Reg. No. : □□□□□□□□□□

Question Paper Code: 41862

B.E. / B.Tech. DEGREE EXAMINATION, MAY 2017

Sixth Semester

Information Technology

14UIT602 - COMPILER DESIGN

(Regulation 2014)

Duration: Three hours Maximum: 100 Marks

Answer ALL Questions

PART A - (10 x 1 = 10 Marks)

1. A compiler can check

   (a) Logical Error
   (b) Syntax Error
   (c) Both Logical and Syntax Errors
   (d) Neither Logical nor Syntax error

2. The output of lexical analyzer is

   (a) A set of regular expressions
   (b) Syntax tree
   (c) Set of tokens
   (d) Strings of character

3. Syntax directed translation scheme is desirable because

   (a) It is based on the syntax
   (b) Its description is independent of any implementation
   (c) It is easy to modify
   (d) All of these

4. A top down parser generates

   (a) Right most derivation
   (b) Right most derivation in reverse
   (c) Left most derivation
   (d) Left most derivation in reverse

5. Intermediate code generation phase gets input from

   (a) Lexical analyzer
   (b) Syntax analyzer
   (c) Semantic analyzer
   (d) Error handling

6. Generation of intermediate code based on an abstract machine model is useful in compilers because

   (a) it makes implementation of lexical analysis and syntax analysis easier

   (b) syntax directed translation can be written for intermediate code generation.

   (c) It enhances the portability of the front end of the compiler

   (d) it is not possible to generate code for real machines directly from high level language programs

7. A compiler that runs on one machine and produces code for a different machine is called

   (a) Cross compilation (b) One pass compilation
   (c) Two pass compilation (d) None of these

8. DAG representation of a basic block allows

   (a) Automatic detection of local common sub expressions

   (b) Automatic detection of induction variables

   (c) Automatic detection of loop variant

   (d) None of these

9. The optimization technique which is typically applied on loops is

   (a) Removal of invariant computation (b) Peephole optimization

   (c) Constant folding (d) All the above

10. The optimization which avoids test at every iteration is

   (a) Loop unrolling (b) Loop jamming

   (c) Constant folding (d) None of these

PART - B (5 x 2 = 10 Marks)

11. Differentiate tokens, patterns and lexeme.

12. Write the algorithm for the construction of a predictive parsing table.

13. Write the three address code and postfix notation for the expression a * - (b + c).

14. What is a DAG? Mention its applications.

15. What is code motion?

PART - C (5 x 16 = 80 Marks)

16. (a) Explain the phases of compiler. and how the following statement will be translated in every phase: (i) a : = b + c * 50 (ii) a : = b * c - d. (16)

Or

(b) (i) Mention any four compiler construction tools with their benefits and drawbacks. (8)

(ii) Describe the need for grouping of phases of compiler. (8)

17. (a) Consider the grammar given below:

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T \ * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

Construct an LR parsing side for the above grammar. Give the moves of LR parser on id*id+id. (16)

Or

(b) (i) What are different storage allocation strategies? Explain. (8)

(ii) Specify a type checker which can handle expressions, statements and functions. (8)

18. (a) (i) Define three-address code. Describe the various methods of implementing three address statement with Example. (8)

(ii) Give the transition schema for converting the assignments into three address code. (8)

Or

(b) (i) Discuss the various methods for translating Boolean expression. (8)

(ii) Explain the process of generating the code for Boolean expression in a single pass using back patching. (8)

19. (a) Define a Directed Acyclic Graph. Construct a DAG and write the sequence of instructions for the expression $a + a * (b - c) + (b - c) * d$. (16)

Or

(b) (i) Define basic blocks. Write a algorithm to partition a sequence of three address statement into basic blocks. (6)

(ii) Briefly explain about simple code generator. (10)

20. (a) Write an algorithm to construct the natural loop of a back edge. (16)

Or

(b) (i) Explain peephole optimization. (8)

(ii) Optimize the following code using various optimization techniques:

```
i=1 ,s=0;

for(i=1;i<=3;i++)

for(j=1;j<=3;j++)

c[i][j]=c[i][j]+a[i][j]+b[i][j];
```

(8)