03 - 09 - 2024

Version :- Java 8

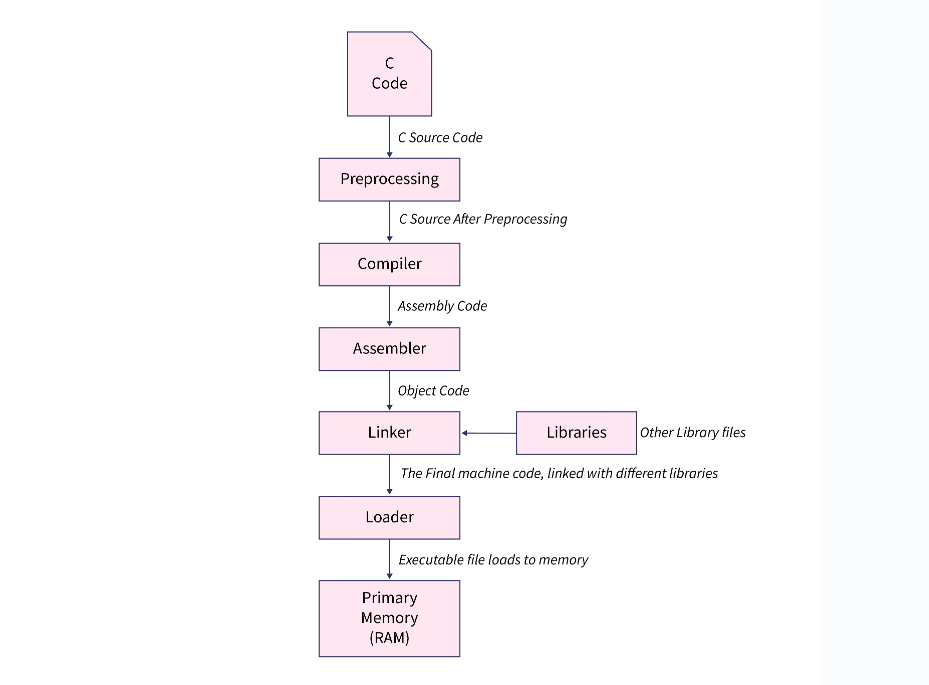
Updates

* Lambda expressions
* Stream API
* Type annotations
* Date Time package
* Update in Concurrancy
* Compact profiles
* The Rhino javascript engine has been replaced with the [Nashorn](http://docs.oracle.com/javase/8/docs/technotes/guides/scripting/nashorn/) Javascript Engine
* JVM changes permagen

Flow of execution in c/c++

**What are the Steps in the Execution of the C Program?**

Below given the image of the steps of execution of the C program into the C compiler.



Now let us discuss the steps of execution of the C program in detail :

**C code:** When you first write the code in the C language, that source code is sent to the Preprocessing section.

**Preprocessing:** In this section our source code is attached to the preprocessor file. Different types of header files are used like the studio.h, math.h, etc. Our C source code is attached to these types of files and the final C Source generates. (some of the preprocessor directives are #include,#define). After generating the preprocessed C source code, the C source code is sent to the compiler section.

**Compiler:** The preprocessed source code moves to the compiler, and an assembly-level code is generated by the compiler after the compilation of the whole C source code program. All the different files which have the C program must be saved with the .c extension. For the compiler to understand whether the program file is a C program file or not, it is necessary to have the '.c' extension. Suppose there is a program file named as first.c, The file first.c will be the source file which will consist of the C source code of the program. Now, when the file is compiled, the first thing the C compiler does is to search for any error. If there is no error, the C compiler will report for no error, after that the compiler will store the file as a .obj file of the same name, which is termed as the object file. So by this process, the compiler will create the first.obj. Although this .obj file will not be executable. After the compilation, the process is continued by the assembler section.

**Assembler:** This part usually generates the Object code, after taking the assembly-level code from the compiler. This object code is quite similar to the machine code or the set of binary digits. After this assembler part, The Linker continues the process, producing an executable.exe file at the end.

**Linker:** Before getting started with this, we should know that the library functions are a part of the C software but not of any C program. Hence, the compiler has no idea about the working of the function, whether it is a printf function or scanf function. The information for each of these functions is kept in the corresponding library, which the compiler ought to be able to connect. The linker does this task. So, when the #include is written, it includes the studio.h library, which is basically used for giving access to the Standard Output and Input. The basic goal of the linker is to link the object file to the library functions so that the programme may be run as an executable file (.exe). In this Linker process, the first.exe file will be created and this file is in an executable format. After this process, the next step is the loader process.

