UNIT I
INTRODUCTION TO COMPILERS

1. **State the two main parts of compilation and its function. (Apr/May-18),(Apr/May-17), (May/June-16)**

- Analysis part
    - breaks up the source program into constituent pieces and creates an intermediate representation of the source program.
- Synthesis part
    - constructs the desired target program from the intermediate representation

2. What is a symbol table? (Nov/Dec-16)

- A Symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier.
- The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly.

3. List the various compiler construction tools. (Nov/Dec-16)

• Scanner generators	[Lexical Analysis]
• Parser generators	[Syntax Analysis]
• Syntax-directed translation engines	[Intermediate Code]
• Data-flow analysis engines	[Code Optimization]
• Code-generator generators	[Code Generation]
• Compiler-construction toolkits	[For all phases]

4. What is a Complier?

- A Complier is a program that reads a program written in one language-the source language-and translates it in to an equivalent program in another language-the target language.
- The compiler reports to its user the presence of errors in the source program

5. State the general phases of a compiler

- Lexical analysis
- Syntax analysis
- Semantic analysis
- Intermediate code generation
- Code optimization
- Code generation

6. What happens in linear analysis?

- This is the phase in which the stream of characters making up the source program is read from left to right and grouped in to tokens that are sequences of characters having collective meaning.

7. What happens in Semantic analysis?

- This is the phase in which certain checks are performed to ensure that the components of a program fit together meaningfully.

8. List the phases that constitute the front end of a compiler.

- The front end consists of those phases or parts of phases that depend primarily on the source language and are largely independent of the target machine. These include
    - Lexical and Syntactic analysis
    - The creation of symbol table

        - Semantic analysis
        - Generation of intermediate code
- A certain amount of code optimization can be done by the front end as well. Also includes error handling that goes along with each of these phases.

9. Mention the back-end phases of a compiler.

- The back end of compiler includes those portions that depend on the target machine and generally those portions do not depend on the source language, just the intermediate language.
- These include
    - Code optimization
    - Code generation, along with error handling and symbol- table operations.

10. Describe the possible error recovery actions in lexical analyzer. (Apr/May-18)

- Deleting an extraneous character
- Inserting a missing character
- Replacing an incorrect character by a correct character
- Transposing two adjacent characters

11. Differentiate token, pattern and lexeme. Give examples (Nov/Dec-16)

- token- Sequence of characters that have a collective meaning.
- pattern- There is a set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token
- lexeme- A sequence of characters in the source program that is matched by the pattern for a token.

12. What are the various parts in LEX program? (Apr/May-17)

A **Lex** program has the following form:

```
Declarations

%%

Transition rules

%%
Auxiliary functions
```

13. List the operations on languages. (May/June-16)

- Union - L U M ={s | s is in L or s is in M}
- Concatenation – LM ={st | s is in L and t is in M}
- Kleene Closure – L* (zero or more concatenations of L)
- Positive Closure – L+ (one or more concatenations of L)

14. Apply the rules used to define a regular expression. Give example. (Apr/May-18)

Regular expression is a method to describe regular language

Rules:

- ε-is a regular expression that denotes {ε} that is the set containing the empty string
- If a is a symbol in ∑,then a is a regular expression that denotes {a}
- Suppose r and s are regular expressions denoting the languages L(r) and L(s) Then,
    - (r)/(s) is a regular expression denoting L(r) U L(s).
    - (r)(s) is a regular expression denoting L(r)L(s)

- ○ (r)* is a regular expression denoting L(r)*.
- ○ (r) is a regular expression denoting L(r).

15. What is Lexical Analysis?

- The first phase of compiler is Lexical Analysis.
- This is also known as linear analysis in which the stream of characters making up the source program is read from left-to-right and grouped into tokens that are sequences of characters having a collective meaning.

16. What is a lexeme? Define a regular set. Nov/Dec 2006

- A Lexeme is a sequence of characters in the source program that is matched by the pattern for a token.
- A language denoted by a regular expression is said to be a regular set

17. What is a sentinel? What is its usage? April/May 2004

A Sentinel is a special character that cannot be part of the source program. Normally we use 'eof' as the sentinel. This is used for speeding-up the lexical analyzer.

