MODULE-I  Ch-Language Processors

1. Explain following.
   (i) Execution Gap (iii) Phases of Compiler
   (ii) Interpreter (iv) Language Migrator

2. List various phases of a language processor. Explain roles of first two phases of it.
   Also explain symbol table.

3. Fill in the Blanks
   (i) A ____________________bridges an execution gap to the machine language of a
      Computer system.
      (1) Detranslator, (2) Preprocessor, (3) Language translator
   (ii) A ____________________bridges the specification gap between two programming
        languages.
      (1) Interpreter (2) Language Migrator (3) Compiler
   (iii) ____________________is designed to hold the value of formal parameters during
        Expansion of macro call.
      (1) Actual Parameter Table (2) Macro Name Table (3) Expansion time variable table
   (iv) Syntax analysis processes the string of tokens built by ________________ to determine the
        statement class.
      (1) Semantic Analysis (2) Lexical Analysis (3) Itself

1. Given a grammar,
   \[ E \rightarrow TA, \]
   \[ A \rightarrow +TA \mid \emptyset, \]
   \[ T \rightarrow VB, \]
   \[ B \rightarrow \ast VB \mid \emptyset, \]
   \[ V \rightarrow id \mid (E) \]
   
   Develop an LL(1) parser table and parse following string using the parsing table.
   \[ id \ast (id + id) \]

2. (i) Convert given regular expression to DFA. The expression is \((a \mid b)\ast abb#\)
    (ii) Write regular expressions of a given language. The language consists of all strings of
         \(a\)'s and \(b\)'s which ends with \(a\) and does not contain \(bb\).

3. What is bottom up parser? Explain operator precedence parser. Let a grammar for a
   language is \(E \rightarrow E + E \mid E \ast E \mid id\). Check validity of following string using stack based
   operator precedence parser. \(id \ast id + id \ast id\)

4. Write unambiguous production rules (grammar) for arithmetic expression containing
   +, -, *, / and \(^\) (exponentiation). Construct parse tree and abstract syntax tree for:
   \(<id> - <id> \ast <id> ^ <id> + <id>\)

5. Construct DFA for following regular expression:
   \(a \ast (b \ast \mid c \ast) (a \mid c) \ast \#\)

6. Describe working of LL(1) parser and parse following string:
   \(|- <id> \ast <id> \ast <id> + <id> - |\)

7. What is operator precedence parsing? Show operator precedence matrix for following
   operators: +, -, *, (, ).
   Parse following string: \(|- <id> + <id> \ast <id> - |\)

8. Explain Left recursion, Left factoring and backtracking in top down parsing.

9. Given the Grammer, evaluate the string \(id - id \ast id\) using shift reduce parser.
   \[ E \rightarrow E - E \]
   \[ E \rightarrow E \ast E \]
10. Given the grammar, perform the top-down parsing for the string \( +*35*45 \)

\[
E \rightarrow \text{id} \\
E = +TE|E \\
T = *VT|V \\
V = 0|1|2|3|......|9 \\
\]

11. Develop Regular expression and DFA for declaring a variable in ‘C’ language.

12. Write complete grammar for an arithmetic expression containing operators ‘+’, ‘-’, ‘*’, ‘$’ using recursive specification and Backus Naur Form (BNF) where ‘$’ is exponentiation operator.

13. (i) Build a DFA for following regular expression.

\[(a \mid b)^*aab#\]

(ii) A language consists of all strings of a’s and b’s which ends with b and does not contain aa. Write regular expression for the language.

14. Parse following strings using given LL(1) parsing table (TABLE-I)

(i) id*id + id * id (ii) id + id + id + id

<table>
<thead>
<tr>
<th>Non-terminal</th>
<th>Source symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>&lt;id&gt;</td>
</tr>
<tr>
<td>E'</td>
<td>+</td>
</tr>
<tr>
<td>E'</td>
<td>*</td>
</tr>
<tr>
<td>E'</td>
<td>$</td>
</tr>
<tr>
<td>T</td>
<td>E' \rightarrow TE'</td>
</tr>
<tr>
<td>T'</td>
<td>E' \rightarrow +TE'</td>
</tr>
<tr>
<td>T'</td>
<td>E' \rightarrow \epsilon</td>
</tr>
<tr>
<td>T'</td>
<td>T' \rightarrow \epsilon</td>
</tr>
<tr>
<td>T'</td>
<td>T' \rightarrow VT'</td>
</tr>
<tr>
<td>T'</td>
<td>T' \rightarrow \epsilon</td>
</tr>
<tr>
<td>V</td>
<td>V' \rightarrow &lt;id&gt;</td>
</tr>
</tbody>
</table>

15. Compare top-down and bottom-up parser.

16. Construct DFA for following regular expression:

\[(a* \mid b*)a*ab#\]

17. (i) Consider following grammar

\[S \rightarrow aSbS \mid bSaS \mid \text{epsilon}\]

Derive the string abab. Draw corresponding parse tree. Are these rules ambiguous? Justify.

