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

Question Paper Code : 71391

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015.

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. Describe the error recovery schemes in the lexical phase of a compiler.

2. Write a regular Definition to represent date in the following format : JAN-5th 2014.

3. What is the role of a passer?

4. Construct a decorated parse tree according to the syntax directed definition, for the following input statement: (4 + 7.5 * 3) / 2

5. Write the 3-address code for ; $x = *y$; $a = \&x$.

6. Place the above generated 3-address code in Triplets and Indirect Triplets.

7. What role does the target machine play on the code generation phase of the compiler?

8. How is Liveness of a variable calculated

9. Generate code for the following C statement assuming three registers are available : $x = a/(b+c) - d*(e+f)$.

10. Write the algorithm that orders the DAG nodes for generating optimal target code.

PART B — (5 × 16 = 80 marks)

11. (a) Prove that the following two regular expressions are equivalent by showing that the minimum state DFA's are same.

(i) (a / b)* (8)

(ii) (a* / b*)* (8)

Or

(b) (i) Describe the error recovery schemes in the lexical phase of a compiler (8)

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

12. (a) (i) Generate SLR Parsing Table for the following grammar (12)

$S \rightarrow Aa \mid bAc \mid Be \mid bBa$

$A \rightarrow d$

$B \rightarrow d$

And parse the sentence "bdc" and "dd".

(ii) Mention in detail any 4 issues in storage organization. (4)

Or

(b) (i) Write down the algorithm to eliminate left-recursion and left-factoring and apply both to the following grammar (8)

$E \rightarrow E+T \mid E-T \mid T$

$T \rightarrow a \mid b \mid (E)$

(ii) Give a syntax-directed definition to differentiate expressions formed by applying the arithmetic operators + and * to the variable x and constants ; expression : $x * (3 * x + x * x)$. (8)

13. (a) For the given program fragment $A[i, j] = B[i, k]$ do the following :

(i) Draw the annotated parse tree with the translation scheme to convert to three address code (6)

(ii) Write the 3-address code (6)

(iii) Determine the address of $A[3,5]$ where, all are integer arrays with size of A as 10 × 10 and B as 10 × 10 with k =2 and the start index position of all arrays is at 1. (Assume the base addresses) (4)

Or

(b) (i) Apply Back-patching to generate intermediate code for the following input.

```
x : 2 + y ;
if x < y then x : = x + y ;
repeat
        y : = y * 2 ;
        while x > 10 do x : = x/2
until x < y
```

Write the semantic rule and derive the Parse tree for the given code. (12)

(ii) What is an Activation Record? Explain how its relevant to the intermediate code generation phase with respect to procedure declarations. (4)

14. (a) (i) Write the Code Generation Algorithm using Dynamic Programming and generate code for the statement x = a / (b – c) — s*(e + f). [Assume all instructions to be unit cost] (12)

(ii) What are the advantages of DAG representation? Give example. (4)

Or

(b) (i) Write the procedure to perform Register Allocation and Assignment with Graph Coloring. (8)

(ii) Construct DAG and optimal target code for the expression $x = ((a+b)/(b-c))-(a+b)*(b-c)+f$. (8)

15. (a) Perform analysis of available expressions on the following code by converting into basic blocks and compute global common sub expression elimination

(i) i : = 0
(ii) a : = n _ 3
(iii) IF i < a THEN loop ELSE end
(iv) LABEL loop
(v) b : = i _ 4
(vi) c : = p + b
(vii) d : = M[c]
(viii) e : = d _ 2
(ix) f : = i _ 4
(x) g := p + f
(xi) M[g] : = e
(xii) i : = i +1
(xiii) a : = n _ 3
(xiv) IF i < a THEN loop ELSE end
(xv) LABEL end. (16)

Or

(b) (i) Explain Loop optimization in detail and apply it to the code in 15 (a). (10)

(ii) What are the optimization techniques applied on procedures calls? Explain with example (6)