Roll No. | | | | | | | | | | |

B.E / B.Tech (Full Time) DEGREE END SEMESTER EXAMINATIONS, APR/ MAY 2019

COMPUTER SCIENCE AND ENGINEERING

VI Semester

CS 8602 – Compiler Design

(REGULATION 2012)

Time: 3 Hours | Answer ALL Questions | Max. Marks 100

PART-A (10 x 2 = 20 Marks)

1. Define lexical analysis
2. Consider the grammar: S -> aSbS | bSaS | €. Show that the grammar is ambiguous for the sentence **abab**
3. What are Inherited and synthesized attributes?
4. How back patching process is used to avoid the problem of generating code in single pass?
5. What is register allocation and assignment?
6. Draw the expression tree labeled with Ershov Numbers for the expression: (a-b)+e*(c+d)
7. Define basic blocks and flow graph
8. Comment on Forward flow and backward flow problems
9. Define Affine Partitioning and blocking
10. What is data dependence?

Part – B (5 x 16 = 80 marks)

11.(a) (i) Construct LALR Parsing Table for the following grammar (10)

S → Aa | bAc | Bc | bBa

A →d

B →d

(ii) Construct Predictive Parsing Table for the following Grammar:

S -> (L)|a , L->L,S|S (6)

12.(a) (i) Translate the expression – (a+b)*(a+b)-(a+b)*d into a sequence of three address statements and explain the representation with Quadruple, triple and Indirect triple (10)

(ii) Draw the dependency graph and indicate the order of evaluation from the topological sort of " **real id1,id2,id3**" (6)

(OR)

(b) Write down the semantic rules and label them using Back patching technique. Assume the address of the first instruction generated is 100. (16)

a= =b && (c= =d | | e= =f)

13. (a) (i) What is tree translation scheme and Generate code by Tree rewriting for Intermediate code : **a[i]=b+1 and explain** (6)

(ii) Generate optimal code using Dynamic Programming technique for the assignment statement **x := (a+b * c) and explain.** Assume unit instruction cost (10)

(OR)

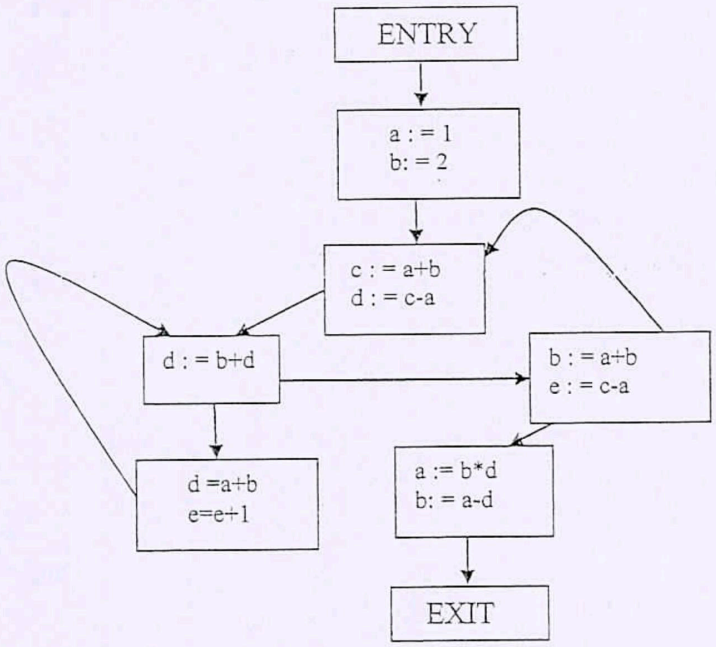
13. (b) Draw labeled expression tree for the Expression **(a-b)+e * (c+d)** and Generate optimal code from this tree with only two registers available and explain (16)

14. (a) (i) Explain constant propagation and partial redundancy elimination with suitable example (10)

(ii) Explain in detail about peephole optimization techniques with suitable example (6)

(OR)

14. (b) For the following flow graph, compute the definitions reaching each and every block and explain the data flow analysis (16)



15.(a).(i) Draw and explain about Iteration space, Data space, Processor space and mapping among them for the following program : (16)

```
float z[50];
for(i=0;i<5;i++)
z[i+5]=z[i];
```

(OR)

15.(b).(i) Explain Data Re-use with its various types. (10)

(ii) What is the need for Parallelism? Explain about loop level parallelism? (6)

Roll No.

