Republic of Iraq
The Ministry of Higher Education
& Scientific Research
2024-2023

Flow up of implementation syllabus

University: Thi - Qar
College: Education for Pure Scie.
Department:compuer scinces
Stage: third
Lecturer name: Mohammed Morad
Academic Status: Assistant Professor
Qualification: Master of IT
Place of work: In same college

Course Instructor	Mohammed Morad Anad
E-mail	Mohameedmurad.comp@utq.edu.iq
Title	Compilers
Course Coordinator	annual
Course Objective	The goal of the course is to teach students the subject of compilers theoretically and practically and to know what happens during the execution of the program inside the computer, starting from the source program all the way to an understandable computer program (the target program). A- Cognitive objectives: **1 −That the student is able to understand the translation** material adequately. 2 −That the student should be able to understand the necessary steps to convert any program from the source language into a language understandable to the computer. 3−The student should distinguish between the six stages of the translator. 4 −The student learns about the progress made in designing compilers. 5 −That the student can understand the stages in which errors are corrected during the implementation of the program. B- Skills objectives of the course: **1 −That the student gains the ability to correct errors** during the implementation of the program during the six stages of the compiler.

	2 –That the student gains the ability to program each stage of the compiler. 3 –That the student will be able to apply the algorithms specific to the work of the translator inside the calculator. 4 –That the student acquires the ability to compile the programming of each stage of the compiler in the form of a single program. C- Emotional and value goals: **1 –The student should appreciate the efforts of scientists in** developing compilers and their importance in implementing programs inside the computer. 2 –That the student appreciates the importance of the compilers course as an important course within computer science courses. 3 –To participate in the discussion during the lecture. 4–The student should take the initiative to solve various extracurricular activities and examples. D- General and qualifying transferable skills (other skills related to employability and personal development): **1–The student must be able to teach the subject**. 2–The student should be able to benefit from the knowledge he has acquired. 3 –The student should be able to use the material in other subjects. 4-The student should be able to apply the material practically.
	A- Teaching and learning methods:

<table>
<tr><td><b>Course Description</b></td><td colspan="5">1 –Lecture method. 2- Question and answer method. 3- Laboratory education to acquire practical skills.
4–Method of discussion. 5- Assigning the student to some group activities and duties.
B-Evaluation methods:
1- Conducting daily oral, daily and semester theoretical and practical tests. 2-Writing reports. 3-Allocating part of the grade to reports and group assignments carried out by the student.</td></tr>
<tr><td><b>Textbook</b></td><td colspan="5">1 –Compilers principles, techniques, and tools, by Alfred V. Aho, Monica S. Lam, Ravi Sethi , and Jeffrey D.Ullman.</td></tr>
<tr><td rowspan="2"><b>Course Assessment</b></td><td>Term Tests</td><td>Laboratory</td><td>Quizzes</td><td>Project</td><td>Final Exam</td></tr>
<tr><td>30</td><td>15</td><td>3</td><td>2</td><td>15 practical + 35 theoretical</td></tr>
<tr><td><b>General Notes</b></td><td colspan="5"><b>Course development plan</b>:
•Adding new topics and examples to the material, which are:
1 –Adding the topic Token, patters and lexemes to the lexical analyzer phase
2 –Add the attributes for token topic to the lexical analyzer phase
3 –Adding the topic “lexical errors” to the “lexical analyzer” phase
4 –Add the context free grammar topic to syntax analyzer
5 –Add the ambiguity topic to the syntax analyzer stage
6 –Adding the construction of predictive parsing table topic to the syntax analyzer phase.
7- Adding the syntax error handling topic to the syntax analyzer phase.</td></tr>
</table>

