INSTITUTE OF AERONAUTICAL ENGINEERING

Dundigal, Hyderabad - 500 043

COMPUTER SCIENCE AND ENGINEERING

COURSE DESCRIPTION FORM

Course Title	**COMPILER DESIGN**			
Course Code	**A50587**			
Regulation	**R13 - JNTUH**			
Course Structure	Lectures	Tutorials	Practicals	Credits
	-	-	3	2
Course Coordinator	Mr. N V Krishna Rao, Associate Professor, CSE			
Team of Instructors	Ms. E Uma Shankari, Assistant Professor, CSE Ms. G Geetha, Assistant Professor, CSE			

I. COURSE OVERVIEW:

A language subset will be defined and used during the lab course. The programming exercises here consist of implementing the basic components of a compiler. The constructs in this subset are found in most programming languages.

II. PREREQUISITE(S):

Level	Credits	Periods/ Week	Prerequisites
UG	2	3	Operating Systems, Computer Programming, Data Structures

III. MARKS DISTRIBUTION:

Sessional Marks	End Semester Exam	Total Marks
There shall be a continuous evaluation during the semester for 25 marks. Day-to-day work in the laboratory shall be evaluated for 15 marks and internal practical examination conducted by the concerned teacher shall be evaluated for 10 marks.	50	75

IV. EVALUATION SCHEME:

S. No	Component	Duration	Marks
1.	Day-to-day Evaluation	-	15
2.	Internal Practical Examination	2.5 hours	10
5.	End Semester Examination	2.5 hours	50

V. COURSE OBJECTIVES:

At the end of the course, the students will be able to:

I. Explain the importance of compiler design.

II. Design and implementation of lexical analyzer using lex tools.

III. Explain the top down and bottom up parsing techniques using programming.

IV. Identify the understanding language peculiarities by designing a complete translator for mini language.

V. Explain that computing science theory can be used as the basis for real applications.

VI. COURSE OUTCOMES:

After completing this course the student must demonstrate the knowledge and ability to:

1. **Understand** the working of lex and yacc compiler for debugging of programs.
2. **Understand** and define the role of lexical analyzer, use of regular expression and transition diagrams.
3. **Understand** and use Context free grammar, and parse tree construction.
4. **Learn** & use the new tools and technologies used for designing a compiler.
5. **Develop** program for solving parser problems.
6. **Learn** how to write programs that execute faster.

VII. LIST OF EXPERIMENTS:

Division of Experiments	List of Experiments
Lexical analyzer	**WEEK-1** * Write a C program to identify whether a given line is a comment or not.
	WEEK-2 Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language.
	WEEK-3 *Write a C program to recognize strings under 'a', 'a*b+', 'abb'.
	WEEK-4 *Write a C program to test whether a given identifier is valid or not.
	WEEK-5 *Write a C program to simulate lexical analyzer for validating operators.
Lexical analyser-using LEX	**WEEK-6** Implement the lexical analyzer using JLex, flex or other lexical analyzer generating tools.
Top down parsing	**WEEK-7** Write a C program for implementing the functionalities of predictive parser for the mini language specified in Note 1.
	WEEK-8 a) *Write a C program for constructing of LL (1) parsing. b) *Write a C program for constructing recursive descent parsing.
Bottom up parsing	**WEEK-9** Write a C program to implement LALR parsing.
	WEEK-10 a) *Write a C program to implement operator precedence parsing. b) *Write a C program to implement SLR Parsing.
YACC	**WEEK-11** Convert the BNF rules into Yacc form and write code to generate abstract syntax tree for the mini language specified in Note 1.

Syntax tree	**WEEK-12** Write a C program to generate machine code from abstract syntax tree generated by the parser. The instruction set specified in Note 2 may be considered as the target code.

*****Content beyond the university prescribed syllabi

Note 1:

Consider the following mini language, a simple procedural high –level language, only operating on integer data, with a syntax looking vaguely like a simple C crossed with pascal. The syntax of the language is defined by the following grammar.

<program>::=<block>

<block>::={<variable definition><slist>}

|{<slist>}

<variabledefinition>::=int <vardeflist>

<vardec>::=<identifier>|<identifier>[<constant>]

<slist>::=<statement>|<statement>;<slist>

<statement>::=<assignment>|<ifstament>|<whilestatement>

|<block>|<printstament>|<empty>

<assignment>::=<identifier>=<expression>

|<identifier>[<expression>]=<expression>

<if statement>::=if<bexpression>then<slist>else<slist>endif

|if<bexpression>then<slisi>endif

<whilestatement>::=while<bexpreession>do<slisi>enddo

<printstatement>:;=print(<expression>)

<expression>::=<expression>::=<expression><addingop><term>|<term>|<addingop>

<term>

<bexprssion>::=<expression><relop><expression>

<relop>::=<|<=|==|>=|>|!=

<addingop>::=+|-

<term>::=<term><multop><factor>|<factor>

<Multop>::=*|/

<factor>::=<constant>|<identifier>|<identifier>[<expression>]

|(<expression>)

<constant>::=<digit>|<digit><constant>

<identifier>::=<identifier><letter or digit>|<letter>

<letter or digit>::=<letter>|<digit>

<letter>:;=a|b|c|d|e|f|g|h|I|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<digit>::=0|1|2|3|4|5|^|7|8|9

<empty>::=has the obvious meaning

Comments(zero or more characters enclosed between the standard C/JAVA Style comment brackets/*…*/)can be inserted .The language has rudimentary support for1-dimenstional array,the declaration int a[3] declares an array of three elements,referenced as a[0],a[1] and a[2].Note also you should worry about the scopping of names.

Note 2:

A simple language written in this language is

```
{int a[3],t1,t2;
T1=2;
A[0]=1;a[1]=2;a[t]=3;
T2=-( a[2]+t1*6)/(a[2]-t1);
If t2>5then
Print(t2)
Else{
Int t3;
T3=99;
T2=25;
Print(-t1+t2*t3);/*this is a comment on 2 lines*/
}endif}
```

Prepared by : Ms. E Uma Shankari, Assistant Professor, CSE
Ms. G Geetha, Assistant Professor, CSE

Date : 11 June, 2015

HOD, CSE