B.E /B.Tech (Full Time) END SEMESTER EXAMINATIONS, Apr / May 2017

Computer Science and Engineering

VI Semester

CS 8602 COMPILER DESIGN

(Regulations 2013)

Time: 3 Hours | Answer ALL Questions | Max. Marks 100

PART-A (10 x 2 = 20 Marks)

1. Write the lex program's regular expression to denote email addresses and show the sequence of moves for the acceptable email address strings preetham_guru@citibank.co.in and preetham.guru@citibank.com
2. State the principles that apply to errors found by syntax analyzers, regardless of parsing technique used.
3. For an exclusive-OR (XOR) logical operator and the corresponding syntax as shown here: E → E1 ^ E2; Write an SDT scheme to generate Intermediate Representation for this XOR operator using the Arithmetic-based representation of Boolean expression.
4. What is Static Single Assignment?
5. What two forms of the addition instruction can be represented with one template on tree rewrite rules? Also, mention the template.
6. The lexically scoped language program that allows for dynamic arrays as is the case with the array `v` in the blocked region labeled in B1 as given below. Write which sections of the AR are used for holding dynamic arrays and show the layout of the AR for the procedure F with the various fields for input arguments and local variables.

```
    procedure F (x: integer, z: integer) {
B0:   {
          integer a, b;
          ... assign value to a
B1:       {
              integer v(a), b, x;
B2:           {
                  integer x, y(8);
                  ...
              }
          }
B3:       {
              integer z, w;
              ...
          }
      }
  }
```

7. Specify the data flow framework **D, V, Λ** and **F** for Reaching Definitions.
8. Write the comparison between Monotone and Distributive Frameworks.
9. What is data reuse and dependence? What are the reuse types?
10. Define Data Dependence of Array Accesses.

Part – B (5 x 16 = 80 marks)
(Question No.11 is Compulsory)

11. Given the following CFG grammar G = ({S,A,B}, S, {a, b, x), P) with Production P:

(1) S → A
(2) S → xb
(3) A → aAb
(4) A → B
(5) B → x

For the above grammar answer the following questions:

i) Compute the set of LR(1) items. 5

ii) Construct the corresponding LR(1) parsing table. 5

iii) Would this grammar be LR(0)? Why or why not? (Note: you do not need to construct the set of LR(0) items) 2

iv) Show the trace of stack content, the input and the rules used during parsing for the input string w = **axb$** 4

12. a) i) Develop an SDT scheme to generate code using the back-patching technique for a repeat – until loop. Develop a similar scheme as that of *do-while* construct using the production below and also taking into account *continue* and *break* statements. See that your solution works for the case of nested loops and *break* and *continue* statements at different nesting levels also. 8

(1) S → **repeat** M_1 L **until** M_2 E;
(2) S → **continue;**
(3) S → **break;**
(4) L_1 → S ; M_2 L_2
(5) L → S
(6) M1 → ε
(7) M2 → ε

ii) Apply the above SDT scheme and generate three address code for the following program statements: 8

```
x = 2+y
repeat
        y = y*2
        while x>10 do x = x/2
until x<y
```

(OR)

b) i) Write the SDT rules to convert array to 3-addresss code 8

ii)Draw the annotated parse tree with relevant SDT scheme for the following statement: **A = B [i] [j] + C [D [i]] [D [k]] + B [i] [j]** 8

Integer values held at **B[i, j]=23, D[i]=30, D[k]=14, C[30,14]=106**
Sizes of the arrays are: **B =80x80, D =80, C=50x80; i=j=k=25**
Assume base address for arrays B, C and D determine their exact array addresses

13. a) i) Generate optimal code using Dynamic Programming technique for the statement: 10
x = (a / (b + c) – d) * e. Assume instructions and their associated costs as given below:

