Victoria University of Bangladesh
Dept. of Computer Science & Engineering
Program:- B.Sc in CSET
Semester:- Fall 2023
Course title:- Compiler
Course code:- CSI 411

Name:- Husrat Jahan Tanshee
ID:- 2520200011
Batch:- 20

Ans to the Q no-(1)

a) <u>Types of Compiler:-</u>

A compiler is a type of software that converts the source code to the object code. In other words, we can say that it converts the high level language to machine/binary language.

Types of compiler:-

(1) <u>Cross ~~co~~ compiler:-</u> They produce an executable machine code for a platform but this platform is not the one on which the compiler is running.

(2) <u>Bootstrap compiler:-</u> These compilers are written in a programming language that they have to compile.

(3) <u>Source to source/transcompile:-</u> These compilers converts the source code of one programming language to the source code of another programming language.

(4) <u>Decompiler:-</u> Basically, it is not a compiler. It is just the reverse of the compiler. It converts the machine code into high-level language.

Phases of Compiler:- The phases of the compilation process are is follows:

① Lexical Analyzer:- ① It takes the high-level language source code as the input.

① It scans the characters of source code from left to right.

① It groups the characters into lexems.

① Each lexeme corresponds to form a token.

① It removes white spaces and comments

① It checks and removes the lexical errors.

② Syntax Analyzer:- ① A syntax analyzer, also known as parser, checks for syntax errors in source code by constructing a parse tree of tokens, adhering to source code grammar rules.

① The graman for such codes is content free grammar.

③ Semantic Analyzer:- ① The syntax analyzer verifies the parse tree, checks code validity and produces an annotated parse tree.

① It performs flow and type checking.

④ Intermediate Code Generator:- ① The process generates an intermediate code which is neither high level nor machine language and is converted to machine language, with the final two phases being platform-dependent, and an example is a three address code.

⑤ Code Optimizer:- (i) The code optimizer converts intermediate code for faster execution using fewer resources, removes unnecessary lines and rearranges it while maintaining the source codes meaning.

⑥ Target Code Generator:- The compiler converts the optimized intermediate code into machine code, marking the final stage of compilation, ensuring that the resulting machine code is relocatable.

b) Advantages of Compiler:-

Compilers offer several benefits, including improved performance, portability, increased security, debugging support and no dependencies. Compiled code runs faster than interpreted code, making it ideal for performance-critical applications. Compilers allow programmers to write high-level programming languages that can be easily translated into machine code for various platforms. They also perform security checks.

such as checking for security errors and enforcing type safety, preventing vulnerabilities. Debugging tools are included in most compilers.

Disadvantage of Compilers:-

Compiling source code can be time-consuming and can hinder productivity due to frequent updates. Compilers can only detect syntax and semantic errors, which may not catch all errors. Compilers can limit program flexibility and consume system resources, potentially affecting other tasks.

(c) Why to learn Compiler design:-

(i) Compilers provide you with the theoretical and practical knowledge that is needed to implement a programming language. Once learned a compiler someone pretty much knows the innards of many programming languages. Judging programming language by its essential will become easy.

(ii) Compilers have a plethora of sophisticated algorithms and data structure within.

(iii) Compilers design helps full implementation of high-level programming languages. Support optimization for computers architecture parallelism. A compiler is a program that translates a high-level language (c, c++, Java) into a low-level language. (object program or machine program).