18. Construct Regular expression for the language

L= {w ε{a,b}/w ends in abb}

Ans: {a/b}*abb.

19. What is recognizer?

Recognizers are machines. These are the machines which accept the strings belonging to certain language. If the valid strings of such language are accepted by the machine then it is said that the corresponding language is accepted by that machine, otherwise it is rejected.

20.Mention the various notational shorthands for representing regular expressions.

- One or more instances (+)
- Zero or one instance (?)
- Character classes ([abc] where a,b,c are alphabet symbols denotes the regular expressions a | b | c.)
- Non regular sets

UNIT II
SYNTAX ANALYSIS

1. What do you mean by handle pruning? (Apr/May-18), (Nov/Dec-16)

A rightmost derivation in reverse can be obtained by handle pruning.

If w is a sentence of the grammar at hand, then w = γn, where γn is the nth right-sentential form of some as yet unknown rightmost derivation

S = γ0 => γ1…=> γn-1 => γn = w

2. Define parser.
- Hierarchical analysis is one in which thetokens are grouped hierarchically into nested collections with collective meaning.
- Also termed as Parsing.

3. Mention the basic issues in parsing.
- Specification of syntax
- Representation of input after parsing.

4. Define a context free grammar.

A context free grammar G is a collection of the following
- V is a set of non terminals
- T is a set of terminals
- S is a start symbol
- P is a set of production rules
- G can be represented as G = (V,T,S,P) Production rules are given in the following form

Non terminal → (V U T)*

5. Briefly explain the concept of derivation.
- Derivation from S means generation of string w from S.
- For constructing derivation two things are important.
  - Choice of non terminal from several others.
  - Choice of rule from production rules for corresponding non terminal.
  - Instead of choosing the arbitrary non terminal one can choose either
    1. leftmost derivation – leftmost non terminal in a sentinel form
    2. or rightmost derivation – rightmost non terminal in a sentinel form

6. Define ambiguous grammar.
- A grammar G is said to be ambiguous if it generates more than one parse tree for some sentence of language L(G).
- i.e. both leftmost and rightmost derivations are same for the given sentence.

7. List the properties of LR parser.
- LR parsers can be constructed to recognize most of the programming languages for which the context free grammar can be written.
- The class of grammar that can be parsed by LR parser is a superset of class of grammars that can be parsed using predictive parsers.
- LR parsers work using non backtracking shift reduce technique yet it is efficient one.

8. Mention the types of LR parser.
- SLR parser - Simple LR parser
- LALR parser - Lookahead LR parser
- CLR Parser -Canonical LR parser

9. What are the problems with top down parsing?

- The following are the problems associated with top down parsing:
  - Backtracking
  - Left recursion
  - Left factoring
  - Ambiguity

10. Write the algorithm for FIRST and FOLLOW.

FIRST:

- If X is terminal, then FIRST(X) IS {X}.
- If X → ε is a production, then add ε to FIRST(X).
- If X is non terminal and X → Y1,Y2..Yk is a production, then place a in FIRST(X) if for some i , a is in FIRST(Yi) , and ε is in all of FIRST(Y1),…FIRST(Yi-1);

FOLLOW:

- Place $ in FOLLOW(S),where S is the start symbol and $ is the input right endmarker.
- If there is a production A → αBβ, then everything in FIRST(β) except for ε is placed in FOLLOW(B).
- If there is a production A → αB, or a production A→ αBβ where FIRST(β) contains ε , then everything in FOLLOW(A) is in FOLLOW(B).

11. What is dangling else problem?

Ambiguity can be eliminated by means of dangling-else grammar which is show below:
stmt → if expr then stmt
| if expr then stmt else stmt | other

12. Write short notes on YACC. (April/May 2019)

- YACC is an automatic tool for generating the parser program.
- YACC stands for Yet Another Compiler Compiler which is basically the utility available from UNIX.
- Basically YACC is LALR parser generator.
- It can report conflict or ambiguities in the form of error messages.

