KINGS COLLEGE OF ENGINEERING

PUNALKULAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2010-2011 / EVEN SEMESTER

SUBJECT CODE\SUBJECT NAME: CS1352 \ PRINCIPLES OF COMPILER DESIGN YEAR/SEM:III/VI

QUESTION BANK

UNIT I- INTRODUCTION TO COMPILING

PART – A (2MARKS)

1. Define a compiler?
2. Describe the Analysis-Synthesis Model of compilation.
3. Define a symbol table.
4. What are the functions of preprocessors?
5. Define Assembly Code.
6. Define tokens, Patterns and lexemes.
7. What are the possible error-recovery actions in lexical Analyzer?
8. What are the three general approaches to the implementation of a Lexical Analyzer?
9. Define a Sentinel.
10. Describe briefly rational preprocessors with an example.
11. What are the reasons for separating the analysis phase of compiling into Lexical analysis and parsing?
12. What is the function of a loader?
13. Give the diagrammatic representation of a language processing system.
14. Explain briefly the producer-consumer pair of a lexical analyzer and parser.

15. Mention the issues in a lexical analyzer.
16. Mention some of the cousins of the compiler.
17. What are rational preprocessors?

PART - B

1. Explain in detail about the role of Lexical analyzer with the possible error recovery actions. *(16)*
2. (a)Describe the following software tools
   i. Structure Editors ii. Pretty printers iii. Interpreters *(8)*
   (b)Write in detail about the cousins of the compiler. *(8)*
3. Describe in detail about input buffering. What are the tools used for constructing a compiler? *(16)*
4. (a)Explain the functions of the Lexical Analyzer with its implementation. *(10)*
   (b)Elaborate specification of tokens. *(6)*
5. (a)What is a compiler? Explain the various phases of compiler in detail, with a neat sketch. *(10)*
   (b) Elaborate on grouping of phases in a compiler. *(6)*
6. (a) Explain the various phases of a compiler in detail. Also write down the output for the following expression after each phase *a: =b*c-d*. *(8)*

   (b) What are the phases of the compiler? Explain the phases in detail. Write down the output of each phase for the expression *a: = b + c * 50*. *(8)*

UNIT II- SYNTAX ANALYSIS

PART – A (2MARKS)

1. Draw a NFA for a* | b*.
2. What do you mean by Handle Pruning?
3. Define LR (0) items.
4. What do you mean by viable prefixes?
5. Define handle.
6. What are the algebraic properties of regular expressions?
7. What is finite automata?
8. What are the goals of error handler in a parser?
9. What is an ambiguous grammar? Give an example.
10. What is phrase level error recovery?
11. What are the disadvantages of operator precedence parsing?
12. What is a predictive parser?
13. Eliminate left recursion from the following grammar

    A->Ac/Aad/bd/c.
14. What is LL (1) grammar? Give the properties of LL (1) grammar.
15. Give the algorithm for Left Factoring a Grammar.
16. What is Left Recursion? Give an example for eliminating the same.

PART – B

1. What is FIRST and FOLLOW? Explain in detail with an example. Write down the necessary algorithm. *(16)*
2. Construct Predictive Parsing table for the following grammar:

   S -> (L) / a

   L -> L, S/S

and check whether the following sentences belong to that grammar or not.

(i) (a,a)

(ii) (a, (a , a))

(iii) *(a, ((a , a), (a , a)))* *(16)*

3. (a) Construct the predictive parser for the following grammar:

S -> (L) | a

L -> L,S | S. *(12)*

(b) Construct the behaviour of the parser on sentence (a, a) using the grammar:

S -> (L) | a

L -> L,S | S. *(4)*

4. (a) Check whether the following grammar is SLR (1) or not. Explain your answer with reasons.

S-> L=R

S->R

*L->*R*

L->id

R->L (8)

(b) For the grammar given below, calculate the operator precedence relation and the precedence functions.

*E -> E + E | E – E | E * E | E / E | E ^ E | (E) | -E | id* *(8)*

5. Check whether the following grammar is a LL(1) grammar

S -> iEtS | iEtSeS | a

E -> b

Also define the FIRST and FOLLOW procedures. *(16)*

6. (a) Consider the grammar given below.

E->E+T

E->T

*T->T*F*

T->F

F->(E)

F->id.

Construct an LR Parsing table for the above grammar. Give the moves of LR parser on *id*id+id*.

