Victoria University of Bangladesh
Name: Md. Ziaul Hoque “Sohel”
Student ID: 2221220031
Course Title: Compiler-scaled
Course Code: CSI-411
Batch: 22nd (evening)
Semester: Fall-2023

Ans to the Que No 1(A)

Types Of Compilers:

there are three types of Compilers:

- Single Pass Compilers
- Two Pass Compilers
- Multi pass Compilers

Phases Of Compilers:

The 6 phases of a compiler are:

1. Lexical Analysis
2. Syntactic Analysis or Parsing
3. Semantic Analysis
4. Intermediate Code Generation
5. Code Optimization
6. Code Generation

Ans to the Que No 1(B)

Advantages of Compiler:

1. Improved performance: Compiled code tends to run faster than interpreted code because it has been translated into machine code that can be directly executed by the computer's processor. This can be particularly important for performance-critical applications, such as scientific simulations or real-time systems.
2. Portability: Compilers allow programmers to write code in a high-level programming language that can be easily translated into machine code for a variety of different platforms. This makes it easier to develop software that can run on different systems without requiring significant changes to the source code.
3. Increased Security: Compilers can help improve the security of software by performing a number of checks on the source code, such as checking for syntax errors and enforcing type safety. This can help prevent certain types of vulnerabilities, such as buffer overflows and type coercion attacks.
4. Debugging support: Most compilers include a number of debugging tools that can help programmers find and fix errors in their code. These tools can include features such as syntax highlighting, error messages, and debuggers that allow programmers to step through their code line by line.
5. No dependencies: Your client or anyone else doesn't need any compiler, interpreter, or third-party program to be installed in their system, for executing the shared executable file of your source code.

Disadvantages of Compiler:

1. Compilation time: Depending on the size and complexity of the source code, compilation can take a significant amount of time. This can be a hindrance to productivity if frequent updates to the code are required.
2. Error detection: Compilers can only detect syntax errors and certain semantic errors, and may not catch all errors in the source code. This means that the compiled program may not behave as expected, and debugging may be required to identify and fix the errors.

3. Portability: Programs compiled for a specific platform or architecture may not be able to run on other platforms or architectures without being recompiled. This can be a limitation if the program needs to be run on multiple platforms.
4. Execution speed: Programs compiled from high-level languages may not be as fast as programs written in low-level languages, as the compiled code may include additional instructions for the compiler to interpret.
5. Lack of flexibility: Compilers can limit the flexibility of programs since changes often requires recompilation.
6. Resource consumption: Compilers can consume system resources, particularly during compilation process, which may affect other tasks on the machine.

Ans to the Que No 1(C)

There are several reasons why compilers are used in software development:

- **Ease of programming:** High-level programming languages are easier for humans to read and write than machine code, which is a series of numbers and symbols that can be difficult for humans to understand. By using a compiler to translate high-level language into machine code, programmers can write code more quickly and easily.
- **Portability:** Compilers allow programmers to write code that can be easily compiled and run on a wide variety of devices and platforms. This is because the source code is independent of the underlying hardware and is only translated into machine code when it is compiled.
- **Abstraction:** Compilers provide a level of abstraction between the programmer and the underlying hardware, allowing programmers to focus on the logic of their programs without having to worry about the specific details of the hardware.
- **Performance:** Compilers can optimize the machine code generated from the source code, resulting in faster and more efficient programs.

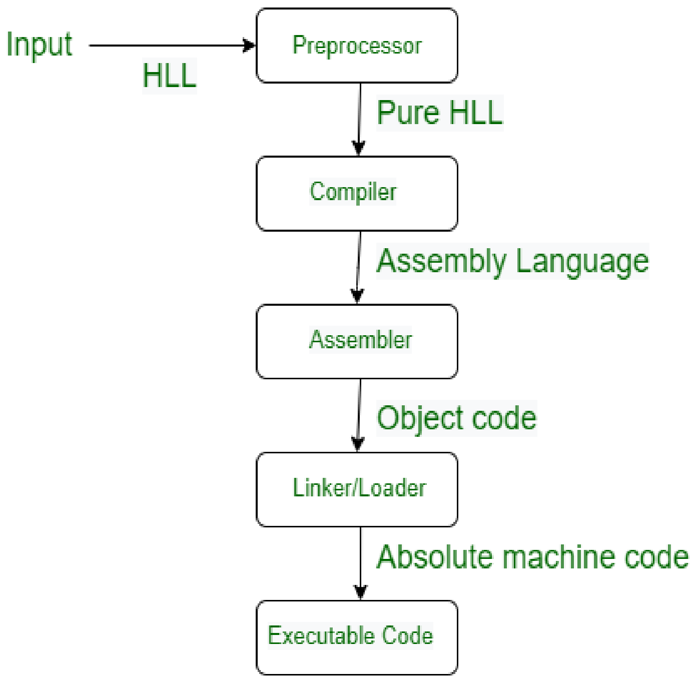
Top-Down Design Model:
In the top-down model, an overview of the system is formulated without going into detail for any part of it. Each part of it then refined into more details, defining it in yet more details until the entire specification is detailed enough to validate the model. if we glance at a haul as a full, it’s going to appear not possible as a result of it’s so complicated For example: Writing a University system program, writing a word processor. Complicated issues may be resolved victimization high down style, conjointly referred to as Stepwise refinement where,

Bottom-Up Design Model:
In this design, individual parts of the system are specified in detail. The parts are linked to form larger components, which are in turn linked until a complete system is formed. Object-oriented language such as C++ or java uses a bottom-up approach where each object is identified first.

Ans to the Que No 2 (A)

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 charge, 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 an inconvenient and complicated task for computer 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 that can be used by the machine. This is known as a language processing system.



Ans to the Que No 2 (B)

cross compiler:

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.

Ans to the Que No 3 (C)

Source Compiler:

A compiler is a software program that transforms a program or code written in a high-level programming language into a low-level machine-readable language. When we write a program or code which can be in a high-level language, such as C, C++.

- END -