13. Define LR(0) items.

- An LR(0) item of a grammar G is a production of G with a dot at some position of the right side.
- Thus, production A → XYZ yields the four items

  A→.XYZ
  A→X.YZ
  A→XY.Z
  A→XYZ.

14. What is meant by viable prefixes?

The set of prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes. An equivalent definition of a viable prefix is that it is a prefix of a right sentential form that does not continue past the right end of the rightmost handle of that sentential form.

15. Define handle.

- A handle of a string is a substring that matches the right side of a production, and whose reduction to the nonterminal on the left side of the production represents one step along the reverse of a rightmost derivation.

- A handle of a right – sentential form γ is a production $A \rightarrow \beta$ and a position of γ where the string β may be found and replaced by A to produce the previous right-sentential form in a rightmost derivation of γ.
    - That is , if $S \Rightarrow \alpha Aw \Rightarrow \alpha\beta w$,then $A \rightarrow \beta$ in the position following α is a handle of $\alpha\beta w$.

16. What are kernel & non-kernel items?

- Kernel items, which include the initial item, $S' \rightarrow .S$, and all items whose dots are not at the left end.
- Non-kernel items, which have their dots at the left end.

17. What is phrase level error recovery?

- Phrase level error recovery is implemented by filling in the blank entries in the predictive parsing table with pointers to error routines.
- These routines may change, insert, or delete symbols on the input and issue appropriate error messages. They may also pop from the stack.

18. Define bottom up parsing?

- It attempts to construct a parse tree for an input string is beginning at leaves and working up towards the root (i.e.) reducing a string „w" to the start symbol of a grammar.
- At each reduction step, a particular substring matching the right side of a production is replaced by the symbol on the left of that production.
- It is a rightmost derivation and it" s also known as shifts reduce parsing.

19. What are the difficulties with top down parsing?

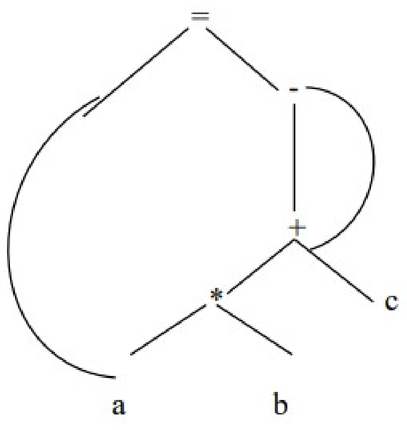
- Left recursion
- Backtracking
- The order in which alternates are tried can affect the language accepted
- When failure is reported. We have very little idea where the error actually occurred.

20. Define top down parsing?

- It can be viewed as an attempt to find the left most derivation for an input string.
- It can be viewed as attempting to construct a parse tree for the input starting from the root and creating the nodes of the parse tree in preorder.

UNIT III -2 MARKS

1. Draw the DAG for the statement a = (a*b+c)–(a*b+c). NOV/DEC 2017



2. Define DAG. MAY/JUNE 2016 , NOV/DEC 2007, MAY/JUNE 2007
A DAG for a basic block is a directed acyclic graph with the following labels on nodes:
i) Leaves are labeled by unique identifiers, either variable names or constants.
ii) Interior nodes are labeled by an operator symbol.
iii)Nodes are also optionally given a sequence of identifiers for labels.

3. When does dangling references occur MAY/JUNE 2016
When there is a reference to storage that has been de-allocated, logical error occurs as it uses dangling reference where the value of de-allocated storage is undefined according to the semantics of most languages.

4. Mention the two rules for type checking. NOV/DEC 2011, APRIL/MAY 2017
Type checker for a language is based on information about the syntactic constructs in the language, the notion of types, and the rules for assigning types to language constructs.