Instruction	Cost
Ri = Ri op Rj	1
Ri = Ri op Mj	2
Mi = Mi op Mj	3
Ri = Rj	1
Ri = Mj	2
Mi = Rj	2

ii) Write the algorithm to generate machine code from labelled expression tree with insufficient supply of registers. 6

(OR)

b) i) Write the algorithm for register allocation. 5
ii) Construct the register-interference graph for the flow and live ranges graph given below in Fig 1. 6
ii) Determine the minimal number of registers for allocating all the variables from the flow graph in Fig 1. 5

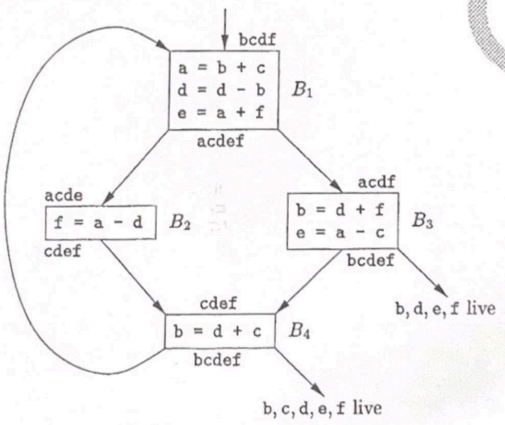


Fig.1

14. a) i) Write the steps to perform optimization over the loop on Induction variable with instruction's strength reduction and loop invariant code motion. 8
ii) Optimize the code given below performing loop invariant code motion and strength reduction on induction variables and eliminating all the induction variables. 8

```
        i = 0;
L1:
        if (i >= 100) goto L2;
        t1 = i * 4;
        t2 = t1 + A;
        t3 = 2 * i;
        *t2 = t3;
        i = i + 1;
        goto L1;
L2:
```

(OR)

b) Perform:

(i) Local and Global Common Subexpression Elimination followed by 6

(ii) Local and Global Copy Propagation for the flow graph given in Fig2. Show the trace of iterative data flow analysis process for GCSE and GCP. 6

(iii) Apply suitable other optimizations and mention them clearly. Finally depict the flow graph after applying all optimizations. 4

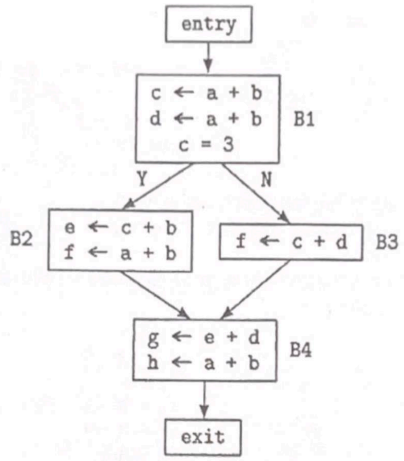


Fig 2.

15. a) i) To solve space partitions in Synchronization Free Parallelism write a simple code generation algorithm to execute partitions of a program sequentially. 6

ii) Rewrite the code given below, to a single loop. Rewrite the loop in terms of number of processors 'p' so the code can be partitioned among 100 processors, with iteration p executed by *'processor p'*. 10

```
for (i=0; i<100; i++)
    a[ i ] = 2 * a[ i ];
for (j=0; j<100; j++)
    a[ j ] = a[ j ] + 1;
```

(OR)

b) i) Consider the following code, use GCD test to analyze and list all four types of data dependencies. 8

```
for (i = 20; i<100; i++)
{   A[ 3i ] = A[ 2i - 1 ] + 4;
    A[ 4i + 2 ] = A[ 6i + 7 ];
}
```

ii) What are the types of Primitive Affine Transforms? Explain each type in detail. 8

Department of Computer Science and Engineering, CEG
Anna University, Chennai - 600025
Assessment II

50 Minutes | CS 8602 COMPILER DESIGN | 23-2-2017
Answer ALL Questions | 20 Marks

B.E CSE 'H' VI

PART A [3 x 2 = 6]

