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]