5. Write down syntax directed definition of a simple desk calculator. NOV/DEC 2016

Production	Semantic rule
$L \rightarrow En$	Print(E.val)
$E \rightarrow E_1 + T$	$E.val := E_1.val + T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 * F$	$T.val := T_1.val * F.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow (E)$	$F.val := E.val$
$F \rightarrow$ **digit**	F.val := **digit**.lexval

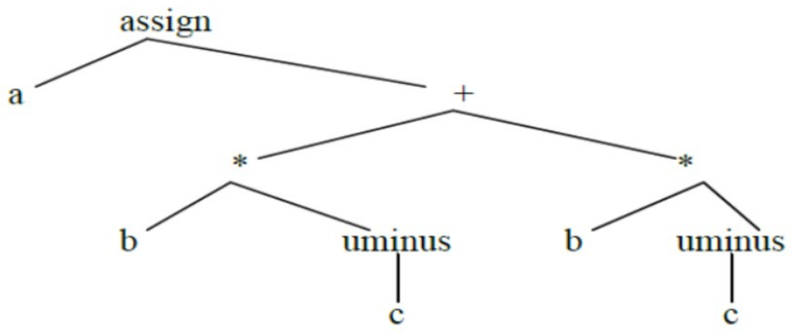
6. What is synthesized attributes?
A synthesized attribute at node N is defined only in terms of attribute values of children of N and at N

7. What is inherited attributes ?
An inherited attribute at node N is defined only in terms of attribute values at N’s parent, N itself and N’s siblings

8. What is a syntax tree? Draw the syntax tree for the assignment statement a := b * -c + b * -c. APRIL/MAY 2011, NOV/DEC 2011 NOV/DEC 2012

A syntax tree depicts the natural hierarchical structure of a source program.

Syntax tree:



9. Mention the rules for type checking. (Apr/May-17)

Rule for type synthesis has the form:

if f has type $s \to t$ **and** x has type s,
then expression $f(x)$ has type t

Rule for type inference has the form:

if $f(x)$ is an expression,
then for some α and β, f has type $\alpha \to \beta$ **and** x has type α

10. What are the functions used to create the nodes of syntax trees?

- Mknode (op, left, right)
- Mkleaf (id,entry)
- Mkleaf (num, val)

11. Define a syntax-directed translation?

- Syntax-directed translation specifies the translation of a construct in terms of Attributes associated with its syntactic components.
- Syntax-directed translation uses a context free grammar to specify the syntactic structure of the input. It is an input- output mapping.

12. Define an attribute. Give the types of an attribute?

- An attribute may represent any quantity, with each grammar symbol,
- it associates a set of attributes and with each production, a set of semantic rules for computing values of the attributes associated with the symbols appearing in that production.
  - Example: a type, a value, a memory location etc.,
    - Synthesized attributes.
    - Inherited attributes.

13. What methods have been proposed for evaluating semantic rules?

Parse – tree methods
Rule – based methods
Oblivious methods

14. What is a topological sort?

A topological sort of a directed acyclic graph is any ordering m1,m2,….,mk of the nodes of the graph such that edges go from nodes earlier in the ordering to later nodes .

15. What is a type expression?

The type of a language construct will be denoted by a “type expression” . Informally a type expression is either a basic type or is formed by applying an operator called a type constructor to other type expressions.

16. What is a type system?

A type system is a collection of rules for assigning type expressions to the various parts of a program. A type checker implements a type system.

17. What are the advantages of generating an intermediate representation?

i) Ease of conversion from the source program to the intermediate code.
ii) Ease with which subsequent processing can be performed from the intermediate code.

18. Define annotated parse tree?

A parse tree showing the values of attributes at each node is called an annotated parse tree. The process of computing an attribute values at the nodes is called annotating parse tree.

19. Write the 3-addr code for the statements a =b*-c + b*-c?

Three address codes are: a=b*-c + b*-c

```
T1 = -c
T2 = b*T1
T3 = -c
T4 = b*T3
T5 = T2+T4
a:= T5.
```

20.Give the 2 attributes of syntax directed translation into 3-addr code?

E.place, the name that will hold the value of E and
E.code , the sequence of 3-addr statements evaluating E.