1. Give examples for inherited and synthesized translations.
2. Define activation tree and what is meant by an activation of a procedure?
3. What is the cost of the following set of instructions?

```
LD R1, c
LD R2, j
MUL R2, R2, R1
ST  a(R2), R1
```

PART B [4 + 10 = 14]

4. Write the target program sequence for a sample call and return for static allocation and stack allocation.

5. a. Translate the following assignment statement into three-address code using the translation scheme and represent it as a parse tree

A [i, j] := B [i, j] + C [A [k, m]]

Determine the address of A [10, 8] , B[10, 8], C [A[6,6]], given that A is an array of size 20 * 20, B of size 10 * 10, C of size 40. All are integer arrays **(10 marks).**

(OR)

5. b. Using *Backpatching* of Boolean Expression and control statements translation technique convert the following if condition to three-address code statements. Assume start address at 100. Draw the parse tree. **(10 marks)**

```
a = c * d
if ((~ (a>b)) && (c<=d)  || ( f != g))
        a = -f * a
else
        a =  b++
p = a + s
```

19/10/13

Roll. No								

B.E / B.Tech. (Full Time) DEGREE END SEMESTER EXAMINATIONS, NOV/DEC 2013

VI Sem CSE BRANCH

CS 9353 – Principles of Compiler Design

(REGULATIONS 2008)

Time: 3 hrs **Max. Marks: 100**

Answer ALL Questions

Part – A (10 x 2 = 20 Marks)

1 Define Analysis and Synthesis of compilation

2 Differentiate Top down parser and Bottom up parser.

3 Give examples for inherited and synthesized translations.

4 Translate the arithmetic expression **a* - (b+c)** into a) Postfix notation b) 3-address code

5 Brief about the techniques for parameter passing

6 How do you calculate the cost for the following instructions:

```
MOV R1,M
SUB  E(R0),*4(R1)
```

7 Define Back patching

8 How can you find the leaders in basic block?

9 Comment on Next-use information

10 Design a syntax directed translation scheme for constructing syntax tree from the arithmetic expressions containing only exponentiation operator and identifiers using unambiguous grammar.

Part – B (5 x 16 = 80 Marks)

11 Find the predictive parser for the given grammar and parse the sentence: **a+b*c** **(16)**

E -> E +T|T
T ->T* F|F
F-> (E)|id

12. (a) (i) Translate the following assignment statement into three-address code using the translation scheme and represent it as a syntax tree **(16)**

$$A[i, j] := B[i, j] + C[A[k, m]]$$

Determine the address of A [10, 8] , B[10, 8], C [A[6,6]], given that A is an array of size 20 * 20, B of size 10 * 10, C of size 40. Assume the base address.

(OR)

12. (b) (i) Describe any two storage allocation strategies in detail. **(10)**

(i) How does the parameter passing technique call-by-value implemented? **(6)**

13. (a) (i) Explain the Back patching process for the following expression: **(10)**

p< q or r < s and t< u

(ii) Define activation tree and what is meant by an activation of a procedure? **(6)**

(OR)

13. (b) Explain about the methods of translating Boolean Expressions **(16)**

14. (a) (i) Explain the various issues involved in the design of code generation **(8)**

(ii) Describe in detail about run-time storage management **(8)**

(OR)

14. (b) (i) Construct a DAG for the following basic block and generate code by labeling the nodes based on the gencode() algorithm discussed. **(8)**

d : = b*c
e: = a+b
b : = b*c
a : = e-d

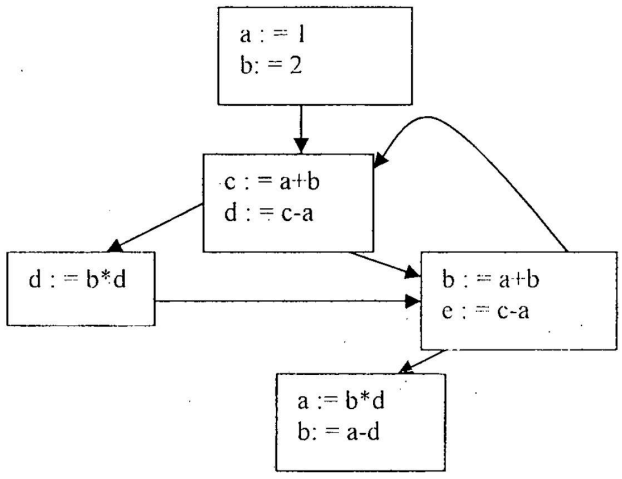
(ii) Generate optimal code using Dynamic Programming technique for the assignment statement, x := (a/b – c) /d. Assume unit instruction costs **(8)**

