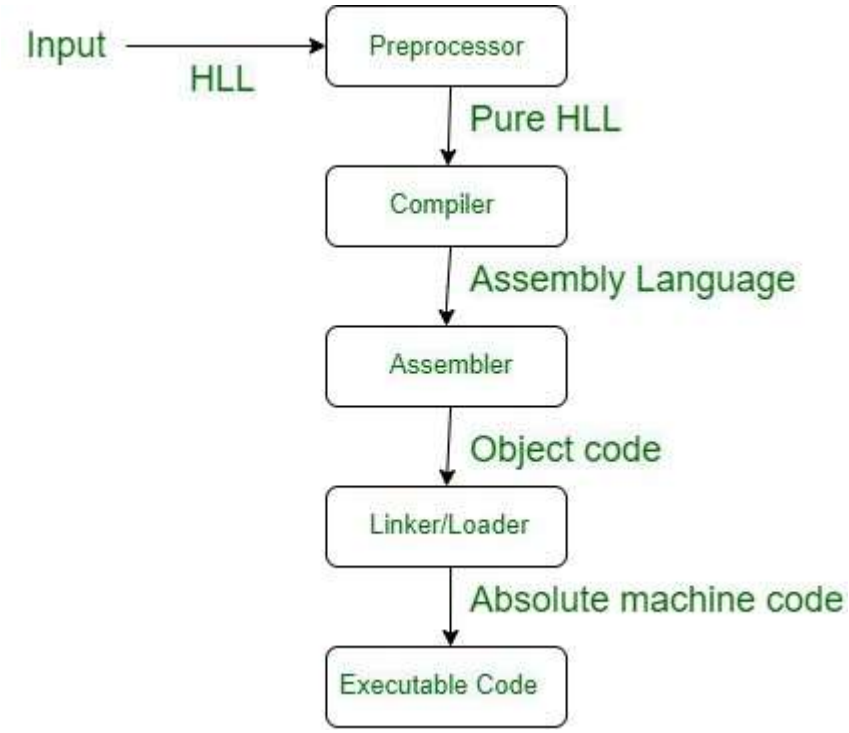
Language Processing System in Compiler Design

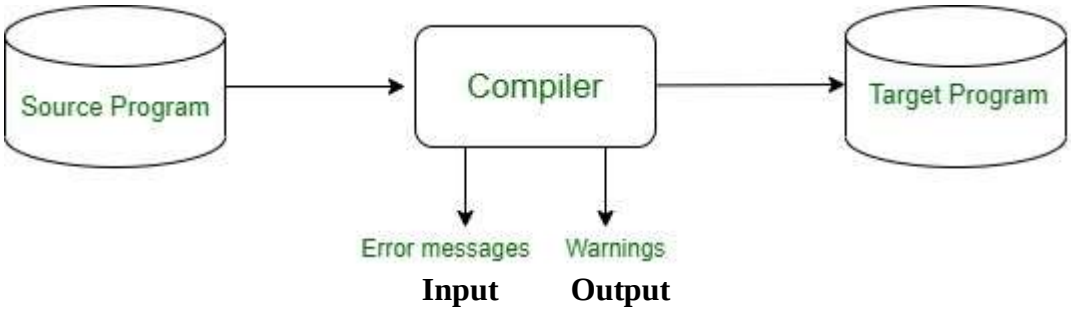


Components of Language processing system:

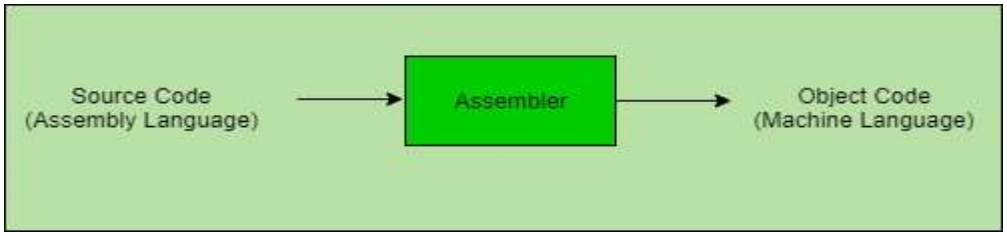
Preprocessor:

It includes all header files and also evaluates whether a macro (A macro is a piece of code that is given a name. Whenever the name is used, it is replaced by the contents of the macro by an interpreter or compiler. The purpose of macros is either to automate the frequency used for sequences or to enable more powerful abstraction) is included. It takes source code as input and produces modified source code as output. The preprocessor is also known as a macro evaluator, the processing is optional that is if any language that does not support #include macros processing is not required.

