Name: Reema Harnekar **Roll no.** : 16CO02

Batch : 01 **Class** : TECO

Experiment No. :1

1. Differentiate between application program and system program.

System Program

- It is the collection of components and art or designing of a given program.
- System program aims to produce software which provides services to the computer hardware.
- System software that executes the application software. It helps in executing the application software.
- These programs allow the application programmer to focus on application to be develop without concerning about the internal details of the System.
- Examples: Assembler Microprocessor, Loader, Linker, Compiler

Application Program:

- These are the set of programs that view computer as a tool for solving a particular problem.
- Application software is a software that are been used by the end user.
- Application programming is used to build application software which includes software like notepad, Word Pad, calculator, web browser, Microsoft excel and many more.
- It interacts with system software which in turn interacts and makes physical hardware functional.
- Examples:Word Processing, Hotel Management System, Accounting Package etc.

System Program	Application Program
1) It is the collection of components and art or designing of a given program.	1) It is the set of programs that view computer as a tool for solving a particular problem.
2) Programming is done using assembly language which interacts with hardware.	2) Application programming is used to build application software which includes software.
3) System software that executes the application software.	3) Application software is a software that are been used by the end user.
4) System programming is used to write low level instructions	4)Application Programming is used to write High level instructions
5)Examples:- Loader, Linker, Compiler	5) Examples:- Library management System, calculator, web browser

2. Difference between system programing and application programing

Application programming is used to build application software which includes software like notepad, Word Pad, calculator, web browser, Microsoft excel and many more. Application software runs on top of system software. It interacts with system software which in turn interacts and makes physical hardware functional.

System programming is used to write low level instructions which are understandable to computer hardware. Programming is done using assembly language which interacts with hardware. This assembly code is specific to hardware. If this code has to be executed on another machine, that machine should have exactly same hardware. System programming is used to make device drivers kind of application. It helps operating system to interact with hardware. Again output is in low level instructions which in turn translated by such drivers to the language understandable by operating system.

3.Compiler

A compiler is a software program that transforms high-level source code that is written by a developer in a high-level programming language into a low level object code (binary code) in machine language, which can be understood by the processor. The process of converting high-level programming into machine language is known as compilation.

Lexical Analysis :

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

```
<token-name, attribute-value>
```

Syntax Analysis :

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.

Semantic Analysis:

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation :

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

4. Assembler

An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. Compilers, on the other hand, must convert generic high-level source code into machine code for a specific processor.
Most programs are written in high-level programming languages and are compiled directly to machine code using a compiler. However, in some cases, assembly code may be used to customize functions and ensure they perform in a specific way. Therefore, IDEs often include assemblers so they can build programs from both high and low-level languages.

5. Loader

The loader is a component of an operating system that carries out the task of preparing a program or application for execution by the OS. It does this by reading the contents of the executable file and then storing these instructions into the RAM, as well as any library elements that are required to be in memory for the program to execute. This is the reason a splash screen appears right before most programs start, often showing what is happening in the background, which is what the loader is currently loading into the memory. When all of that is done, the program is ready to execute. For small programs, this process is almost instantaneous, but for large and complex applications with large libraries required for execution, such as games as well as 3D and CAD software, this could take longer. The loading speed is also dependent on the speed of the CPU and RAM.

Not all code and libraries are loaded at program startup, only the ones required for actually running the program. Other libraries are loaded as the program runs, or only as required. This is especially true for applications such as games that only need assets loaded for the current level or location that the player is in.

Though loaders in different operating systems might have their own nuances and

specialized functions native to that particular operating system, they still serve basically the same function. The following are the responsibilities of a loader:

1. Validate the program for memory requirements, permissions, etc.
2. Copy necessary files, such as the program image or required libraries, from the disk into the memory
3. Copy required command-line arguments into the stack
4. Link the starting point of the program and link any other required library
5. Initialize the registers
6. Jump to the program starting point in memory

6.Linker

In computer science, a linker is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program.

Computer programs are usually made up of multiple modules that span separate object files, each being a compiled computer program. The program as a whole refers to these separately compiled object files using symbols. The linker combines these separate files into a single, unified program; resolving the symbolic references as it goes along.
Dynamic linking is a similar process, available on many operating systems, which postpones the resolution of some symbols until the program is executed. When the program is run, these dynamice link libraries are loaded as well. Dynamic linking does not require a linker.

7.Difference between compiler and interpreter

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

8.Difference between assembler and compiler

The job of the compiler is to take the preprocessed source code and translate it into the assembly code. Then the job of assembler is to takes the assembly code from the compiler and translates it to the machine code.