<u>Top down and Bottom up parsing:-</u>

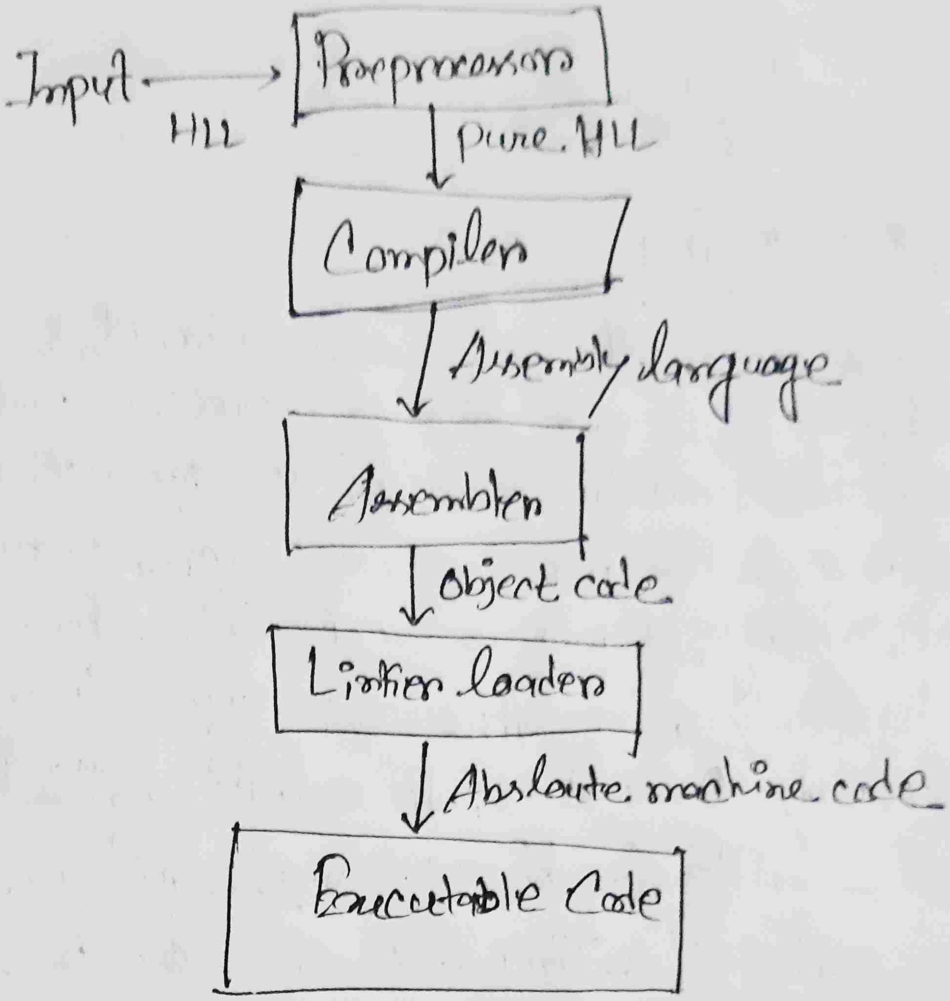
Top down approach starts evaluating the parse tree from the top and move downwards for parsing other nodes. Top down parsing attempts to find the left most derivation for a given string. Top down parsing uses leftmost derivation. Top down parsing searches for a production rule to be used to ~~con~~ construct a string.

Bottom up parsing technique starts evaluating the parse tree from the lowest level of the tree and move upwards for parsing the node. Bottom up parsing attempts to reduce the input string to first symbol of the grammar. Bottom up parsing uses the rightmost derivation. Bottom up parsing searches for a production rule to be used to reduce a string to get a starting symbol of grammar.

Ans to the Q no - (2)

a) Language processing system:-

The computer is an intelligent combination of software and hardware. Hardware is ~~sof~~ simply a piece of mechanical equipment and its functions are being compiled by the relevant software. The hardware considers instructions as electronic charge, which is equivalent to the binary language ~~has only 0~~ in software programming. The binary language has only 0s and 1s. To enlighten, the hardware code has to be written in binary format, which is just a series of 0s and 1s. Writing such code ~~comb~~ would be an inconvenient and complicated task for programmers, so we write programs in a high level language, which is convenient for us to comprehend and memorize. These programs are then fed into a series of devices and operating system (OS) components to obtain the desired ~~code~~ code that can be used by the machine. This is known as a "language processing system".



b) Cross Compiler:-

Compilers are the tool used to translate high-level programming language to low-level programming language. The simple compiler works in one system only, but what will happen if we need a compiler that can compile code from another platform, to perform such compilation, the cross compiler is introduced.

A cross compiler is a compiler capable of creating ~~executable~~ executable code for a platform other than the one on which the compiler is running. For example, a cross compiler executes on machine X and produces machine code for machine Y. In paravirtualization, one computer runs multiple operating systems and a cross compiler could generate an executable for each of them from one main source.

ⓒ Source to source compiler:-

A source to source compiler is also referred to by three other name, the first is the source to source translator, the second is transcompiler and the third one is transpiler. If we try to summarize the work on the source to source compiler is a sentence, it would be as follows:

A source to source compiler is given as input the source code of a program to which it returns a source code with the overall same functionality in the same or different programming language.

Unlike the general compiler whose work is to convert a high-level programming language to a machine language that is binary, the source to source compiler converts one source code from one programming language to another language which is at the same level of compilation from machine language.