Compiler Design Aug 1996

Part – A

1 a) What are the different phases of a compiler? Explain briefly with the help of a neat diagram.

b) For the following Pascal keywords write the state diagram and also write program segments to implement the lexical analyzer for these keywords. 12.

BEGIN	IF
END	THEN
ELSE	

2 a) Write the derivations and parse tree for the string (id + id) * (id + id) using the following grammar.

$E \rightarrow E + T/T$

$E \rightarrow E * F/T$

$F \rightarrow (E)/id$ 10.

b) Define the term handle. Show how operator precedence relations can be used in recognizing handles. Use an appropriate example. 10.

3 a) For the following grammar compute the FIRST and FOLLOW for each of the nonterminals.

$E \rightarrow TE'$

$E' \rightarrow +E/\varepsilon$

$T \rightarrow FT'$

$T' \rightarrow \&T/\varepsilon$

$F \rightarrow PF'$

$F' \rightarrow *F/\varepsilon$

$P \rightarrow (E)/a/b/\varepsilon$ 12

b) Explain SLR parsing technique with an example. 8.

4 a) Write the three-address codes and quadruples for the following program segments. Assume two bytes are required to store each element of the array. 10.

```
SUM = 0;
I : = 0;
While I ≤ 20 do
Begin
SUM = SUM + A[I];
I : = I + 1
End;
```

b) Explain with an example the simple translation scheme for an assignment statement.10.

5 a) Explain with examples the indirection that may be used in the symbol table. 8.

b) Explain how scope information is represented by symbol table organization using hash tables. 12.

Part - B

6 a) Explain with an example each the different sources of errors. 10.

b) Explain how error detection and recovery are done in LL(1) parsing. 10.

7 Consider the following matrix multiplication routine; 20

```
begin
for i : = 1 to n do
for j : = 1 to n do
```

```
C[i,j] : = 0;
for i : = 1 to n do
for j : = 1 to n do
for k : = 1 to n do
C[i,j] : = C[i,j] + A[i,k] * B[k,j]
End;
```

i) Assuming A, B and C are allocated static storage and there are two bytes per word in a byte addressable memory, produce the three address codes for the matrix multiplication program.
ii) Partition the three address codes into basic blocks
iii) Marks the loops in the flow-graph.
iv) Move loop invariant computations out of the loop.

8 a) Explain the different Problems involved in code generation. 9.
b) Describe the code generation algorithms. 6.
c) Describe the functions GETREG( ). 5.

Compiler Design February 1997

Part – A

1. a) Briefly explain the different phases of a compiler and their functions with a neat diagram
b) Construct an NFA for recognizing the following tokens:
1) a(aa)*
2) (a+b)*a
convert the NFA to an equivalent DFA and make the token recognized by each state of DFA. 12.

2. a) Explain the following with suitable examples: 9.
1) Bootstrapping
2) Lookahead operator
3) Non-context-free language constructs?

b) Remove left recursion for the following grammar and then write recursive descent parser for the same. 11.
$E \rightarrow E * F \mid F$
$F \rightarrow (E) \mid id$

3 a) Briefly explain the working of LL(1) parser. 5.
b) For the following grammar calculate LR(0) items. 8.
$S' \rightarrow S$
$S \rightarrow AS \mid c$
$A \rightarrow aA \mid b$
c) Write SLR (1) parsing table for the grammar in Q. No 3(b) 7.

4 a) Briefly explain the differences between SLR (1), Cononical LR and LALR parsing methods. 8.

Incomplete

Compiler Design August 1997

Part –A

1 a) Explain: 8.
1) Lexical analysis phase.

2) Syntax analysis phase with examples.

b) Write transition diagram for recognizing. 12.

i) Identifier = letter (letter/digit) *

ii) Constant = (+|-| t) digit digit *

And code for each state of DFA recognizing above tokens.