**Loader:** Whenever the command is given for the execution of a particular program, The loader plays an important role. With the help of the loader, the .exe file is loaded in the RAM and the CPU is informed of the starting point of the address of the program where it is loaded.

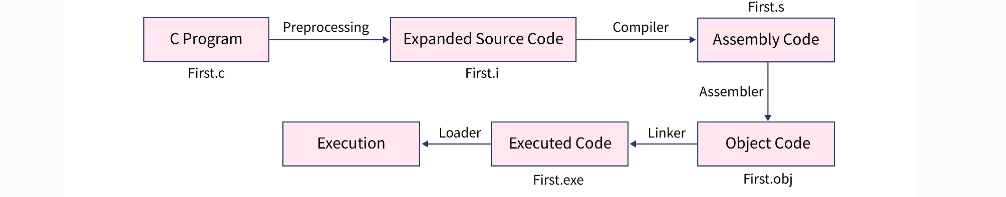
Let us understand the above diagram in detail:

**Instruction Register:** The current instructions which are executed by the CPU, is held by this.

**Program Counter:** The address of the next instructions which is to be executed by the CPU, is held by this.

**Accumulator:** The accumulators usually store all the information which are related to the calculations.

The first instruction is informed by the loader to the Program Counter, and after that, the execution is initiated, and the Program Counter handles the task.



**Platform**

Os = h/w based platform

Also s/w only platform

Java is Programming language , Technology as well as Platform.

Software development kit of java = jdk

**src.zip vs rt.jar vs java api docs**

hello world

java application execution flow

overview of jvm architecture

**compiler switch**

gcc --help

-save -temps =>

main.i => intermediate file => header file

windows

.dll .o

linux

.lib .so

java

.jar

**Lang types**

Machine level

Low level

High level language

**Programming styles**

Oop

Procedural

Functional

**Operators :**

3 type

- unary

- Bunary

- ternary

Logger file

**Framework**

To solve specific problem already written code . it can contain one to many libraries

**Technology**

Lang is use to implement business language

Tech is use to create s/w

Every tech is not a language

Using tech we develop application

**Jre**

Platform provide execution environment

webRunner – a java browser

java can run in this webRunner and the class that run in this browser is Applet

**standardisation**

ansi - c

iso - cpp

ecma - js , c#

jcp – java

**Java API**

In java , java api is nothing but java class provided by sun/oracle

Jdk = sdk for java

Jre = rt.jar + jvm

Install = deploy

04 - 09 - 2024

Java has only access modifier

Variable declared in class = field

**System.out.prinln();**

System is final class declared in lang package

Out is final static variable of Printstream type

PrintStream is in io pkg

Println is method in PrintStream class

**Difference between src.java vs rt.java vs java docs**

Src.java contains all source code

It also contains java folder which contain all packages shuch as io lang etc

Also rt.jar contains all compiled code of java

Rt.jar is in jre

**Workspace**

Folder contains multiple project

For every class java compiler creates .class file

Difference between c and cpp

Difference between cpp and java

Memory leakage in java

**Pillors of OOP**

In oop there are minor and major pillors

Major pillors

* Abstractions
* Encapsulations
* Modular
* Hierarchical

Minor pillors

* Polymorphism
* Concurrency
* Persistence

**Buzzword or features of java**

Simple

Java syntax is simple than c / cpp

Architectural Neutral

Java code is not specific to cpu

Developer do not write code to target

Portable

Size of data on any type of machine 64 bit or 32 bit is same

.java file i.e. source ccode and .class file are portable

Note : jvm is platform dependent

Java is Multithreaded language

**JVM components**

https://dzone.com/articles/jvm-architecture-explained

1. Class loader subsystem

Comp. of jvm provide 3 type of class loader

And dev. Can provide custom class loader

1. Bootstrap class loader

Jre->lib bootstrap path

Src

1. Extension class loader

Java api from Ext directory inside lib

1. System class loader

To load classes by developer

1. Custom class loader
2. Runtime data areas

5 diff areas

1. Method area
2. Heap
3. Java stacks
4. Pc registers
5. Native method stacks
6. Execution engine
7. Interpreter
8. Jit compiler : for optimisation
9. Garbage Collector

Native method interface

Java follow c syntax

And cpp concepts

Jvm concept if from pascal

Java is made by developers who works in unix