Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 91355

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2014.

Sixth Semester

Computer Science and Engineering

CS 2352/CS 62/10144 CS 602 — PRINCIPLES OF COMPILER DESIGN

(Regulation 2008/2010)

(Common to PTCS 2352–Principles of Compiler Design for B.E. (Part–Time) Fifth Semester–Computer Science and Engineering–Regulation 2009)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is the role of lexical analyzer?
2. Write regular expression to describe a languages consist of strings made of even numbers a and b.
3. List out the various storage allocation strategies.
4. Write a CF grammar to represent palindrome.
5. What are the types of intermediate languages?
6. Give syntax directed translation for case statement.
7. Differentiate between basic block and flow graph.
8. Draw DAG to represent $a[i] = b[i]$; $a[i] = \& t$;
9. Represent the following in flow graph
   $i = 1$; $sum = 0$; $while(i <= 10)\{sum+ = i; i + +;\}$
10. What is global data flow analysis?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Explain the need for grouping of phases of compiler. (8)

    (ii) Explain a language for specifying the lexical analyzer. (8)

    Or

    (b) (i) Write short notes on compiler construction tools. (8)

    (ii) Explain — specification and recognition of tokens. (8)

12. (a) (i) Explain the specification of simple type checker. (8)

(ii) Explain — runtime environment with suitable example. (8)

Or

(b) Find the LALR for the given grammar and parse the sentence (a+b)*c

$E \rightarrow E+T/T, T \rightarrow T*F/F, F \rightarrow (E)/id$. (16)

13. (a) Generate intermediate code for the following code segment along with the required syntax directed translation scheme

While (i<10)

If (i % 2 = = 0)

Evensum = evensum + i;

Else

Oddsum =oddsum + i;

Or

(b) Generate intermediate code for the following code segment along with the required syntax directed translation scheme. (16)

s=s+a[i][j];

14. (a) (i) Explain register allocation and assignment with suitable example. (8)

(ii) Explain — code generation phase with simple code generation algorithm. (8)

Or

(b) (i) Generate DAG representation of the following code and list out the applications of DAG representation. (8)

i = 1; while (i<=10) do

sum+ = a[i];

(ii) Explain — Generating code from DAG with suitable example. (8)

15. (a) (i) Explain — principle sources of optimization. (8)

(ii) Illustrate optimization of basic blocks with an example. (8)

Or

(b) Explain peephole optimization and various code improving Transformations. (16)