Compiler: The compiler takes the modified code as input and produces the target code as output.



Assembler: The assembler takes the target code as input and produces real locatable machine code as output.



The Assembler is used to translate the program written in Assembly language into machine code. The source program is an input of an assembler that contains assembly

language instructions. The output generated by the assembler is the object code or machine code understandable by the computer. Assembler is basically the 1st interface that is able to communicate humans with the machine. We need an assembler to fill the gap between human and machine so that they can communicate with each other. code written in assembly language is some sort of mnemonics(instructions) like ADD, MUL, MUX, SUB, DIV, MOV and so on. and the assembler is basically able to convert these mnemonics in binary code. Here, these mnemonics also depend upon the architecture of the machine.

Linker: Linker or link editor is a program that takes a collection of objects (created by assemblers and compilers) and combines them into an executable program.

Loader: The loader keeps the linked program in the main memory.

Executable code: It is low-level and machine-specific code that the machine can easily understand. Once the job of the linker and loader is done the object code is finally converted it into executable code.

Language Processor:

INTERPRETER	COMPILER	ASSEMBLER
Translates high-level language into machine code	Translates high-level language into machine code	Translates assembly language into machine code
Translates source one line at a time	Translates all code at the same time	Uses the processors instruction set to convert
Needed every time you run the program	Only needed once to create an executable file	Runs quickly as conversation between two low level languages is just reliant on the processors instructions set
Returns a list of errors found and on which lines	Will only inform you of the first error it finds	
Runs more slowly as it is being translated every time the code is run	Once compiled runs quickly but compiling can take a long time	

Types of Computer Architecture:

1. **Instruction set architecture (ISA)**
2. **Microarchitecture**
3. **Client-server architecture**
4. **Single instruction, multiple data (SIMD) architecture**
5. **Multicore architecture**
6. **CICS / RISC Architecture**

Instruction set architecture (ISA) is a bridge between the software and hardware of a computer. It functions as a programmer's viewpoint on a machine. Computers can only comprehend binary language (0 and 1), but humans can comprehend high-level language (if-else, while, conditions, and the like). Consequently, ISA plays a crucial role in user-computer communications by translating high-level language into binary language.

Arithmetic/logic instructions: These instructions execute various mathematical or logical processing elements solely on a single or maybe more operands (data inputs).

Data transfer instructions: These instructions move commands from the memory or into the processor registers, or vice versa.

Branch and jump instructions: These instructions are essential to interrupt the logical sequence of instructions and jump to other destinations.

2. Microarchitecture

Microarchitecture, unlike ISA, focuses on the implementation of how instructions will be executed at a lower level. This is influenced by the microprocessor's structural design.

Microarchitecture is a technique in which the instruction set architecture incorporates a processor. Engineering specialists and hardware scientists execute ISA with various microarchitectures that vary according to the development of new technologies. Therefore, processors may be physically designed to execute a certain instruction set without modifying the ISA.

Simply put, microarchitecture is the purpose-built logical arrangement of the microprocessor's electrical components and data pathways. It facilitates the optimum execution of instructions.

3. Client-server architecture

Multiple clients (remote processors) may request and get services from a single, centralized server in a client-server system (host computer). Client computers allow users to request services from the server and receive the server's reply. Servers receive and react to client inquiries.

A server should provide clients with a standardized, transparent interface so that they are unaware of the system's features (software and hardware components) that are used to provide the service.

Clients are often located on desktops or laptops, while servers are typically located somewhere else on the network, on more powerful hardware. This computer architecture is most efficient when the clients and the servers frequently perform pre-specified responsibilities

4. Single instruction, multiple data (SIMD) architecture

Single instruction, multiple data (SIMD) computer systems can process multiple data points concurrently. This cleared the path for supercomputers and other devices with incredible performance capabilities. In this form of design, all processors receive an identical command from the control unit yet operate on distinct data packets. The shared memory unit requires numerous modules to interact with all CPUs concurrently.

5.Multicore architecture

Multicore is a framework wherein a single physical processor has the logic of multiple processors. A multicore architecture integrates numerous processing cores onto only one integrated circuit. The goal is to develop a system capable of doing more tasks concurrently, improving overall system performance.