If we talk about the main difference, then the main difference between assembler and compiler is that compiler takes the source code and translates it into the assembly code whereas assembler takes the assembly code generated by the compiler and translates it into the machine code. The program that is written in a source language is read by compiler. The compiler is a computer program that translates the source code into the assembly language, and this assembly language code is sent to the assembler. There are types of compilers such as single pass compiler, multi-pass compiler, load and go compiler and debugging and optimization compiler. Assembler takes the assembly code generated by the compiler and translates it into the machine code. The main job of assembler is to take the data as input and form the relocatable machine code.

There is compiler available that perform the task of the assembler and directly generate the machine code but that does not mean we cannot study the functions of assembler. There is a lot of difference between assembly code and machine code. All of these different type of compilers perform different functions and are different on the basis of working.

Assembly code is the mnemonic version of machine code whereas machine code uses the binary codes for the representation of operations of memory address. Two passes are preform in assembler that are first pass that identifies the assembly code and store that code in symbol table and then second pass that scan the code again and perform the operations on the code. There are two steps in which compiling is performed one step is analysis part in which the source code is broken into pieces and perform the intermediate representation. In synthesis part the target code is form the intermediate representation. There are phases of the compiler that are a lexical analyzer, syntax analyzer, semantic analyze, intermediate code generate, a code optimizer, code generator, symbol table and error handler.

9. Difference between assembler and interpreter

An assembler can be considered a special type of compiler, which only translates Assembly language to machine code. Interpreters are tools that execute instruction written in some language. Interpreter systems may include a compiler to pre-compile code before interpretation, but an interpreter cannot be called a special type of a compiler. Assemblers produce an object code, which might have to be linked using linker programs in order to run on a machine, but most interpreters can complete the execution of a program by themselves. An assembler will typically do a one to one translation, but this is not true for most interpreters. Because Assembly language has a one to one mapping with machine code, an assembler may be used for producing code that runs very efficiently for occasions in which performance is very important (for e.g. graphics engines, embedded systems with limited hardware resources compared to a personal computer like microwaves, washing machines, etc.). On the other hand, interpreters are used when you need high portability. For example, the same Java bytecode can be run on different platforms by using the appropriate interpreter (JVM).

Name: Reema Harnekar **Roll no.** : 16CO02

Batch : 01 **Class** : TECO

Experiment No. :2

TEXT EDITOR :

A text editor is a computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs). Typically, a text editor provides an "empty" display screen (or "scrollable page") with a fixed-line length and visible line numbers. You can then fill the lines in with text, line by line. A special command line lets you move to a new page, scroll forward or backward, make global changes in the document, save the document, and perform other actions. After saving a document, you can then print it or display it. Before printing or displaying it, you may be able to format it for some specific output device or class of output device. Text editors can be used to enter program language source statements or to create documents such as technical manuals.

WORD PROCESSOR :

A word processor is a computer program that provides special capabilities beyond that of a text editor such as the WordPad program that comes as part of Microsoft's Windows operating systems. The term originated to distinguish text building programs that were "easy to use" from conventional text editors, and to suggest that the program was more than just an "editor." An early user of this term was Wang, which made a popular workstation system designed especially for secretaries and anyone else who created business letters and other documents.

In general, word processors screen the user from structural or printer-formatting markup (although WordPerfect and other word processors optionally let you see the markup they insert in your text). Without visible markup, it's possible to describe a word processor as having aWYSIWYG(what you see is what you get) user interface.

LINE EDITOR :

A line editor is a basic type of computer-based text editor whereby one line of a file can be edited at a time. Line editors were the precursor to document editing software that is commonly used today. Line editors were used before interactive video graphic screens were commonly available in computers. The editing of a single line at a time was due to the unavailability of graphical interface screens, cursors and memory, hence an average computer operator used a teleprinter, whereby a printer was directly attached to a keyboard and changes could not be made once the text had been typed. Typically the text is not entered into the document until a complete line has been typed. The operator can view the typed form once entered, but cannot go back to a previous line to edit it.

DEBUG MONITOR :

A debug monitor is very powerful graphical or console mode tool that monitors all the activities that are handled by the WinDriver Kernel. You can use the debug monitor to see how each command that is sent to the kernel is executed. A

WinDriver Kernel is a driver development toolkit inside ones computer that simplifies the creation of drivers. A driver is used in a computer so that the computer can read the devices that are in the computer or that get attached to the computer. If you were to hook up a printer to your computer, you would first need to install its driver so that the computer could create graphics or a console so that you could control your printer through the computer. The same thing goes for audio devices, internet devices, video devices.

STREAM EDITOR :

The sed stream editor is a text editor that performs editing operations on information coming from standard input or a file. Sed edits line-by-line and in a non-interactive way.

This means that you make all of the editing decisions as you are calling the command and sed will execute the directions automatically. This may seem confusing or unintuitive, but it is a very powerful and fast way to transform text.