(ii) Write regular definitions for producing real numbers of programming language ‘C’.
18. Write unambiguous production rules to produce arithmetic expression consisting of +, -, *, /, ^ (exponent), id. Use them for parsing id ^ id ^ id * id + id / id using shift -reduce parser (Naive bottom up parsing). Also lists limitation(s) of the method.

19. (i) Remove left recursion from following production rules:
   A -> AaB | x
   B -> BCb | y
   C -> Cc | epsilon

   (ii) Explain left factoring by giving example.

20. Given a grammar
   S -> XS | dS | ε
   X -> Y | Zb | aY
   Y -> cZ
   Z -> e

   Develop an LL(1) parsing table and check whether the string “dace” is accepted or not?

21. Write a regular expression for the language consisting of all strings ending with 1 and does not contain substring 00. Convert the resultant regular expression into Deterministic Finite Automata.

22. Perform lexical, syntax and semantic analysis on below C statement
   
   a = b + c * d * 100 + e / f

   Where data type of b, c & e are integers and remaining all variables are float.

23. Define Simple Phrase and Handle. Using Handle and Simple Phrase trace the bottom up parsing algorithm.

   Grammar is :
   E -> T+ E | T - E | T
   T -> T * V | T / V | V
   V -> a | b | c | d

   String is : a - b * c + d
24. When Left-factoring on a grammar is applied? Apply left-factoring on the below given grammar and perform Predictive Parsing.

Grammar is

\[ S \rightarrow i \ E \ t \ S \ | \ i \ E \ t \ S \ e \ S \ | \ a \]

\[ E \rightarrow b \]

String is: \( i \ b \ t \ a \ e \ i \ b \ t \ a \)
1. Explain assembly scheme with suitable example

2. Let us consider a two pass assembler and assume that each instruction is one word.

   Given an assembly program and code for Mnemonics.

<table>
<thead>
<tr>
<th>START 101</th>
<th>Mnemonics</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ A</td>
<td>STOP</td>
<td>00</td>
</tr>
<tr>
<td>READ B</td>
<td>MULT</td>
<td>03</td>
</tr>
<tr>
<td>MOVER BREG, A</td>
<td>MOVER</td>
<td>04</td>
</tr>
<tr>
<td>MULT</td>
<td>MOVEM</td>
<td>05</td>
</tr>
<tr>
<td>BREG, B</td>
<td>READ</td>
<td>09</td>
</tr>
<tr>
<td>MOVEM</td>
<td>DS</td>
<td>02</td>
</tr>
<tr>
<td>DREG, D</td>
<td>START</td>
<td>01</td>
</tr>
<tr>
<td>STOP</td>
<td>END</td>
<td>02</td>
</tr>
</tbody>
</table>

   (i) Show content of symbol table at the end of pass-one of an assembler.

   (ii) Write intermediate code representation of the assembly program. Use variant-II of intermediate code representation

3. Explain the role of Mnemonic Opcode Table, Symbol Table, Literal Table and POOL Table in assembling process of assembly language program.

4. Compare single pass assembler and two pass assembler. Explain two pass assembler in detail with suitable example.

5. Explain analysis and synthesis phases of an assembler by clearly stating their tasks.

6. Explain and show usage by giving examples of following assembler directives:
   ORIGIN, EQU, LTORG, START.

7. Explain & compare various intermediate code forms (representations) for an assembler.

8. Explain two pass assembler.

9. Explain the data structure of single pass assembler.

10. Write difference between one pass and two pass assembler.

11. What are advanced assembler directives. Explain any two with suitable example.

12. Consider following assembly program. Show (i) Contents of Symbol Table (ii) intermediate codes using Variant I representation (iii) corresponding machine codes

   START 100
   READ A
READ B
READ C
MOVER AREG, A
ADD AREG, B
ADD AREG, C
MULT AREG, C
MOVEM AREG, RESULT
PRINT RESULT
STOP
A DS 1
B DS 1
C DS 1
RESULT DS 1
END

Instruction opcodes:

READ – 09, MOVER – 04, MOVEM – 05, ADD – 01, MULT – 03, PRINT – 10, STOP – 00

Assembler-directive codes: START – 01, END - 02

Register code: AREG – 01
MODULE-IV
Ch- Macro Processors

1. (i) Write a macro that moves n numbers from the first operand to the second operand, where n is specified as third operand of the macro.
(ii) Write a macro which takes B, C, and D as parameters and calculates B*C + C*D.

