Final Revision Theoretical Part (Compiler)

1) What is the difference between assembler and compiler?

Assembler	Compiler
Assembler is a translator for the assembly language of a particular computer which is a symbolic form of one machine language (Low level)	A compiler is a translator for high level language like C, C++

2) Explain the function of the following programs that are related to compilers?

- **Compiler:** Translates a program as a whole.
- **Interpreters:** Translates program one statement at a time.
- **Pre-processors:** Modify or generate code before compilation.
- **Linkers:** Combine object files into an executable or shared library.
- **Loaders:** Load and prepare executable files for execution in memory.
- **Debuggers:** Help find and fix bugs in a program during runtime.
- **Profilers:** Measure and analyze program performance for optimization.

3) What is meant by cross compiler?

- Cross compiler refers to a compiler that runs on one platform but generates code for a different platform.
- It enables developers to write code on one system and compile it for another system with a different architecture or operating system.

4) What is constant folding?

- A compiler optimization technique that simplifies expressions with constant values during compilation, replacing them with their computed results.
- It reduces runtime computations and improves code efficiency.
- Ex: i = 320 * 200 * 32 ➔ i = 2048000

5) Lexical analysis, syntax, and semantic errors are types of errors that a compiler must deal with. Explain the three types and give an example for each.

	Lexical	Syntax	Semantic
Definition	Lexical errors occur when the code contains invalid or unrecognized tokens	هنا ممكن نقول ان error in language rule Errors in the structure or arrangement of code.	Mistakes in the logic or meaning of the code, where the code runs but doesn't give the expected result.
Example	whle True: print("Hello,world!")	int a = 5 (semicolon is missing)	int a = "hello"; Memory overflow

Semantic Errors :

Static semantic errors: Which can be detected by the compiler

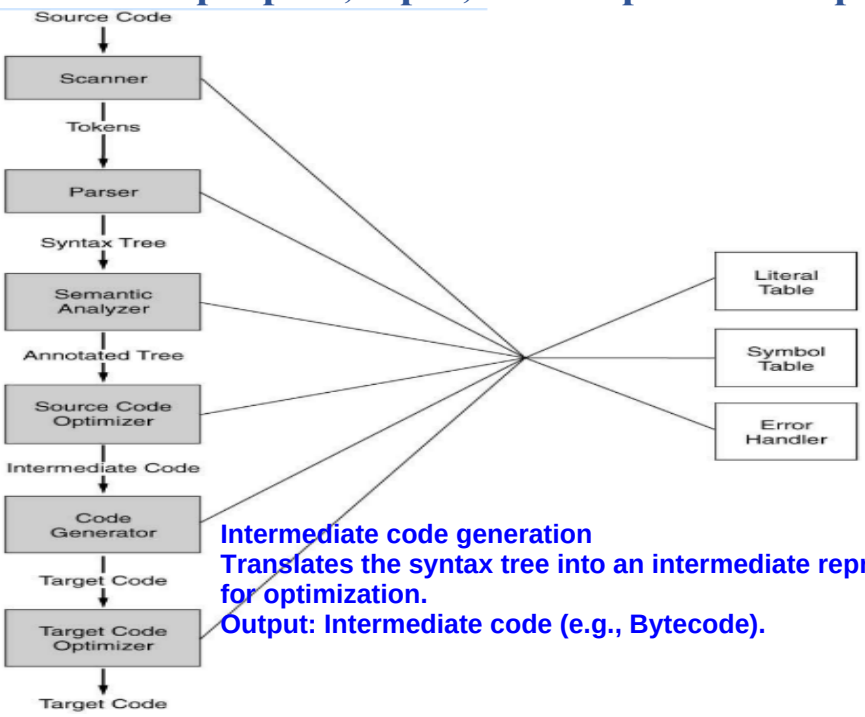
Run time errors: Runtime errors cause the program to crash or abort in some way

Logic errors:Cause a program to run to completion ,but produce the incorrect output or result

Those errors are not detected by the compiler

Omnia Khaled

6) Sketch the graph of the phases of the compiler translation process. Briefly describe the purpose, input, and output of each phase.



Intermediate code generation
Translates the syntax tree into an intermediate representation (IR) that is easier for optimization.
Output: Intermediate code (e.g., Bytecode).

- **Scanner (Lexical analysis):** it collects sequences of characters into meaningful units called tokens.
- **Parser (Syntax analysis):**it determines the structure of the program. The results of syntax analysis are a parse tree or a syntax tree.
- **Semantic Analyzer:** Check the meaning. The output is an annotated parse tree.
- **Source code optimizer:** The code improvement depends only on the source code, and as a separate phase.
- **Code generator:** It takes the intermediate code or IR and generates code for the target machine.
- **Target Code Optimizer:** It improves the target code generated by the code generator.

7) At what phases are regular expressions used in compilers?

- The scanner (Lexical Analysis)

8) Which phase (or phases) of a compiler...

a) check(s) Assignment statements must end with a semicolon (;) **parser**
b) check(s) for type mismatches? **semantic analyzer.**
c) generate(s) a syntax tree? **parser**
d) check(s) that A function is called with the correct number of arguments? **semantic analyzer.**
e) perform(s) constant folding? **source code optimizer.**
f) is/are independent of the underlying machine? **scanner, parser, semantic analyzer, source code optimizer**
g) directly read(s) the input characters? **Scanner**

9) Define the token.

- The token is the smallest meaningful unit of a language.

10) Explain finite state automata in lexical analysis phase?

- **Finite state automata**
  - A language or technique for writing programs that recognize tokens that match the rules.

11) What is meant by regular expressions? What are the three basic operations in regular expressions?