2 a) construct an NFA and a transition table for DFA, for recognising following tokens.

i) 0 ii) 101 iii) 1*0t 12.

b) Discuss non-context free language. Construct with an example. 3

3 a) Give operator-precedence-parting algorithm and discuss the same with an example.8

b) Write a recursive descent parser for the grammar. 12.

$S \rightarrow a \mid \wedge \mid (T)$

$T \rightarrow T, S|S$

4 a) Discuss the working of an LR parser with a neat diagram. 8.

b) Construct sets-of-items and deterministic finite automation for the grammar. 12.

$E' \rightarrow E$

$E \rightarrow E + T|T$

$T \rightarrow T^* F|F$

$F \rightarrow id$

5 a) Discuss the following Intermediate codes: 8.

i) Postfix notation.

ii) Syntax trees.

iii) Quadruples.

iv) Triples.

b) Discuss data structures used for symbol tables. 12.

Part – B

6 a) Discuss. 10.

i) Error recovery.

ii) Semantic errors.

iii) Dynamic errors with examples.

b) Discuss error recovery in LR parsing. 10.

7 a) Generate three-address code for the following program fragment. Assume A and B are allocated static storage and four bytes for each elements of A and b. 12.

```
Begin
        PROD : = 0;
        I : = 1;
        do
        Begin
                PROD : = PROD + A[I] * B[i];
                I : = I + 1;
        End
        While I ≤ 20
End
```

b) Construct the DAG for the following basic block: 8.

1) D : = B * C
2) E : = A + B
3) B : = B * C
4) G : = A + B
5) A : = E – D
6) F : = E * B
7) IF F ≤ 10 GOTO (1)

8 a) Write short notes on any four: (5 * 4)
- i) Code-generation algorithm.
- ii) Code-notion.
- iii) Panic node.
- iv) Global register allocation
- v) Induction variables.

Compiler Design March/April 1998

Part A

1 a) What are compiler writing tools? Explain the principal aids provided by the existing compiler – compilers.
b) Briefly describe boot strapping.
c) Write the state diagram for accepting a decimal real number. Also write the program segments.

2. a) Define a parser. Explain the concepts of top-down and bottom-up parser. 6.
b) Explain the stack implementation of shift-reduce parsing. 6.
c) Define an operator precedence grammar. Consider the grammar,

$E \Rightarrow E + E$ $\quad$ $E \Rightarrow E – E$
$E \Rightarrow E * E$ $\quad$ $E \Rightarrow E \mid E$
$E \Rightarrow E \uparrow E$ $\quad$ $E \Rightarrow (E)$
$E \Rightarrow id$

Construct the operator-precedence relation table assuming the following operator precedence relations.
1) $\uparrow$ is of highest precedence and right-associative.
2) * and | are of next highest precedence and are left-associative.

3 a) Explain the concept of recursive-descent parsing with an example. 8.
b) Mention the importance of LR parsers. Write the parsing table and the moves made for the input (id*id) + (id*id) by an L parser based on the following grammar. 12.

4 a) Explain the need of intermediate code. Mention the different types of intermediate code.
b) For the following expression write the three address code, quadruples and triples. 9.
While A < C and B < D do
If A = 1 then C:=C + 1
Else While A <= D do A := A + 2
c) Explain the abstract translation scheme for the translation of assignment statements.

5 a) Explain the need of symbol table in the design of a compiler. 4.
b) Briefly describe the operations that are commonly performed and the different information stored in the symbol table. 8.
c) With the help of a neat diagram explain how scope information is represented in the symbol table organization of ALGOL. 8.

Part – B

6 a) Explain the different sources of errors with an example each. 7.
b) Explain the concept of minimum distance correction of syntactic errors. 7.
c) Explain the recovery in operator precedence parsing. 6.