Name different storage allocation strategies used in run time environment. (Apr/May-18)

The strategies are:

- Static allocation
- Stack allocation
- Heap allocation

2.What is an sctivation record? Give the structure of activation record?

- The activation record is a block of memory used for managing the information needed by a single execution of a procedure.
- Various fields f activation record are:
  - Temporary variables
  - Local variables
  - Saved machine registers
  - Control link
  - Access link
  - Actual parameters
  - Return values

3. What is dynamic scoping?

- In dynamic scoping a use of non-local variable refers to the non-local data declared in most recently called and still active procedure.
- Therefore each time new findings are set up for local names called procedure. In dynamic scoping symbol tables can be required at run time.

4. Define symbol table.

Symbol table is a data structure used by the compiler to keep track of semantics of the variables. It stores information about scope and binding information about names.

5. What are the various ways to pass a parameter in a function?

- Call by value
- Call by reference
- Copy-restore
- Call by name

6. Suggest a suitable approach for computing hash function.

- Using hash function we should obtain exact locations of name in symbol table.
- The hash function should result in uniform distribution of names in symbol table.
- The hash function should be such that there will be minimum number of collisions. Collision is such a situation where hash function results in same location for storing the names.

7. What are the functions for constructing syntax trees for expressions?

- The construction of a syntax tree for an expression is similar to the translation of the expression into postfix form.
- Each node in a syntax tree can be implemented as a record with several fields.

8. Give short note about call-by-name?

- Call by name, at every reference to a formal parameter in a procedure body the name of the corresponding actual parameter is evaluated.
- Access is then made to the effective parameter.

9. How parameters are passed to procedures in call-by-value method?

- This mechanism transmits values of the parameters of call to the called program.
- The transfer is one way only and therefore the only way to returned can be the value of a function.

```
Main ( )
{ print (5);
}
Void print (int n)
{ printf (“%d”, n); }
```

10. Define static allocations and stack allocations

- Static allocation is defined as lays out for all data objects at compile time.
  - Names are bound to storage as a program is compiled, so there is no need for a run time support package.

11. **What are the issues in the design of code generator?**
- Input to the generator
- Target programs
- Instruction selection
- Register allocation
- Evaluation order.

12. Give the variety of forms in target program.
- Absolute machine language.
- Relocatable machine language.
- Assembly language.

13. What is code motion?
- Code motion is an optimization technique in which amount of code in a loop is decreased.
- This transformation is applicable to the expression that yields the same result independent of the number of times the loop is executed.
- Such an expression is placed before the loop.

14. Give the factors of instruction selections.
- Uniformity and completeness of the instruction sets
- Instruction speed and machine idioms
- Size of the instruction sets.

15. Define environment and state.

The term environment refers to a function that maps a name to a storage location.
The term state refers to a function that maps a storage location to the value held there.

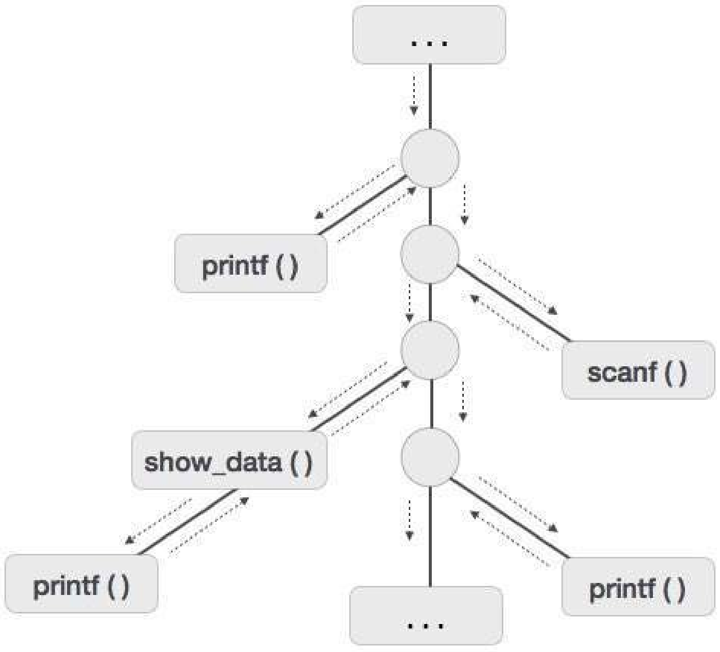
17. Write three address code sequence for the assignment statement d:=(a-b)+(a-c)+(a-c) (May/June-16)