15. (a) (i) Discuss peep hole optimization and the transformations on peep hole optimization **(8)**

(ii) Explain the Optimization of Basic block **(8)**

(OR)

15. (b) (i) For the following flow graph, compute the definitions reaching each and every block **(16)**



B.E / B.Tech (Full-Time) Degree End Semester Examinations, April / May 2013
Anna University, Chennai

Computer Science and Engineering
Sixth Semester

CS9353 PRINCIPLES OF COMPILER DESIGN

(Regulations 2008)

Time 3 Hrs **Answer all Questions** **100 Marks**

PART A – (10 X 2 = 20 Marks)

1. Name few tools that could be used for the various phases of the compiler
2. Describe all viable prefixes for the following grammar.

   S → + SS | * SS | (S) | a
3. The following grammar is for expressions involving operator '+' for integer and floating point operands. Floating point numbers are distinguished by having a decimal point. Give an SDD to determine the type of each term T and expression E.

   E → E + T | T

   T → num . num | num
4. Consider the production **D → proc id; D; S** for recognizing a procedure. Rewrite the production and write the semantic actions so that the declarations in this procedure will be stored in a new symbol table or in new scope.
5. Write semantic rules to handle the Boolean 'exclusive-or' operator.
6. What is the need for next-use information?
7. What are the advantages of representing the input using DAG?
8. Justify the need for tree-rewriting procedure for generating code using template based method.
9. What are dominators and how loops are identified from flow graphs using dominators?
10. What type of optimization computes the available expressions?

PART B – (5 x 16 = 80)

11. i. Let synthesized attribute ***val*** give the value of the binary number generated by S in the following grammar. For example on input 101.101 ***S.val*** = 5.625. Use synthesized attribute to determine ***S.val***. [8]

S → L . L | L

L → L B | B

B → 0 | 1

ii. Describe the storage allocation strategies used at run time for static allocation, stack allocation and heap allocation. [8]

12. a. i. With reference to the phases of the compiler, identify the sequence of steps that would take place for the input ans := **(a + - b * 6.3) ^ 2 ^ (1 – k)** [8]

ii. Write the algorithm and construct the predictive parsing table for the grammar

S → L = R | R

L → *R | id

R → L

Find whether this grammar is LL(1) or not and give reason for your answer. [8]

(OR)

b. Write the algorithm and construct SLR parsing table for the following context free grammar. Check whether the string id + id id * is a valid string. [16]

E → E + T | T

T → T F | F

F → F * | id

13. i. Write the translation scheme for translating assignment statements having scalar variables and array references to three-address statements. [8]

ii. Using the above translation scheme construct an annotated parse tree for the following assignment statement and generate the intermediate language representation.

A[I, J + B[K]*L] := C[I*J, K+L] * D [8]

where A and C are integer arrays of sizes 20 x 50, B is an integer array of size 10, integer size 4 bytes, and first index position for all the arrays and all dimensions is 1.

