Source: https://www.javatpoint.com/compiler-tutorial

22CS302 COMPILER DESIGN

Hours Per Week :

L	T	P	C
2	2	0	3

PREREQUISITE KNOWLEDGE: Programming for problem solving- I & II and Formal languages and automata theory.

COURSE DESCRIPTION AND OBJECTIVES:

This course introduces the foundation for understanding the theory and practice of compilers and compiler design concepts; symbol table management, compiler parsing techniques, semantic analysis and optimized code generation. This course introduced the concepts of lexical analyzer, parser, code generation and code optimization techniques. The objective of this course is to enable the student to acquire the knowledge of various phases of compiler such as lexical analyzer, parser, code optimization and code generation.

MODULE-1

UNIT-1 **8L+8T+0P=16 Hours**

INTRODUCTION

The evolution of programming languages and basic language processing system; The structure of a compiler; Bootstrapping; Lexical analyser and its Role; Input buffering; Specifications and recognition of tokens; LEX.

UNIT-2 **8L+8T+0P=16 Hours**

SYNTAX ANALYSIS

The role of the parser; Context-free grammars; Types of parsers with examples, YACC.

Semantic Analysis: Type checking; Syntax directed definition (SDD) and translation schemes (TS); Application of SDD and TS; Translation of expressions and control flow statements.

PRACTICES:

- Implement various phases of compiler in detail. Write down the output of each phase for expression Total = (b + c) + (b + c) * 50.
- Construct the symbol table for any input file with the help of LEX tool.
- Consider the context free grammar.
  S→SS+, S→SS*, S→a and the string aa+a*.
  i) Give the leftmost derivation for the string.
  ii) Give the rightmost derivation of the string.
  iii) Is the grammar ambiguous or not.
- Check whether the following grammar is a LL (1) grammar.
  S →iEtS | iEtSeS | a, E → b.
- Construct the FIRST and FOLLOW procedures for the following grammar.
  S → Aa | bAC | dc | bda, A → d.
- Consider the grammar,
  E→TE', E'→+TE' | €, T → FT', T'→ *FT' | €, F → (E) | id.
  Construct a predictive parsing table for the grammar given above. Verify whetherthe input string id + id * id is accepted by the grammar or not.

MODULE-2

UNIT-1 **8L+8T+0P=16 hours**

INTERMEDIATE REPRESENTATIONS

Three address code; Syntax tree; DAG.

Run-Time Environment: Storage organization; Stack allocation - Activation Trees, Activation Records.

UNIT-2 **8L+8T+0P=16 Hours**

OPTIMIZATION AND CODE GENERATION

The principal sources of optimization; Basic blocks and flow graphs; Local optimization; Global optimization and loop optimization.

Code Generation: Issues in the design of code generator; Code-generation algorithm – register allocation and assignment and peephole optimization.

PRACTICES:

- Translate the executable statements of the following C-code segment into three address code.

```
int i:
int a[10]
i = 0;
While (I <= 10) {
a[i] = i + 1; i + + ;
}
```

- Compute the DAG for the following three address statements. Considering this DAG as an example, explain the process of code generation from DAG.

  t1 = a + b t2 = c + d t3 = e – t2 t4 = t1 - t3

- What is Data flow equation? Represent the Data flow information for the following

  a = b + c; d = c * d; e = a – c; f = d + e.

- Draw a flow graph for the below code. Show the basic blocks clearly in your control flow graph?

```
If ( i&gt;=0){
sum = B[0];
i = 0;
L1: if (A[i]&lt; B[i]){
j=i;
L2:
if( B[i]&gt;=0){
sum = sum +B[j];
}
j = j+1
if ( j&lt;N) goto L2;
}
i = i+1
if ( i&lt;N) goto L1;
}
```

SKILLS:

- ✓ *Design parsers using top-down and bottom-up approaches.*
- ✓ *Usage of tools like LEX and YACC.*
- ✓ *Design a simple code generation*

COURSE OUTCOMES:

Upon successful completion of this course, students will have the ability to:

CO No.	Course Outcomes	Blooms Level	Module No.	Mapping with POs
1	Apply the different phases of compiler with various examples.	Apply	1	1, 12
2	Design different parsing and optimization techniques in the design of compile.	Design	1	1, 2, 12
3	Analyze the code optimization techniques.	Analyze	2	1, 2, 3, 12
4	Analyze the algorithm for compiler segments and evaluate the algorithm for optimized code generation.	Analyze	2	1, 2, 3,12

TEXT BOOKS:

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ulman, “Compilers: Principles, Techniques and Tools”, 3rd Edition, Pearson Education, 2019.
2. Thomson, “Introduction to Theory of Computation”, 2nd Edition, Sipser, 2016.

REFERENCE BOOKS:

1. V. Raghavan, “Principles of Compiler Design”, 2nd Edition, Mc Graw Hill, 2016.
2. John R.Levin, Tony Mason and Doug Brown, “Lex & YAAC”, 2nd Edition, O Reilly, 2012.
3. Ms. Manisha Bharambe, “Compiler Construction”, 2nd Edition, Nirali Prakashan, 2017.