(12)

(b) What is a shift-reduce parser? Explain in detail the conflicts that may occur during shift- reduce parsing. *(4)*

UNIT III-INTERMEDIATE LANGUAGES

PART – A (2MARKS)

1. How would you represent the following equation using the DAG, *a: =b*-c + b*-c*. What is the purpose of DAG?
2. What is the intermediate code representation for the expression *a or b and not c ?*
3. How would you map names to Values?
4. What are the various methods of implementing three address statements?
5. Suggest a Suitable approach for completing hash function.
6. What are the methods of representing a syntax tree?
7. Give the Syntax directed definition of if-else statement.
8. What is backpatching?
9. What are the applications of DAG?
10. Define marker non-terminals with an example.
11. Why are quadruples preferred over triples in an optimizing Complier?
12. Give the triple representation of a ternary operation x:= y[i]
13. Give the Semantic rules for the production S → while E do S1.

14. Let A be a 10x20 array with $low_1=low_2=1$. Therefore $n_1=10$ and $n_2=20$. Take w to be 4. Give the annotated parse tree for the assignment x:=A[y,z]
15. What is short-circuit or jumping code?

PART-B

1. How would you generate the intermediate code for the flow of control statements?Explain with examples. *(16)*
2. (a)What are the various ways of calling procedures? Explain in detail. *(8)*

   (b)What is a three-address code? Mention its types. How would you implement the three address statements? Explain with examples. *(8)*
3. How would you generate intermediate code for the flow of control statements?

   Explain with examples. *(16)*
4. (a)Describe the method of generating syntax-directed definition for Control statements.

   (8)

   (b)Give the semantic rules for declarations in a procedure. *(8)*
5. (a) How Back patching can be used the generate code for Boolean expressions and flow of control statements. *(8)*

   (b)Explain how the types and relative addresses of declared names are computed

   and how scope information is dealt with.

   (8)
6. (a) Describe in detail the syntax-directed translation of case statements. *(8)*

   (b) Explain in detail the translation of assignment statements. *(8)*

UNIT IV- CODE GENERATION

PART – A (2MARKS)

1. How would you calculate the cost of an instruction?
2. What is an activation record for a procedure?
3. What is a Basic Block?
4. What are the steps involved in partitioning a Sequence of three address statements into basic blocks?
5. What are the limitations of static allocation?
6. What is dead code elimination?
7. Give the primary structure preserving transformations on Basic Blocks.
8. Draw the diagram of the general activation record and give the purpose of any two fields.
9. What is stack allocation?
10. What are dags and how are they useful in implementing transformations on basic blocks?
11. What is peephole optimization?
12. Mention the transformations that are characteristic of peephole optimizations.
13. What are machine idioms?
14. Construct the dag for the following basic block:

    *d: = b * c*

    e: = a + b

    *b: = b * c*

    a: = e - d

15. Give the applications of dags.
16. What are register descriptors?
17. Briefly describe address descriptors.

PART-B

1. (a) Explain the issues in design of code generator. *(8)*
   (b) Explain peephole optimization. *(8)*
2. (a) Discuss run time storage management of a code generator. *(8)*
   (b) Explain DAG representation of the basic blocks with an example. *(8)*
3. (a) Explain the simple code generator with a suitable example. *(8)*
   (b) Describe about the stack allocation in memory management. *(8)*
4. (a) What are the different storage allocation strategies? *(8)*
   (b) What are steps needed to compute the next use information? *(8)*
5. (a) Write detailed notes on Basic blocks and flow graphs. *(8)*
   (b) How would you construct a DAG for a Basic block? Explain with an example. *(8)*

UNIT V- CODE OPTIMIZATION

PART – A (2MARKS)

1. What are called optimizations and what is an optimization compiler?
2. Mention the criteria for code-improving transformations.
3. Mention the function preserving, code improving transformations.
4. What is code motion? Give an example.
5. What is constant folding?
6. What are induction variables? What is induction variable elimination?
7. What is a cross-compiler? Give an example.
8. What are the properties of optimizing compilers?

9. Mention the different storage allocation strategies.
10. Mention the limitations of static allocation.
11. What are calling sequences and give brief notes on its types.
12. When does a dangling reference occur? Give its impact on programs.
13. Give the situations in which stack allocation can not be used.
14. Mention the different types of parameter passing.
15. What are the two approaches of implementing Dynamic Scope? Give the difference between the two.

PART – B

1. (a) Explain the principle sources of optimization in detail. *(8)*
   (b) What are the various ways of calling procedures? *(8)*
2. (a) Discuss about the following:
   i). Copy Propagation ii) Dead-code Elimination and iii) Code motion *(6)*
   (b) Describe in detail about the stack allocation in memory management. *(10)*
3. (a) Write about Data flow analysis of structural programs. *(8)*
   (b) Describe the various storage allocation strategies. *(8)*
4. (a) Describe in detail the source language issues. *(8)*
   (b) Explain in detail access to nonlocal names. *(8)*
5. (a) Elaborate storage organization. *(8)*
   (b) Write detailed notes on parameter passing. *(8)*
6. (a) Explain optimization of basic blocks. *(8)*

(b) Explain the various approaches to compiler development.

(8)