**Experiment No 1**

**Name:** Faiyaz Shaikh

**Roll No:** 17CO45

**Case Study**

**System program:**

System programming involves designing and writing computer programs that allow the computer hardware to interface with the programmer and the user, leading to the effective execution of application software on the computer system. Typical system programs include the operating system and firmware, programming tools such as compilers, assemblers, I/O routines, interpreters, scheduler, loaders and linkers as well as the runtime libraries of the computer programming languages.

**Assembler:**

An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer. An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components. An assembler is sometimes referred to as the compiler of assembly language.

**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. The processor executes object code, which indicates when binary high and low signals are required in the arithmetic logic unit of the processor.

**Macroprocessor:**

A general-purpose macro processor or general purpose preprocessor is a macro processor that is not tied to or integrated with a particular language or piece of software. A macro processor is a program that copies a stream of text from one place to another, making a systematic set of replacements as it does so. Macro processors are often embedded in other programs, such as assemblers and compilers. Sometimes they are standalone programs that can be used to process any kind of text. Macro processors have been used for language expansion (defining new language constructs that can be expressed in terms of existing language components), for systematic text replacements that require decision making, and for text reformatting.

**Loader:**

A loader is a major component of an operating system that ensures all necessary programs and libraries are loaded, which is essential during the startup phase of running a program. It places the libraries and programs into the main memory in order to prepare them for execution. Loading involves reading the contents of the executable file that contains the instructions of the program and then doing other preparatory tasks that are required in order to prepare the executable for running, all of which takes anywhere from a few seconds to minutes depending on the size of the program that needs to run.

**Linker:**

It 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.

**Assembler v/s Compiler:**

| Compiler | Assembler |
| --- | --- |
| Compiler converts the source code written by the programmer to a machine level language. | Assembler converts the assembly code into the machine code. |
| Compiler input source code. | Assembler input assembly language code. |
| It converts the whole code into machine language at a time. | But the Assembler can’t do this at once. |
| A Compiler is more intelligent than an Assembler. | But, an Assembler is less intelligent than a Compiler. |
| The compilation phases are lexical analyzer, syntax analyzer, semantic analyzer, intermediate code generated, a code optimizer, code generator, and error handler | Assembler makes two phases over the given input, first phase and the second phase. |
| The output of compiler is a mnemonic version of machine code. | The output of assembler is binary code. |
| C, C++, Java, and C# are examples of compiled languages. | GAS, GNU is an example of an assembler. |

**Compiler v/s Interpretor:**

| Basis for comparison | Compiler | Interpreter |
| --- | --- | --- |
| Input | It takes an entire program at a time. | It takes a single line of code or instruction at a time. |
| Output | It generates intermediate object code. | It does not produce any intermediate object code. |
| Working mechanism | The compilation is done before execution. | Compilation and execution take place simultaneously. |
| Speed | Comparatively faster | Slower |
| Memory | Memory requirement is more due to the creation of object code. | It requires less memory as it does not create intermediate object code. |
| Errors | Display all errors after compilation, all at the same time. | Displays error of each line one by one. |
| Error detection | Difficult | Easier comparatively |
| Pertaining Programming languages | C, C++, C#, Scala, typescript uses compiler. | PHP, Perl, Python, Ruby uses an interpreter. |

**Assembler v/s 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).

**Device Driver:**

A device driver is a particular form of software application that is designed to enable interaction with hardware devices. Without the required device driver, the corresponding hardware device fails to work. A device driver usually communicates with the hardware by means of the communications subsystem or computer bus to which the hardware is connected. Device drivers are operating system-specific and hardware-dependent. A device driver acts as a translator between the hardware device and the programs or operating systems that use it.

**Operating System:**

An operating system (OS), in its most general sense, is software that allows a user to run other applications on a computing device. While it is possible for a software application to interface directly with hardware, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details.

**Editor:**

Editors or text editors are software programs that enable the user to create and edit text files. In the field of programming, the term editor usually refers to source code editors that include many special features for writing and editing code. Notepad, Wordpad are some of the common editors used on Windows OS and vi, emacs, Jed, pico are the editors on UNIX OS. Features normally associated with text editors are — moving the cursor, deleting, replacing, pasting, finding, finding and replacing, saving etc.

**Debugger:**

A debugger is a software program used to test and find bugs (errors) in other programs. A debugger is also known as a debugging tool. A debugger is a computer program used by programmers to test and debug a target program. Debuggers may use instruction-set simulators, rather than running a program directly on the processor to achieve a higher level of control over its execution.

**Interpreter:**

An interpreter is a computer program that is used to directly execute program instructions written using one of the many high-level programming languages. The interpreter transforms the high-level program into an intermediate language that it then executes, or it could parse the high-level source code and then performs the commands directly, which is done line by line or statement by statement.