7 a) Explain the principal sources of code optimization. 6.
b) Write three address codes, form basic blocks, draw flow graph, and move the loop invariant code outside the loop. Also assume that four bytes are required to store each element of the array. 8.
Min = A[0];
Max = A[0];

```
for(i=1;i<n;i++)
{
  if A[i] > Max
          Max = A[i];
  If A[i] < Min
          Min = A[i];
}
```

c) Write a note on directed acyclic graph. 6

8 a) Describe the different problems of code generation. 6.

b) Explain next-use information, register descriptors and address descriptors. 8.

c) Write the code sequence generated for the following three address codes. 6.

T := A – B
U := A – C
V := T + U
W := V + U

Compiler Design Sept/Oct 1998

Part A

1 a) Explain briefly the phases of a compiler with an example. 10.

b) Explain the implementation of lexical analyser with an example. 10.

2 a) Construct non-deterministic finite automata for the given regular expression using Thompson's algorithm. 10.
(a/b) * abb.

b) Construct DFA using the subset algorithm for the regular expression (a/b)*abb. 10.

3 a) Compute the FIRST and FOLLOW symbols for each of the non-terminals in the grammar given below after eliminating left recursion. 10.

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

b) Determine the operator precedence relations for the grammar. 10.

$E \rightarrow E + E \mid E - E * E \mid E / E \mid E \uparrow E \mid (E) \mid - E \mid id$

i) $\uparrow$ is of highest precedence and right associative.
ii) * and / are the next highest precedence and left associative and
iii) + and – are the lowest precedence and left associative.

4 a) Construct the syntax directed translation scheme that translates arithmetic expression from postfix notation into infix notation. Give annotated parse trees for the input 13–5* and 135*-. 6.

b) Translate the arithmetic expression 6.
a *-(b + C) into
i) A syntax tree.
ii) Postfix notation.
iii) Three-address code.

c) Explain the LR Parsing algorithm with an example. 8.

5 a) Explain the concept of the shift-reduce on input with the sentence $id_1 + id_2 * id_3$ 10.

b) Explain briefly the different data structure used for symbol table. 10.

Part – B

6 a) Discuss error recovery in operator precedence parsing. Trace out the behavior of the operator precedence LR and LL error-correcting parsers on input.
a) (id + (*id)) 8.
b) Explain with an example minimum distance correction of syntactic errors. 6.
c) Explain briefly the different types of errors encountered in each phase of compiler. 6.

7 a) Generate code for the following C program:

```
main( )
{
  int i;
  int a[10];
  while(i<=10)
  a[i]=0;
}
```

b) Construct the dag for the following basic block. 6.
d = b * c
e = a + b
b = b * c
a = e – d
c) Explain the need for optimization with at least two examples. 8.

8 Write short notes on the following: (4 x 5)
a) Context free grammars.
b) Top-down parsing.
c) Panic mode recovery.
d) Semantic errors.

Compiler Design March/April 1999

Part – A

1 a) Briefly explain the lexical analysis phase of a compiler and indicate the output for the each of the following statements.
i) IF(5 .EQ. MAX) GOTO 100 ii) DO 20 I = 1, 10, 2. 10.
b) Write transition diagrams for recognising C language reserved words given below; 10.
case, char, if, not, struct, switch.

2 a) Explain operator-precedence grammar and construct operator precedence relation matrix for the following operators: 12.
- / % left to right associativity, highest priority.
- = right to left associativity. Next highest priority.

b) Explain the working of a predictive parser. 8.

3 a) Compute the FIRST and FOLLOW symbols for each of the non-terminals. 12.
$E \rightarrow TE'$ $E' \rightarrow +E/\varepsilon$ $T \rightarrow FT'$
$T' \rightarrow T/\varepsilon$ $F \rightarrow PF'$ $F' \rightarrow *F/\varepsilon$ $P \rightarrow (E)\ |a|b|e$
b) Discuss LR parser technique and advantages of LR parsers. 8.