(OR)

b. Consider the following program segment.

```
for ( i = 0; i < 10 ; i ++)
begin
        if ( ti < tj ) then begin  tk = ti;  i = i + 1;  end
        else  begin  tk = tj; j = j + 1;  end

end
```

i. Write the production rules needed for recognizing the above program block along with the translations using back patching for each rule. [8]

ii. Construct annotated parse tree and generate three-address code as intermediate language for the grammar given. [8]

14. a. i. Discuss any four important issues in the code generation phase. [6]

ii. Apply the code generation algorithm for the following basic block and find the object code produced for a machine with exactly two registers namely R1 and R2. The next use information, address descriptors, register descriptors should be taken into account while looking for registers to carry out the computation. Initially the register descriptors for all registers are empty and it is not required to keep any variable live on exit from the basic block. [10]

d= b * c; e = a + b; b = b* c; f = a [i];

(OR)

b. i. Construct a DAG for the following basic block and find the optimal ordering of instructions for code generation which requires minimal number of registers. [10]

a [i] = b; *p = c; d = a[j]; e = * p; * p = a [i]

ii. Consider the following expression a: = b+c*d-e/f * g , and generate three address code for the same. Construct a register interference graph and determine the minimum number of registers required to evaluate this expression. [6]

15. a. i. Define peephole and explain all the transformations carried out in peephole optimization. [8]

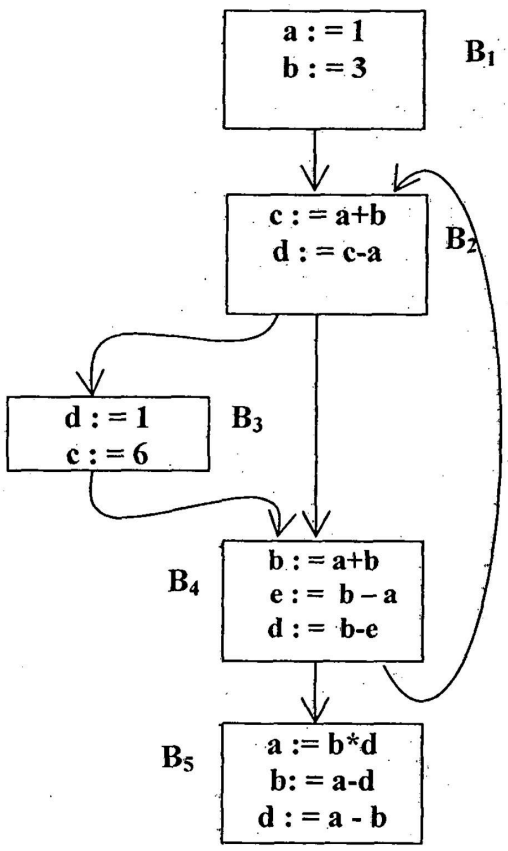
ii. Optimize the following code using all known optimization techniques by explaining in a few words what each optimization refers to. [8]

```
    dp = 0; i = 0;
L:  t1 = i * 8;
    t 2 = A [ t1]; t3 = i * 8 ; t4 = B [t3 ]; t5 = t2 * t4; dp = dp + t5; i = i + 1;
    if ( i < n ) goto L
```

(OR)

b. i. Write the data flow equations for basic blocks and derive the data flow equations for the most familiar three program control constructs [8]

ii. Compute the definitions in and out of each basic block for the flowgraph given here. [8]



Roll No.									

B.E. / B.Tech. (Full Time) DEGREE END SEMESTER EXAMINATIONS, APR / MAY 2012

COMPUTER SCIENCE AND ENGINEERING BRANCH

SIXTH SEMESTER

CS9353 – PRINCIPLES OF COMPILER DESIGN

(REGULATION 2009)

Time : 3 Hours **Max Mark : 100**

Answer ALL Questions

PART A – (10 x 2 = 20 marks)

1. What is panic mode error recovery?
2. Write regular definition for the language that has all strings of letters that contain the five vowels in order
3. Write the syntax directed translation scheme for the input expression (4 + 7.5 * 3) / 2 and construct annotated parse tree
4. Write type expressions for the C -function **fun** that has two character arguments and returns a pointer to a character.
5. Generate short circuit code for: **if (not (a or b) and c) stmts;**
6. Generate the 3 - address code for the statement in the above question in quadruple and indirect triplet.
7. What are address descriptors and register descriptors
8. Generate target code for the instruction: *p + r = *q (assume enough registers are available)
9. What is code hoisting? Give its usage.
10. How unreachable code is identified ?

