جامعة فيلادلفيا Philadelphia University	**اسم النموذج : خطة المادة الدراسية**	**رمز النموذج :** QFO-AP-VA-008
	الجهة المصدرة: نائب الرئيس للشؤون الأكاديمية	**رقم الإصدار: (Rev)** 2
	الجهة المدققة : اللجنة العليا لضمان الجودة	**تاريخ الإصدار:** 2021-5-4
		عدد صفحات النموذج : 4

Course Title: Compiler Construction				**Course code:** 750324	
Course Level: 3				**Course prerequisite (s) and/or co requisite (s):** (750224)	
Lecture Time: : 08:15 - 09:45				**Credit hours:** 3	
	UR ☐	**FR** ☐	**DR** ■	**C** ■	**E** ☐

Academic Staff Specifics

Name	Rank	Office	Office Hours	E-mail Address

The Learning Style Used in Teaching the Course

The Learning Style			
Blended Learning ☐			
Electronic Learning ☐			
Face-to-Face Learning ■			
Face-to-Face	**Electronic**	**Blended**	**Percentage**

Course module description:

This module aims to show how to apply the theory of language translation introduced in the prerequisite courses to build compilers and interpreters. It covers the building of translators both from scratch and using compiler generators. In the process, the module also identifies and explores the main issues of the design of translators. Topics include compiler design, lexical analysis, parsing, symbol tables, declaration and storage management, code generation. The construction of a compiler/interpreter for a small language is a necessary component of this module, so students can obtain the necessary skills.

Course/Module Objectives:

In the realm of software development, it is imperative to construct scanners and parsers using both top-down and bottom-up paradigms. This can be achieved through manual implementation or with the assistance of tools. Additionally, the creation of abstract syntax trees is closely associated with this process.

This paper aims to discuss the architecture and functionality of symbol tables, with a particular focus on their utilisation in type checking and other semantic tests.

This paper aims to elucidate the process by which executable code can be derived from an abstract syntax tree.

This response will provide an academic explanation of miscellaneous forms of optimisation, with a focus on the utilisation of liveness analysis for register allocation. Optimization techniques play a crucial role in enhancing the performance of computer programmes. Various forms of optimisation exist, each targeting different aspects of programme execution. One such form is liveness analysis, which aids in efficient register allocation. Liveness analysis is a static analysis technique that determines the lifespan of variables within a programme. It identifies the points in the programme where a variable is used and defines its live range, which encompasses the instructions between its last use and its next definition. By analysing the liveness of variables, compilers can make informed decisions regarding register allocation. Register allocation is the process of assigning variables to

Explain the operational characteristics of the algorithms under consideration for garbage collection.

Textbook:

Title: Compiler Design
Author(s): Kaushal Kishor Rastogi
Publisher: New Delhi: Global Academic Publishers & Distributors
Year of Publication: 2014

Support material (s):

Slides: MIT Computer Language Engineering Courseware

Teaching methods:

Lectures, tutorials, problem solving, laboratory

Duration: 15 weeks, 45 hours in total
Lectures: 36 hours
Tutorials: 06 hours
Laboratory: 03 hours

Learning outcomes:

- **Knowledge and understanding**

  A1. Recognize basics of automata theory, principle and the structure of translators

  A2. Recognize the relevant basic concepts inherent to translators, their components, the difference between compilers and interpreters, the difference between LL(1), LR (1), and SLR grammars, the difference between bottom up and top down parsers.

- **Cognitive Skills (thinking and analysis).**

  B1. Analyze and Compare bottom up and top-down parsing

  B2. Design a compiler for a simple programming language.

- **Practical Skills**

  C1. Plan and undertake a group project in the area of compilers

  C2. Implement specific components of translators

- **Transferable Skills and Personal Qualities**

  D1. Display an integrated approach to the deployment of communication skills, use IT skills and display mature computer literacy

Learning Outcomes Achievements:

A1, A2, B1, and B2 are achieved through lectures and assessed by quizzes and examinations

C1, C2, and D1 are achieved and assessed by homework: practical and research works

Assessment instruments

- Quizzes.& Home works
- Final examination: 40 marks

Allocation of Marks	
Assessment Instruments	**Mark**
Mid Examination	**30**
Final Examination	**40**
Quizzes, Home works	**30**
Total	**100**

Assignments All assignments will be announced or handed out in class. Many assignments will require programming in Python. All individual assignments, whether programming or not, are to be done individually. While you may discuss the assignment in general terms with others, your solutions should be composed, designed, written and tested by you alone. If you need help, consult the TA or the instructor.

Documentation and academic honesty

Submit your home work covered with a sheet containing your name, number, course title and number, and type and number of the home work (e.g. tutorial, assignment, and project).

Any completed homework must be handed in to my office (room IT 602) by 15:00 on the due date. After the deadline "zero" will be awarded. You must keep a duplicate copy of your work because it may be needed while the original is being marked.

You should hand in with your assignments:

1- A printed listing of your test programs (if any).
2- A brief report to explain your findings.
3- Your solution of questions.