4 a) Show that the following grammar is not SLR (1) type: 10.
$S \rightarrow L = R$
$S \rightarrow R$
$L \rightarrow *R$
$L \rightarrow i\delta$
$R \rightarrow L$

b) Clearly bring out the features of following techniques: 10.
i) SLR ii) Canonical LR iii) LALR

5 a) Give quadruple, triple representation of the following: 10.
i) A[I] : = - B * C * D[j] + E
ii) F != -E * (G + H)

b) Explain how scope information is represented in case of ALGOL language. 10.

Part B

6 a) Discuss error recovery 9in operator – precedence parsing. 8.

b) Convert the following program fragment to three-adders code and optimize the same 12.

```
 Begin
   I : = 1;
12 while I <= 10 do
   Begin
           C1 : = 3.14;
           C2 : = 180.0;
           C3 : = C1/C2;
           X[I] : = X[i] * (3;
           I : = I + 1;
   End
```

7 a) Represent the following using DAG: 12.

1) S_1 : = 2 * I	2) S_2 : = addr(A) – 2	3) S_3 : = $S_2[S_1]$
4) S_4 : = 2 * I	5) S_5 : = addr(B) – 2	6) S_6 : = $S_5[S_4]$
7) S_7 : = S_3 * S_6	8) S_8 : = PROD + S_7	9) PROD : = S_8
10) S_9 : = I + 1	11) I : = S_9	12) IF I <= 20 Goto (I)

b) Write Function GETREG and describe its use. 8.

8 Write short notes on:
i) Code generation algorithm
ii) Induction Variables.
iii) Basic blocks.
iv) Error recovery in LL Parsing.
v) Global register allocation.

Compiler Design Sept/Oct 1999

Part - A

1 a) Bring out the functions of different phases of a compiler with a diagram. 8.

b) Give (i) the parse tree and (ii) the intermdeiate code for the statement: 8.
while A > B and a <= 2 * B - 5 do
A : = A + B

c) Illustruate local optimisation with simple examples. 4.

2 a) Explain the essential requirements of operator precdence grammar with an example. Also, give operator precedence parsing algorithm. 12.

b) Construct DFA using the subset algorithm for the regular expression (a/b) * a b b. 8.

3 a) Briefly explain the working of LL(1) parser. 6.

b) For the following grammar obtain the canonical collection of sets of LR (0) items. 8.
E' -> E
E -> E + T | T

T -> T * F | F
F -> (E) / id
c) Obtain the SLR table for the grammar of (b) above. 6.

4 a) Construct the syntax directed translation scheme that translates arithmetic expression from postfix notation into infix notation. Give annotated parse trees for the input 2 4 - 6 * and 2 4 6 * -. 10.
b) Explain with an example the simple translation scheme for an assignment statement. 10.

5 a) Explain the meaning of quadruples, triples and indirect pipes with an example. 6.
b) What is activated record? Explain the purpose of each item in an activtion record with an example. 6.
c) Write a note on the implementation of a simple stack allocation scheme. 8.

Part - B

6 a) Explain with an example minimum distance correction of syntactic errors. 6.
b) Explain error recovery in operator precedence parser. 8.
c) Explain the types of errors likely the occur in different phases of a compiler. 6.
7 a) Discuss the code-generation algorithm with a sequence of quadruples constituting a basic block as input. 6.
b) Generate the machine code sequence for the following 3 address - code sequence: 6.
T : = A - B
U : = A - C
V : = T + U
W : = V + U
c) Explain the DAG representation of basic blocks. Assume a sequence of 3-address code for illustruation. 8.
8 Write explanatory notes on the following: 5x4 = 20
i) Flow graphs ii) Regular expression.
iii) Top-down parsing iii) Global register allocation.

