Akashi College	Year	2024	Course Title	Compiler

Course Information

Course Code	6521	Course Category	Specialized / Compulsory
Class Format	Lecture	Credits	School Credit: 1
Department	Electrical and Computer Engineering Computer Engineering Course	Student Grade	5th
Term	First Semester	Classes per Week	2
Textbook and/or Teaching Materials			
Instructor	MIURA Kinya		

Course Objectives

The aim of this course is to gain the knowledge necessary to develop the compiler and to gain a better understanding of the programming language and the execution of the program.
The specific goals are:
[1] To understand the theoretical fundamentals and methods of lexical analysis
[2] To understand the theoretical fundamentals and methods of syntactic analysis.
[3] To understand the methods of semantic analysis and code generation.
Learn these techniques to develop your compiler creation skills. By learning the theoretical fundamentals of these methods, acquire the basic ability to design programming language and its compiler, and increase practical programming skills through a deeper understanding of the execution of the program.

Rubric

	Ideal Level	Standard Level	Unacceptable Level
Achievement 1	Understand the theoretical basis and methods of lexical analysis and can write a program of lexical analysis precisely.	Understand the theoretical basics and methods of lexical analysis mostly, and can write a program of lexical analysis mostly.	Do not understand the theoretical basis and methods of lexical analysis, and cannot write a program of lexical analysis.
Achievement 2	Understand the theoretical basics and methods of syntactic analysis and can write a program of syntactic analysis precisely.	Understand the theoretical basics and techniques of parsing mostly, and can write a program of syntactic analysis mostly.	Do not understand the theoretical fundamentals and techniques of syntactic analysis, and cannot write a program of syntactic analysis.
Achievement 3	Understand semantic analysis and code generation methods, and can write a program of semantic analysis and code generation precisely.	Understand semantic analysis and code generation techniques mostly and can write a program of semantic analysis and code generation mostly.	Do not understand semantic analysis and code generation techniques, and cannot write a program of semantic analysis and code generation.

Assigned Department Objectives

Teaching Method

Outline	When writing programs, it is common to use a programming language that is highly human-readable (a high-level language). The compiler is a software that converts programs written in such a high-level language into a language (machine language) that the CPU can interpret and execute. In this course, we will lecture students various theories about the syntax and semantics of programming languages and the various methods that have been developed based on them for converting programming languages into machine languages.
Style	The lecture is mainly based on the content of textbook, but should be supplemented with handouts if required. Also, tasks will be assigned as appropriate. We have a practical training on 15th week. The contact person is Yukihiro Hamada.
Notice	Before taking the lectures, it is desirable for students to study microcomputers (assembly language), programming II, data structures and algorithms, discrete mathematics (finite automata, and formal language theory), or equivalent subjects. Students who miss 1/3 or more of classes will not be eligible for evaluation.

Characteristics of Class / Division in Learning

☐ Active Learning	☑ Aided by ICT	☐ Applicable to Remote Class	☐ Instructor Professionally Experienced

Course Plan

			Theme	Goals
1st Semester	1st Quarter	1st	Compiler overview	Can explain the theoretical model of the compiler, the general compilation process, and the structure of the compiler.
		2nd	Lexical analysis 1 of 3	Can handle regular expressions (RE) and finite automaton (FA) as the theoretical basis of the compiler's lexical analysis).
		3rd	Lexical analysis 2 of 3	Can construct a finite automaton that accepts lexical structures expressed in regular expressions.
		4th	Lexical analysis 3 of 3	Can explain the lexical analysis program using state transition tables.
		5th	Grammar (Formal language theory) 1 of 2	Can handle formal language theory, in particular the context-free grammar (CFG) commonly used in syntax definitions of programming language. Can also explain BNF, extended BNF, and syntax diagrams.

		6th	Grammar (Formal language theory) 2 of 2	Concepts in formal language theory: Can explain the derivation of the symbolic column, the deriving of the leftmost/rightmost column, the parsing tree, the ambiguity of the grammar, etc.
		7th	Syntactic analysis 1 of 3	Can explain recursive downward parsing, especially LL(1) parsing. Can also solve the problem of left recursion.
		8th	Midterm exam It is given during class.	
	2nd Quarter	9th	Syntactic analysis 2 of 3	Understand and can explain LR parsing.
		10th	Syntactic analysis 3 of 3	Can explain how to handle ambiguous grammar and errors.
		11th	Semantic analysis 1 of 2	Can explain the semantics analysis and how to map names to the objects that represent in the language (name resolution). Can also explain scopes and namespaces.
		12th	Semantic analysis 2 of 2	Can explain the handling of forward references, type checking, type conversion, and error handling.
		13th	Code generation 1 of 2	Define a model for a specific execution environment and can explain the generation of code corresponding to function calls. Can also explain how to allocate storage area for local variables, etc.
		14th	Code generation 2 of 2	Can explain how to generate codes for various statements and expressions.
		15th	Practical training	Can create a simple language processing system using "bison" and "flex".
		16th	Final exam	

Evaluation Method and Weight (%)

	Examination	Task	Mutual Evaluations between students	Behavior	Portfolio	Other	Total
Subtotal	80	20	0	0	0	0	100
Basic Proficiency	0	0	0	0	0	0	0
Specialized Proficiency	80	20	0	0	0	0	100
Cross Area Proficiency	0	0	0	0	0	0	0