PART B – (5 x 16 = 80 marks)

11. i. Explain in detail the different parameter passing modes. (8)
    ii. Tabulate the comparison of storage allocation strategies used at run time. (8)

12. (a) i. Find the non recursive predictive parsing table, for the following grammar;

G -> S
S -> (L) | a
L -> L ; S | S (8)

12. (a) ii. Write the non recursive predictive parsing algorithm and parse for the input string (a;a) (8)

OR

12. (b) i. Construct the SLR Parsing table and parse the string "((( )) " for the grammar Y -> ( ) | (Y) (10)

12. (b) ii. Explain the front end of a compiler with a suitable arithmetic expression and mention the need for grouping the phases in a compiler. (6)

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

i. Write the syntax directed translation scheme to convert to 3-address code (4)
ii. Draw the annotated parse tree with the translation scheme (4)
iii. Write the 3-address code (4)
iv. Determine the address of **A[4, 5]** where, all are integer arrays with size of A as10 x10, B as 10x10x10 and C as 10 and the start index position of all arrays is at 1. (Assume the base addresses) (4)

OR

13. (b) Consider the following program fragment:

```
begin
while a > b do
        begin
                x = y + z
                a = a - b
        end
        x = y - z
end
```

i. Determine a syntax directed translation scheme to convert to 3-address code using backpatching (6)
ii. Draw the annotated parse tree with the generated translation scheme (6)
iii. Write the 3-address code assuming a base address (4)

14. (a) i. Mention the issues related to the generation of code and explain any four in detail. (8)

14. (a) ii. Write the algorithm to generate code with shortest instruction sequence to evaluate statements in a basic block and apply the same to the following statements in a basic block:

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

(8)

OR

14.(b) i. Generate optimal code using Dynamic Programming technique for the assignment statement: x = (a / b + c) / (d – e). Assume the instructions and their associated cost as mentioned in Table 1. (8)

Table1. Instruction and its cost

Instruction	Cost	Instruction	Cost	Instruction	Cost
Ri = Ri op Rj	1	Ri = Mj	2	Mi= Mi op Mj	4
Ri= Ri op Mj	2	Mi = Rj	2	Ri=Rj	1

With Answers.

ANNA UNIVERSITY

B.E. DEGREE END SEMESTER EXAMINATIONS, APRIL / MAY 2011

COMPUTER SCIENCE AND ENGINEERING

Semester VI

CS9353 PRINCIPLES OF COMPILER DESIGN

Time 3 Hrs *Answer all Questions* *100 Marks*

PART A – (10 X 2 = 20 Marks)

100: if i<n goto 102
101: goto 113
102: if j<m goto 104
103: goto 113
104: if ti<tj goto 106
105: goto 109
106: tk = ti
107: i = i+1
108: goto 11
109: tk = tj
110: j = j+1
111: k = k+1
112: goto 100
113: ...

1. Comment on the target languages of interpreters and compilers? Should the target language always be a low level language?
2. How does a scanner recover from lexical errors?
3. Design an unambiguous syntax directed translation scheme for constructing syntax tree from the arithmetic expression containing only exponentiation operator.
4. What is meant by call-by-reference? Identify any one constraint in terms of actual argument.
5. Generate triplets for **a[i] := b; c := d[j].** When triplets are preferred over quadruples?
6. Consider the production **D → proc id; D; S** for recognizing a procedure. Rewrite the production and write the semantic actions so that the declarations in this procedure will be stored in a new symbol table or in new scope.
7. What are the steps to be carried out while collecting the next use information of identifiers from **x := y op z** in backward scan?
8. What is meant by contiguous evaluation of expressions in dynamic programming code generation algorithm? State it's advantage.
9. What are dominators and how loops are identified from flow graphs using dominators?
10. What is meant by available expressions? What type of optimization computes the available expressions?