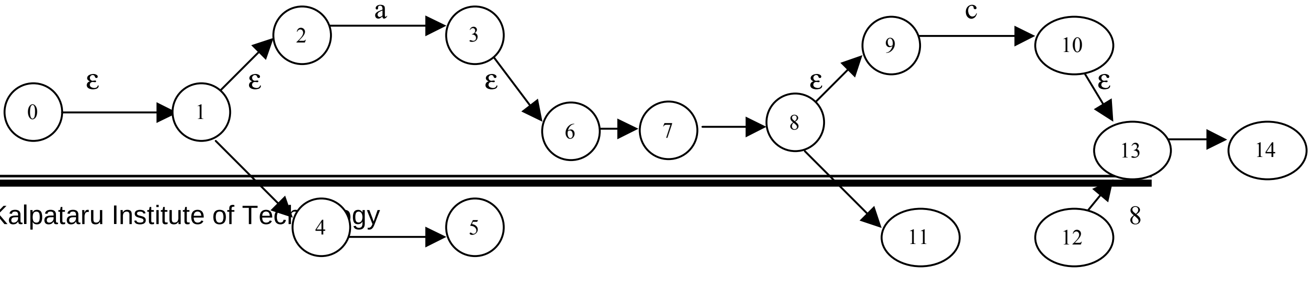
Compiler Design March/April 2000

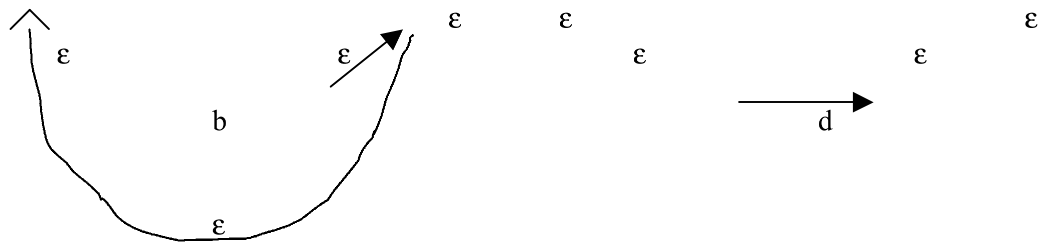
Part - A

1 a) Explain the role of Task management in the design of compiler. 10.
b) Explain the kind of association, which exists between lexical and syntax analysis phases of compiler. 5
c) Explain the different compiler working tools available. 5.

2 a) How input buffering is implemented in lexical analysis? 5
b) Write rules, which are used, for construction of regular expressions. 5
c) Write algorithm for constructing a DFA from non deterministic one. 10.

3 a) Design DFA for the following NFA using subset construction algorithm. 12.





b) Briefly explain the components of context-free grammar. 8.

4 a) Is the following grammar ambiguous? If yes why? 4.

Stat → if cond then stat |
 if cond then stat else stat |
 other - stat

b) Consider the following grammar: 10.

1) E → E + E
2) E → E * E
3) E → (E)
4) E → (id)

Trace shift reduce parsing actions using stack during parsing the statement id1 * id2 + id3.

c) What is operator grammar? Convert the following grammar into operator grammar. 6

E → EAE | (E) | - E | id
A → + | - | * | / | ↑

5 a) Write algorithm for computing operator - precedence relations, assume LEADING(A) and TRAILING(A) for non terminal A are already available. 10.

b) Write algorithm to eliminate left recursion from a grammar with no cycles or ε - productions

c) Eliminate the left-recursion of the following grammar:

E → E + T | T
T → T * F | F
F → (E) | id

Part - B

6 a) Briefly explain: 8.

i) Semantic actions
ii) Translations on the parse tree by considering suitable e.g.

b) Explain different sources of error. 10.

c) What is error recovery? 2.

7 a) What are the principal sources of code optimization. 10.

b) Generate three-address code for the following program segments. Assume A is an allocated static storage and 2 bytes for each elements of A. 10.

```
begin
        SUM : = 0;
        I : = 1;
        do
                begin
                        SUM : = SUM + A[I];
                        I : = I + 1;
```

```
                    end
            while I ≤ 15
    end;
```

8 Write short notes on any four:

e) DAG
ii) Problems in code generation.
iii) Minimum distance matching.
iv) Quadruples.
v) Global register allocation.