- **Regular expressions:** a language or technique for writing rules that describe tokens and represent patterns of strings of characters.
- **Regular Expression Operations**
  - Choice among alternatives (|)
  - Concatenation and Repetition or "closure" (*)

12) Definition of a DFA

- A DFA (Deterministic Finite Automation) consist of:
  - an alphabet $\sum$, A set of states S , a transition function T: $S \times \sum \rightarrow S$
  - a start state , a set of accepting states

13) Difference between DFA, NFA

DFA	NFA
Deterministic finite automate	Non-deterministic finite automate
DFA cannot use Empty String transition.	NFA can use Empty String transition.
For a given state, on a given input we reach a deterministic and unique state.	For a given state, on a given input we reach more than one state.
All DFA are NFA.	Not all NFA are DFA.
DFA can be understood as one machine	NFA can be understood as multiple little machines computing at the same time.

14) Define abstract syntax tree.

- The parse tree contains more information than is absolutely necessary for a compiler

An abstract syntax tree (AST) is a simplified, abstracted version of the parse tree that captures only the essential structure of the source code suited for compiler processing.

15) What is ambiguous grammar? What are the basic methods to remove ambiguity?

- **Ambiguous grammar:** A grammar that generates a string with two distinct parse trees.
- **Method to remove ambiguity:**
  - One is to state a rule that specifies in each ambiguous case which of the parse trees (or syntax trees) is the correct one, called a disambiguating rule.
  - The alternative is to Change the grammar into a form that forces the construction of the correct parse tree, thus removing the ambiguity

16) Define annotated parse tree.

- A parse tree showing the values of the attributes at each node.

17) Explain how to describe semantic using syntax directed semantic definitions.

- **A syntax directed definition** is an extension of context free grammar in which each grammar symbol is assigned certain attributes.
- **Approach**:
  - Each grammar symbol is associated with a set of attributes.
  - Each production is associated with a set of semantic rules.

18) Explain Attribute grammar.

- Attributes have values that hold semantic information about the (non)terminal.

19) Briefly describe the purpose of dependency graphs.

- Dependency graphs are useful tools for determining an evaluation order for the attribute instances in each parse tree.

20) Why Top-Down parsers cannot handle Left Recursive Grammars.

- Because it is said that it would lead to infinite recursively
  - Left-recursive grammar can cause an infinite loop in parsing as productions are expanded without consuming input.
  - In this case, guessing right doesn't help because the tree keeps expanding without advancing the input, leading to an endless parsing process.

21) What is the purpose of EBNF, in contrast to BNF?

of grammer compared to standard BNF

- EBNF improves the readability and conciseness of BNF through extensions:
  - Kleene star -- a sequence of zero or more elements of the class marked.
  - Square brackets may be used to indicate optional elements.

22) Closure Properties of (Context Free Grammar)

Language

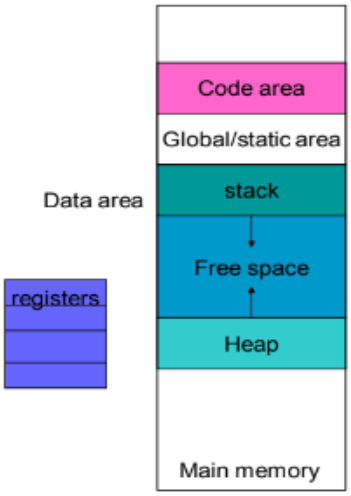
- CFL's are closed under union, concatenation, and Kleene closure.
- Also, under reversal, homomorphisms and inverse homomorphisms.
- But not under intersection or difference.

23) Describe what must be done on a return from a function call.

- Sequence of operations performed when return from procedure calls.
  - Find the arguments and pass them back to the caller.
  - Free the caller environment
  - Restore the caller environment including PC.

24) Draw a diagram illustrating the Memory Organization during program execution.

- **Main memory**
  - **Store large amount of data**
  - **Slower access**
- **Registers**
  - **Very small amount of data**
  - **Faster access**



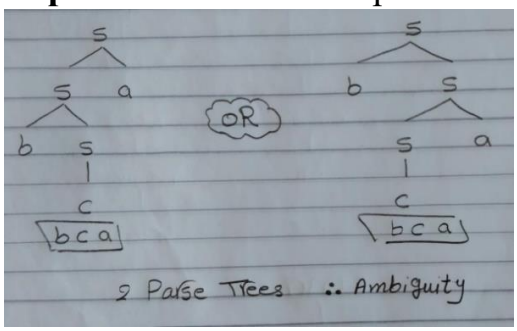
25) Discuss the purpose of procedure activation record and its general organization?

- **Activation record** is used to manage the information needed by a single execution of a procedure.
- **Activation record** is pushed into the stack when a procedure is called and it is popped when the control returns to the caller function
- **The contents of activation records**

Space for arguments (parameters)
Space for bookkeeping information, including return address
Space for local data
Space for local temporaries

True or False

1) **The following grammar is ambiguous ?**
   S → Sa
   S→bS
   S→c
   True
   Explain: more than one pares tree



2) **The regular expressions a+a and a*a describe the same set of strings**
   False
   Explain: a+a = {aa,aaa,aaaa,….} **but** a*a= {a,aa,aaa,aaaa,….}
3) **A parser transforms a stream of characters into a stream of tokens**
   False
   Explain: Scanner transforms a stream of characters into a stream of tokens
4) **The regular expressions a+a and a*aa describe the same set of strings.**
   True
   Explain: a+a; set of strings {aa,aaa,aaaa,.....} **but** a*aa; set of strings {aa,aaa,aaaa,...}
5) **S => b | A**
   A => Aa | a
   This grammar is ambiguous.
   False
   Explain: only one parse tree