For the research report, you are required to write a report similar to a research paper. It should include:

- **Abstract**: It describes the main synopsis of your paper.
- **Introduction**: It provides background information necessary to understand the research and getting readers interested in your subject. The introduction is where you put your problem in context and is likely where the bulk of your sources will appear.
- **Methods (Algorithms and Implementation)**: Describe your methods here. Summarize the algorithms generally, highlight features relevant to your project, and refer readers to your references for further details.
- **Results and Discussion (Benchmarking and Analysis)**: This section is the most important part of your paper. It is here that you demonstrate the work you have accomplished on this project and explain its significance. The quality of your analysis will impact your final grade more than any other component on the paper. You should therefore plan to spend the bulk of your project time not just gathering data, but determining what it ultimately means and deciding how best to showcase these findings.
- **Conclusion**: The conclusion should give your reader the points to "take home" from your paper. It should state clearly what your results demonstrate about the problem you were tackling in the paper. It should also generalize your findings, putting them into a useful context that can be built upon. All generalizations should be supported by your data,

however; the discussion should prove these points, so that when the reader gets to the conclusion, the statements are logical and seem self-evident.

- **Bibliography:** Refer to any reference that you used in your assignment. Citations in the body of the paper should refer to a bibliography at the end of the paper.

- **Protection by Copyright**

1. Coursework, laboratory exercises, reports, and essays submitted for assessment must be your own work, unless in the case of group projects a joint effort is expected and is indicated as such.
2. Use of quotations or data from the work of others is entirely acceptable, and is often very valuable provided that the source of the quotation or data is given. Failure to provide a source or put quotation marks around material that is taken from elsewhere gives the appearance that the comments are ostensibly your own. When quoting word-for-word from the work of another person quotation marks or indenting (setting the quotation in from the margin) must be used and the source of the quoted material must be acknowledged.
3. Sources of quotations used should be listed in full in a bibliography at the end of your piece of work.

- **Avoiding Plagiarism**.

1. Unacknowledged direct copying from the work of another person, or the close paraphrasing of somebody else's work, is called plagiarism and is a serious offence, equated with cheating in examinations. This applies to copying both from other students' work and from published sources such as books, reports or journal articles.
2. Paraphrasing, when the original statement is still identifiable and has no acknowledgement, is plagiarism. A close paraphrase of another person's work must have an acknowledgement to the source. It is not acceptable for you to put together unacknowledged passages from the same or from different sources linking these together with a few words or sentences of your own and changing a few words from the original text: this is regarded as over-dependence on other sources, which is a form of plagiarism.
3. Direct quotations from an earlier piece of your own work, if not attributed, suggest that your work is original, when in fact it is not. The direct copying of one's own writings qualifies as plagiarism if the fact that the work has been or is to be presented elsewhere is not acknowledged.
4. Plagiarism is a serious offence and will always result in imposition of a penalty. In deciding upon the penalty the Department will take into account factors such as the year of study, the extent and proportion of the work that has been plagiarized, and the apparent intent of the student. The penalties that can be imposed range from a minimum of a zero mark for the work (without allowing resubmission) through caution to disciplinary measures (such as suspension or expulsion).

Course/module academic calendar

week	Basic and support material to be covered	Homework/reports and their due dates
(1)	Introduction to Translators and Compilers	
(2)	Lexical Analysis: Introduction, Regular Expressions, Grammar language	
(3)	Lexical Analysis: Finite Automata (DFA, NFA), Lex: Lexical generator tool	
(4)	**Tutorial1**: Handwritten Lexical Analyzer Syntactic Analysis: Introduction	
(5)	Syntactic Analysis: Bottom Up Parsing Shift reduce parsing	Implementation of a hand written lexical

		analyzer
(6) First exam.	Shift Reduce parsing Parse table	
(7)	**Tutorial2** Syntax Analysis: Top Down Parsing (1)	
(8)	Syntax Analysis: Top Down Parsing (2)	
(9)	**Tutorial3** Semantic Analysis: Introduction	
(10)	Type Checking (1)	Implementation of a simple calculator using lex/yacc tools
(11) Second exam.	Type checking (2) **Tutorial 4**	
(12)	Intermediate Representation (1)	
(13)	Intermediate Representation (2) **Tutorial 5**	
(14)	Machine Code Generation (1)	
(15)	Machine Code Generation (2) **Tutorial 6**	
(16) Final Exam.	Revision	

Expected workload:
On average students need to spend 2 hours of study and preparation for each 50-minute lecture/tutorial.

Attendance policy:
Absence from lectures and/or tutorials shall not exceed 15%. Students who exceed the 15% limit without a medical or emergency excuse acceptable to and approved by the Dean of the relevant college/faculty shall not be allowed to take the final examination and shall receive a mark of zero for the course. If the excuse is approved by the Dean, the student shall be considered to have withdrawn from the course.

Module references

Books:
1. Keith Cooper, Linda Torczon, Engineering a Compiler, Imprint: Morgan Kaufmann, 2011
2. Alfred V. Aho, Ravi Sethi and Jeffry D. Ulman, Compilers: Principles, Techniques and Tools, Addison Wesley Longman, 2007
3. W. Appel, Modern Compiler Implementation in Java, Prentice Hall, 2002
4. D. Watt, Brown, Programming Language Processors in Java: Compilers and Interpreters, Prentice hall, 2000

Website:
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-035-computer-language-engineering-spring-2010/ **(MIT CourseWare)**