Course weekly Outline

week	Date	Theoretical material	Lab. Experiment Assignments	Notes
1	17/9/2023	**definitions , compiler, assembler, linker**	Theoretical material: Introduction to programming libraries for programming strings, letters, and other symbols	
2	24/9/2023	**definition of compiler phases and error handler, symbol table manager**	Introducing the functions for dealing with strings, letters, and symbols	
3	01/10/2023	**construction tools, type of grammar description with example**	Programming a group of mobile phones that are part of the compiler programming	
4	08/10/2023	**converting one type to another of grammar description**	Programming other functions involved in creating and working the compiler	
5	15/10/2023	**Finite state automata FSA, with its structure representation and its two types. - closure function**	Programming an algorithm for a finite state machine	
6	22/10/2023	**Sub phases of lexical analysis: 1-algorithm of converting any transition diagram (T.D) to non-deterministic finite state automata(NDFSA).**	Programming an algorithm for converting a machine state transition diagram into an indefinite finite machine	
7	29/10/2023	**2- Algorithm of converting NDFSA to DFSA**	Programming an algorithm for converting an indefinite finite state machine into a definite finite state machine	
8	05/11/2023	**3-minimization of DFSA**	Programming an algorithm for minimizing the states of a finite state machine	
9	12/11/2023	**FSA accepter (recognizer) algorithm .**	Programming an algorithm The process of accepting or distinguishing finite states of strings	
10	19/11/2023	**AHO algorithm for tokens recognition.**	Programming and implementing the Ahoo algorithm to distinguish symbols or extracts	

11	26/11/2023	**reviewing**	review	
12	03/12/2023	**Syntax analyzer: - architecture of parsing, grammar derivation (right-most and left most).**	Programming that distinguishes and implements the precedence of transactions in the programming code	
13	10/12/2023	**Recursion, its types (immediate left recursion and not immediate left recursion), elimination of left recursion.**	Programming an algorithm to remove direct and indirect left repetitions from context-free rules	
14	17/12/2023	**First and follow algorithm.**	Programming functions to find the first and next set of variables in the products of the free context rule	
15	24/1/2023	**Exams**		
16	31/1/2023	**Exams**		

Half-year Break

17	28/1/2024	**Bottom up parser (shift reduce parser) with specifying of handle.**	Arabized programming from bottom to top	
18	04/2/2024	**Operator precedence parser.**	Parsed programming of transaction precedence	
19	11/2/2024	**LR parser**	LR parsed programming	
20	18/2/2024	**SLR parser**	LSR parsing programming	
21	25/2/2024	**LALR parser**	LALR parsing programming	
22	03/3/2024	**syntax directed translation**	Complete the programming of the LALR parser	
23	10/3/2024	**semantic analyzer : static semantic checks dynamic semantic checks examples**	Programming examples semantic analyzer	

24	**17/3/2024**	**intermediate code generation polish notation (infix, prefix, postfix)**	Programming examples of generating intermediate code with expressions (infix, prefix, postfix)	
25	**24/3/2024**	**Triples, three address code, quadruples. Converting between one code type to another.**	Programming the three-address and four-address code generation and programming the conversion from one type to another	
26	**31/3/2024**	**reviewing**	review	
27	**07/4/2024**	**code optimizer: introduction , principles of optimization peephole optimization**	Programming examples of improving the intermediate code and making it perfect in this phase	
28	**14/4/2024**	**Optimization of blocks loops in flow graph.**	Programming examples to illustrate data flow analysis and improve its transmission and translation	
29	**21/4/2024**	**global data flow analysis, code improvement transformation**	Programming examples explaining obtaining the target code at execution time	
30	**28/4/2024**	**Code generation: target machine run time storage management, basic blocks and flow graph.**	Programming examples: Generate simple code and define its registers	
31	**05/5/2024**	**Simple code generator registers allocation and assignment. the dag representation of basic blocks, generating code from dag**	Programming examples: Generate simple code and define its registers	
32	**12/6/2024**	**Exams**		

Flow up of implementation syllabus

تؤيد اللجنة العلمية مطابقة الخطة التدريسية لمفردات منهج المادة الدراسية

……………………………………
Instructor Signature(Lab.)

………………………………………
Instructor Signature(Theoretical)

………………………………………………………………………………
1st Scientific committee member

………………………………………………………………………………
2nd Scientific committee member

………………………………………………………………………………
3rd Scientific committee member

………………………………………………..
Head of Scientific committee

…………………………………………………
Dean