

Victoria University of Bangladesh
Dept. of Computer Science & Engineering

Program:- B.Sc in CSIT,

Semester:- Fall 2023

Course title:- Compilers

Course code:- CSI 411

Name:- Husnat Jahan Tanshee

ID:- 2520200011

Batch:- 20

Final Assessment

①

Ans to the Q no - 1(a)

Types of compiler:-

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.

Here are types of compiler:-

① Cross compiler:-

They produce an executable machine code for a platform but this ~~plat~~ platform is not the one on which the compiler is running.

② Bootstrap compiler:-

These compilers are written in a programming language that they have to compile.

③ Source to source / transcompiler:-

These compilers convert the source code of one programming language to the source code of another programming language.

④ Decompiler:-

Basically, it is not a compiler. It is just the reverse of the the compiler. It converts the machine code into high-level language.

Phases of compiler compiler:-

The phases of the compilation process are as follows:-

① Lexical analyzer:-

- ① It takes the high-level language source code as the input.
- ② It scans the characters of source code from left to right.
- ③ It groups the characters into lexemes.
- ④ Each lexeme corresponds to form a token.
- ⑤ It removes white space and comments.
- ⑥ 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.
- ② The grammar for such codes is context free grammars.

③ Semantic analyzer:-

- ① The syntax analyzer verifies the parse tree, checks code validity and produces an annotated parse tree.
- ② ~~It~~ It performs flow and type checking.

④ Intermediate code generator:-

④ The process generates and 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:-

④ The code optimizer converts intermediate code for faster execution using fewer ~~resources~~ resources, removes unnecessary lines and rearranges it while maintaining the source code's meaning.

⑥ Target code generator:-

④ The compiler converts the optimized intermediate code into machine code, marking the final stage of compilation, ensuring that the resulting machine code is relocatable.

Ans to the Q no- 1(b)

④ 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 language that can be

④

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.

④ 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.

Ans to the Q no - 1 (c)

④ Why to learn compiler design:-

① 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.

② Compilers have a plethora of sophisticated algorithms and data structure within

⑤

① Compiler design helps full implementation of high-level programming languages. Support optimization for computer 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).

② Top down and Bottom ~~down~~^{up} parsing:-

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 left most derivation. Top down parsing searches for a production rule to be used to construct a string.

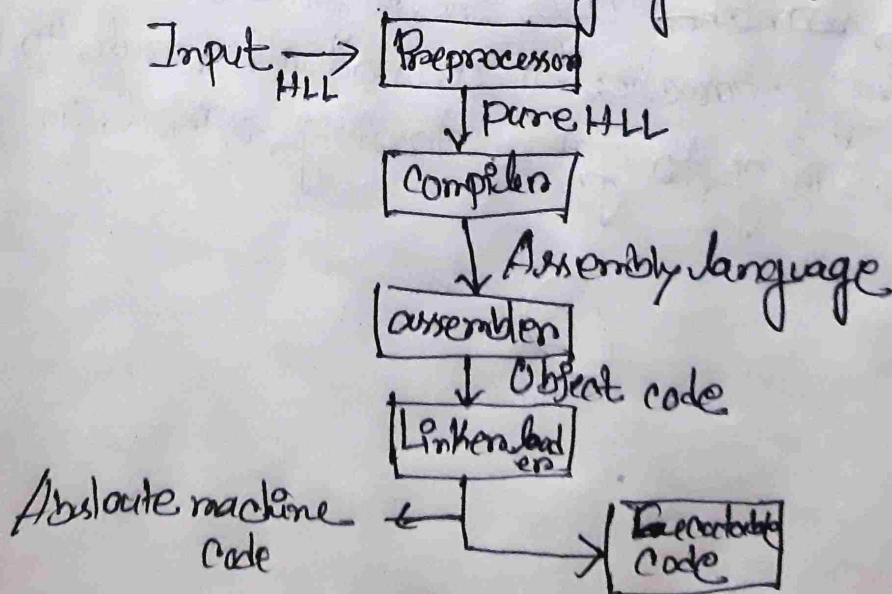
Bottom up parsing technique starts evaluating the parse tree from the lowest level of the tree and move upwards for parsing the node. Bottom up parsing attempts to reduce the input string to first symbol of the grammar. Bottom up parsing searches uses the rightmost derivation. Bottom up parsing searches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

⑥

Ans to the Q no - 2(a)

④ Language processing system:-

The computer is an intelligent combination of software and hardware. Hardware is simply a piece of mechanical equipment and its functions are being compiled by the relevant software. The hardware considers instructions as electronic change, which is equivalent to the binary language 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 would be as inconvenient and complicated task for programmers, so we write programmes in a high level language, which is convenient for us to comprehend and memorize. These programmes are then fed into a series of devices and operating system (OS) components to obtain the desired code that can be used by the machine. This is known as "Language Processing System".



⑦

Ans to the Q no- 2(b)

① Cross compilers:-

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 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.

Ans to the Q no- 2(c)

① Source to source compilers:-

A source to source compiler is also referred to by three other names, 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 in a sentence, it would be as follows:-

⑧

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~~ general compiler whose work is to convert a high level programming language to 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.

— x —

②