Statement	Code generation	Register descriptor	Address descriptor
t:=a-b	MOV a,R0 SUB b,R0	R0 contains t	t in R0
u:=a-c	MOV a,R1 SUB c,R1	R0 contains t R1 contains u	t in R0 u in R1
v:=t+u	ADD R1,R0	R0 contains v R1 contains u	u in R1 v in R0
d:=v+u	ADD R1,R0 MOV R0,d	R0 contains d	d in R0 d in R0 and memory

16. Draw the activation tree for the following code (Apr/May-18)

```
int main()
{
printf("Enter Your Name");
scanf("%s",username);
int show_data(username);
printf("Press Any Key to Continue...");
...
int show_data(char *user)
{
printf("Your Name is %s",username);
return 0;
}
}
```

Activation Tree



17. List Dynamic Storage Allocation techniques. **(Nov/Dec-16)**

- Explicit Allocation of Fixed-Sized Blocks
- Explicit Allocation of Variable-Sized Blocks
- Implicit Deallocation

UNIT V
2 MARKS

1. What are the properties of optimizing compiler?
(May/June-16)

- The source code should be such that it should produce minimum amount of target code.
- There should not be any unreachable code.
- Dead code should be completely removed from source language.
- The optimizing compilers should apply following code improving transformations on source language.
  - common sub-expression elimination
  - dead code elimination
  - code movement
  - strength reduction

2.What are the characteristics of peephole optimization? (Nov/Dec-16)

- Redundant-instruction elimination
- Flow-of-control optimizations.
- Algebraic simplifications
- Use of machine idioms

3. Define basic block and flow graph.

- A basic block is a sequence of consecutive statements in which flow of Control enters at the beginning and leaves at the end without halt or possibility Of branching except at the end.
- A flow graph is defined as the adding of flow of control information to the Set of basic blocks making up a program by constructing a directed graph.

4. Write the step to partition a sequence of 3 address statements into basic blocks.

- First determine the set of leaders, the first statement of basic blocks.
  - The first statement is a leader.
  - Any statement that is the target of a conditional or unconditional goto is a leader.
  - Any statement that immediately follows a goto or conditional goto statement is a leader.
- For each leader, its basic blocks consists of the leader and all statements Up to but not including the next leader or the end of the program.

5. Give the important classes of local transformations on basic blocks

- Structure preservation transformations
- Algebraic transformations.

6. Describe algebraic transformations.

- It can be used to change the set of expressions computed by a basic blocks into A algebraically equivalent sets.
- The useful ones are those that simplify the Expressions place expensive operations by cheaper ones.

$$X = X + 0$$
$$X = X * 1$$

7. What is meant by register descriptors and address descriptors?

- A register descriptor keeps track of what is currently in each register.
  - It is Consulted whenever a new register is needed.
- An address descriptor keeps track of the location where ever the current Value of the name can be found at run time.
  - The location might be a register, a Stack location, a memory address,

8. What are the actions to perform the code generation algorithms?

- Invoke a function get reg to determine the location L.
- Consult the address descriptor for y to determine y‟, the current location of y.
- If the current values of y and/or z have no next uses, are not live on exit from the block, and are in register, alter the register descriptor.

9. Write the labels on nodes in DAG.

- A DAG for a basic block is a directed acyclic graph with the following Labels on nodes:
  - Leaves are labeled by unique identifiers, either variable names or constants.
  - Interior nodes are labeled by an operator symbol.
  - Nodes are also optionally given a sequence of identifiers for labels.