PART B – (5 x 16 = 80)

11. i. Let synthesized attribute *val* give the value of the binary number generated by S in the following grammar. Design a syntax directed translation scheme for computing the decimal equivalent for the given binary number. For example on input 101.101 *S.val* = 5.625. [8]

$S \rightarrow L_1 . L_2 \mid L$

$L \rightarrow L_1 B \mid B$

$B \rightarrow 0 \mid 1$

ii. Describe the storage allocation strategies used at run time for static allocation, stack allocation and heap allocation. [8]

12. a. i. Construct the predictive parsing table for the grammar.

$S \rightarrow L = R \mid R$

$L \rightarrow *R \mid id$

$R \rightarrow L$

Find whether this grammar is LL(1) or not and give reason for your answer. [8]

ii. List down the activities taking place in the machine independent phases of compilers. Identify the output of those phases for the input ans := (a + - b * 5.3) ^ 2 ^ (1 – f) [8]

(OR)

b. i. Construct SLR parsing table for the following context free grammar [10]

$E \rightarrow E + T \mid T$

$T \rightarrow T\ F \mid F$

$F \rightarrow F^* \mid 0 \mid 1$

ii. Explain the parsing algorithm used by LR parsers and parse any one valid string from the language of the above grammar using the SLR parsing table constructed. [6]

13. a. i. Write the translation scheme for translating assignment statements having scalar variables and array references to three-address statements. [8]

ii. Using the above translation scheme construct an annotated parse tree for the following assignment statement and generate the intermediate language representation.

A[I, J + B [K]] := C[I * J, K + L] * D + - E [8]

where A and C are integer arrays of sizes 20 x 50, B is an integer array of size 10, integer size 4 bytes, and first index position for all the arrays and all dimensions is 1.

(OR)

b. Consider the following program segment.

while (i < N and j < M)
begin
if (ti < tj) then begin tk = ti; i = i + 1; end
else begin tk = tj; j = j + 1; end
k = k + 1;
end

i. Write the production rules needed for recognizing the above program block along with the translations using back patching for each rule. [8]

ii. Construct annotated parse tree and generate three-address code as intermediate language. [8]

14. a. i. Discuss any four important issues in the code generation phase. [8]

ii. Apply the code generation algorithm for the following basic block and find the object code produced for a machine with exactly two registers namely R1 and R2. The next use information, address descriptors, register descriptors should be taken into account while looking for registers to carry out the computation. Initially the register descriptors for all registers are empty and it is not required to keep any variable live on exit from the basic block. [8]

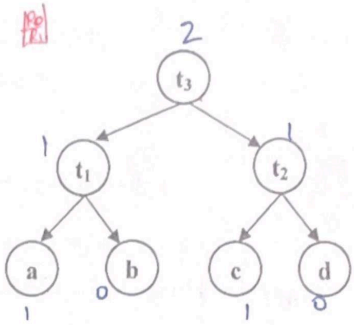
d = b + c; e = a + b; b = b + i; a = e + d; g = e + b;

(OR)

b. i. Construct a DAG for the following basic block and find the optimal ordering of instructions for code generation which requires minimal number of registers. [8]

$t_1 = a + b$; $t_2 = t_1 - c$; $t_3 = d + e$; $t_4 = t_2 * t_3$; $t_5 = t_4 - e$; $t_6 = t_1 + t_4$; $t_7 = t_6 * t_5$;

ii. Consider the DAG given here for some basic block. Compute the labels for each node using label computation algorithm. Generate object code by applying the code generation algorithm meant for labeled tree for a machine with registers R0 and R1 in the register stack, R0 on top and enough number of temporaries in temporary variables stack with T0 on top. [8]



15. a. i. Define peephole and explain all the transformations carried out in peephole optimization. [8]

ii. Apply all applicable basic block optimizations in the following basic block and produce the optimal code. Identify the loop invariant computations and move them into a new basic block. [8]

L1 : a = 1; t = i * 4; b = 2 * a; c = p + q; d = a << 10;
e = p + q; f = e * d; g = q; p = p + g; h = p + q;
m = m + h; m = m + f ; i = i + 1;
if i <=10 goto L1;

(OR)

b. i. Write the data flow equations for basic blocks and derive the data flow equations for the most familiar three program control constructs [8]

ii. Compute the definitions in and out of each basic block from the flowgraph given here. [8]