2. Compare the features of subroutine and macros with respect to following.
   (i) Execution speed
   (ii) Processing requirement by assembler.
   (iii) Flexibility and generality

3. Explain advanced macro facilities with suitable example.

4. Explain design specification tasks for macro preprocessor with suitable example.

5. Explain following terms with suitable example.
   (i) Expansion time variable
   (ii) Positional parameter
   (iii) Semantic Expansion
   (iv) Macro Preprocessor

6. Do as directed
   (i) A macro is a unit of specification for ________ through expansion. (Fill in the blank)
   (ii) Macro definition is enclosed between a __________ statement and a____________ statement. (Fill in the blank)
   (iii) A ____________________________ is designed to hold the names of all macro defined in a program. (Fill in the blank)
   (iv) An ____________________________ counter is maintained to count the number of nested macro calls. (Fill in the blank)
   (v) A lexical expansion is typically employed to replace occurrence of formal parameters by corresponding actual parameters. (State True/False)
   (vi) Default specification of parameters is useful in situations where a parameter has the different value in most calls. (State True/False)
   (vii) Macro definition table is maintained to hold value of sequencing symbols. (State True/False)

7. Explain with examples - expansion time variables, expansion time statements - AIF
and AGO for macro programming. Show their usage for expansion time loop by giving example.

8. Describe tasks and data structures considered for the design of a macro preprocessor.

9. Define two macros of your choice to illustrate nested calls to these macros. Also show their corresponding expansion.

10. Explain attributes of formal parameters, default specifications of parameter and semantic expansion for macro by giving examples.

11. What is macro in programming language. Write an algorithm for macro definition.

12. Draw a flow chart and explain simple one pass macro processor.

13. Explain macro expansion in details.

14. (i) Define a macro taking A and B as parameters to compute $A = A * B + B * B + A + B$

(ii) Explain positional parameters, keyword parameters and default value parameters for macros.

15. (i) Define a macro taking starting_location and N as parameters to find summation of all N numbers stored at locations starting from starting_location. The result is to be stored at starting_location.

(ii) Illustrate expansion of nested macro calls by giving example.

16. What is Macro-Preprocessor? Explain steps of macro-preprocessor design.
1. Describe in detail how relocation and linking is performed.

2. (i) Write a brief note on MS-DOS linker.
   (ii) What is Overlay? Explain the execution of an overlay structured program.

3. What is program relocation? Explain characteristics of self relocating programs.

4. Explain Absolute Loader with example.

5. Explain BSS loader with example.


7. Explain linking and loading in MS DOS.

8. Explain design of a linker by addressing issues of relocation and linking.
1. Explain memory allocation in block structured language.

2. Given following expression, \( x = -a \times b + a \times b \)
   
   (i) Write three address code for the expression.
   
   (ii) Optimize the three address code if it is possible to do so.
   
   (iii) Give triple implementation for the three address code of the expression.

3. Compare one pass and two pass compilers. Explain various parameter passing mechanisms for functions.

4. Describe various optimizing transformations commonly used in compilers.

5. Explain various steps of code optimization with example.

6. Explain triple and quadruple representation with example.

7. Discuss parameters for Activation Records.

8. What are the issues in code generation in relation to compilation of expression? Explain each issue in brief.

9. Write Three address codes and triple representation for
   
   \( x = x \times y \times z + x \times y + y \times z \)

10. Prepare Symbol Table & Quadruple Table using Value Numbers method

<table>
<thead>
<tr>
<th>Stmt No</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( A = 29.3 \times D )</td>
</tr>
<tr>
<td>17</td>
<td>( B = 24.5 )</td>
</tr>
<tr>
<td>31</td>
<td>( C = A \times B + W )</td>
</tr>
<tr>
<td>49</td>
<td>( X = A \times B + Y )</td>
</tr>
</tbody>
</table>

Subject Co-ordinator: Prof. Jalpa Shah

Head of the Department: Prof. Jayshree Upadhyay

Copy to:
(1) Student Notice Board
(2) Library Photocopy File
(3) Subject Coordinator File
(4) HoD- Subject File