assembly language. Compilers, interpreters.
1	Meta languages, T diagrams. Programming languages genealogy and concepts.
2	Language and syntax. Regular languages. Analysis of context-free languages. The method of recursive-descent. Table driven top-down parsing. Bottom-up parsing.
3	Assemblers and machine code. Addressing modes. Include files. Separation of the source in to different files. The C programming language. C preprocessor.
4	Structured programming. Separate compilation. Data protection in C. Heartbleed bug analysis, and what is the connection to the programming language used. Reading from buffers in C language.

Week	Topic
5	Data types. Elementary data types. Scalar, range, integers, real, boolean, arrays, enumerations, sets, records. Compatibility between numeric data types. Set logical operation implementation by using bitmasks. Comparing implementation of boolean types in C++ and Modula-2.
6	Expressions and assignments. Code generation according to the stack principle. Delayed code generation. Indexed variables and record fields. More on language design: MISRA standard analysis. How safety can be achieved? How to define and measure programming language complexity?
7	Structured programming: functions. Procedures and concept of locality. API and ABI. Calling conventions. Runtime organization of the store. Stack pointer, Frame pointer.
8	Separate compilation approaches. Namespaces. Java classes and packages. Analysing C# namespaces.

Week	Topic
9	Modules and separate compilation. The principle of information hiding. Separate compilation. Symbol files. Addressing external objects. Object Pascal (delpih), Modula-2 and Oberon approaches.
10	Does Object oriented programming exist? Type extension. Type projections. Typed functions.
11	Open arrays, pointers, and procedure types. Dynamic data structures and pointers.
12	Conditional and repeated statements and boolead expressions. Comparisons and jumps.
13	Code optimizations and frontend/backend structure. Simple optimizations.
14	Code optimizations: avoiding repeated evaluations, register allocation. preparation for the Final exam.
15	Functional languages. Lambda calculus. Lazy evaluation.

Student Learning Outcomes:

The following chart shows alignment between course-specific and program student learning outcomes and program goals. [Note: in determining course-specific outcomes, it is important to review the curriculum map to relate the appropriate skill level if specified (e.g. beginner, intermediate, and advanced. Outcomes should be clear, attainable, and measurable.)

Program Goal	Program Student Learning Outcomes	Course Learning Objectives
To provide with knowledge and skills in design, development and management of efficient computing systems that solve real-life industrial and academic problems.	The students completing the course are expected to possess the following skills and abilities:	1. Theory: Introduce the essential elements of programming language translations.
		2. Theory: Discuss the structure and concepts of modern programming languages.
	2. Be able to assume different roles in a project, including leadership and management.	3. Theory: Be acquainted with language & grammar theories. Be able to describe the language grammar in a dedicated notation.
	3. Possess the skills of development of translators:	4. Have understanding of compilation steps: lexical and syntax analysis. Creation of abstract syntax tree. Generating native code. Linkers and loaders.
2. To nurture the professional values of teamwork, integrity and leadership.		5. Programming: Learn to write recoursive-descent parser and code generator for an existing hardware

Program Goal	Program Student Learning Outcomes	Course Learning Objectives
		6. Tools: Have experience using compiler compiler tools to create autogenerated translators.

Method of Evaluation:

Student learning will be evaluated on the basis of the following weighted components:

- Homework assignments: 50% of final grade
- Grade of midterm examinations: 25%
- Grade of final examination: 25%

Make-up Procedures:

Students are required to take tests, exams and quizzes when they are scheduled by the instructor. In the event that a student misses a test, exam or quiz, the instructor is under no obligation to give a make-up, unless the student brings convincing, objective evidence that it was impossible for the student to take it at the scheduled time due to a medical or other emergency. Students should give instructors written notice of any absences from tests, exams or quizzes BEFORE the test, exam or quiz. In the event of an unscheduled quiz, the student should have a good reason for absence. If there is no good reason for the absence, it is up to the instructor to decide how or whether to give a make-up exam or take into account the missing work when calculating the final grade.

Library and Media/Technology Use

To enhance their overall learning in the course, students are strongly encouraged to use supplemental online and reference materials available in the library.

Policy on Grade Appeal

Students are entitled to appeal grades in line with the university's Grades Policies policy which is available online at http://policies.aua.am

Standards for Academic Integrity

Students are required to conduct themselves in an academically responsible and ethical manner in line with the Student Code of Ethics. Acts of academic dishonesty impair the academic integrity of AUA and create an unfair academic advantage for the student involved and other member(s) of the academic community. These acts are subject to disciplinary measures as prescribed in the AUA Code of Student Ethics http://policies.aua.am/policy/10

The Student Code of Conduct can be found at http://policies.aua.am/policy/101

Special Needs:
Students requiring special accommodations for learning should contact the Center for Student Success by the end of the Drop/Add period with such requests. studentsuccess@aua.am, http://studentsuccess.aua.am/disability-support-services/

Template for Assignment-Specific Rubric

Assignment Name	**Advanced**	**Proficient**	**Developing**	**Not Yet Competent**
	a) b) c)	a) b) c)	a) b) c)	a) b) c)
—	——	——	——	——
Relevant SLOs				
—	—	—		
Students will be able to:	*Course-Specific Learning Outcome:*	*Program Specific Learning Outcomes:*		