10. Give the applications of DAG.

- Automatically detect the common sub expressions
- Determine which identifiers have their values used in the block.
- Determine which statements compute values that could be used outside the blocks.

11. Define Peephole optimization.

- A Statement by statement code generation strategy often produces target code that contains redundant instructions and suboptimal constructs.
- "Optimizing" is misleading because there is no guarantee that the resulting code is optimal.
- It is a method for trying to improve the performance of the target program by examining the short sequence of target instructions and replacing this instructions by shorter or faster sequence.

12. What are the structure preserving transformations on basic blocks?

- Common sub-expression elimination
- Dead-code elimination
- Renaming of temporary variables
- Interchange of two independent adjacent statement

13. Define Dead-code elimination with ex.

- It is defined as the process in which the statement x=y+z appear in a basic block, where x is a dead that is never subsequently used.
- Then this statement maybe safely removed without changing the value of basic blocks.

14. Define reduction in strength with ex.

- Reduction in strength replaces expensive operations by equivalent cheaper ones on the target machines.
- Certain machine instructions are cheaper than others and can often be used as special cases of more expensive operators.
- Ex:
  - X^2 is invariably cheaper to implement as x*x than as a call to an exponentiation routine.

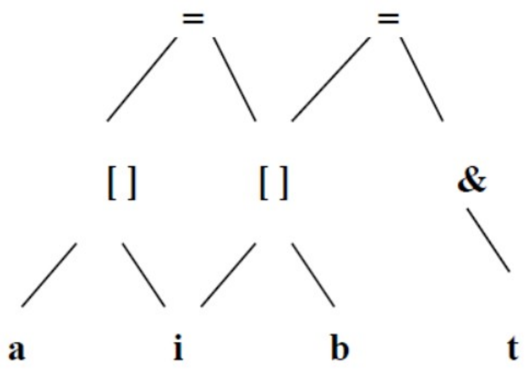
15. Define code optimization and optimizing compiler

- The term code-optimization refers to techniques a compiler can employ in an attempt to produce a better object language program than the most obvious for a given source program.
- Compilers that apply code-improving transformations are called Optimizing-compilers.

16. **Name the techniques in Loop optimization. MAY/JUNE 2014**

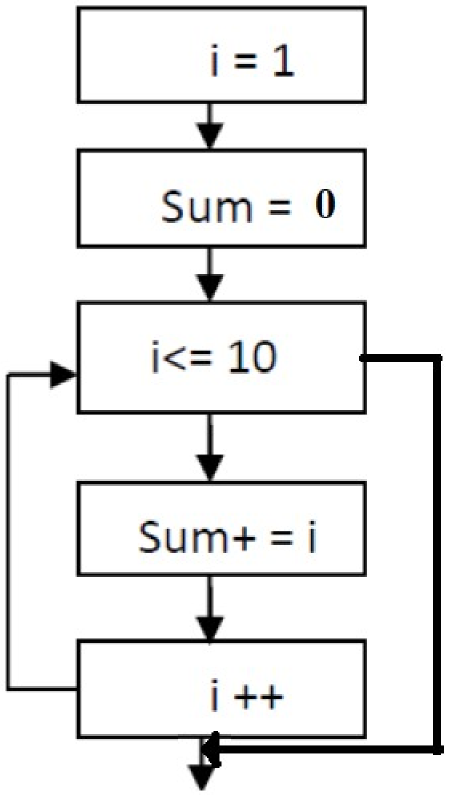
Code Motion, Induction variable elimination, Reduction in strength

17. Draw DAG to represent a[i]=b[i]; a[i]=&t;



18.**Represent the following in flow graph NOV/DEC 2014**

i=1; sum=0;while (i<=10){sum+=i;i++;}



19.How to perform register assignment for outer loops? MAY/JUNE 2012

Outer loop L1 contains an inner loop L2 names allocated registers in L2 need not be allocated registers in L1- L2

20.When is a flow graph reducible? APRIL/MAY 2012 MAY/JUNE 2012

A flow graph is reducible if and only if we can partition the edges into two disjoint groups often called the forward edges and back edges.