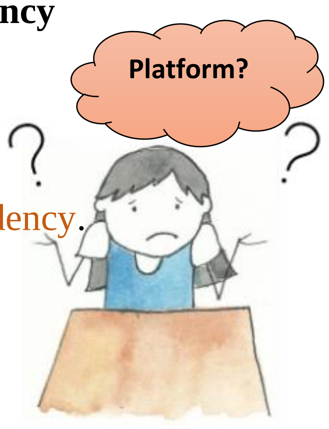
PLATFORM DEPENDENCY

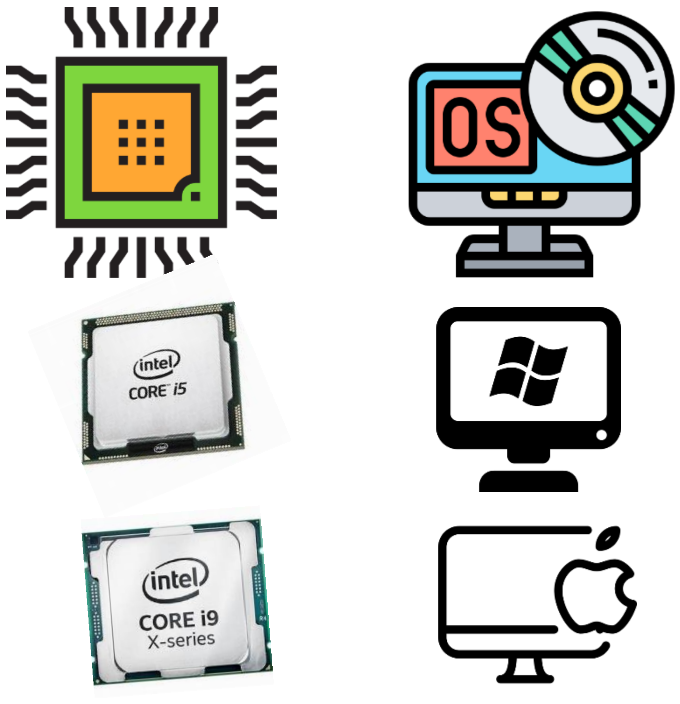
One of the most important features of java is **platform independency** and to understand how java achieved platform indecency, one must learn what is platform dependency.



Let us understand what is **platform** to understand platform dependency.

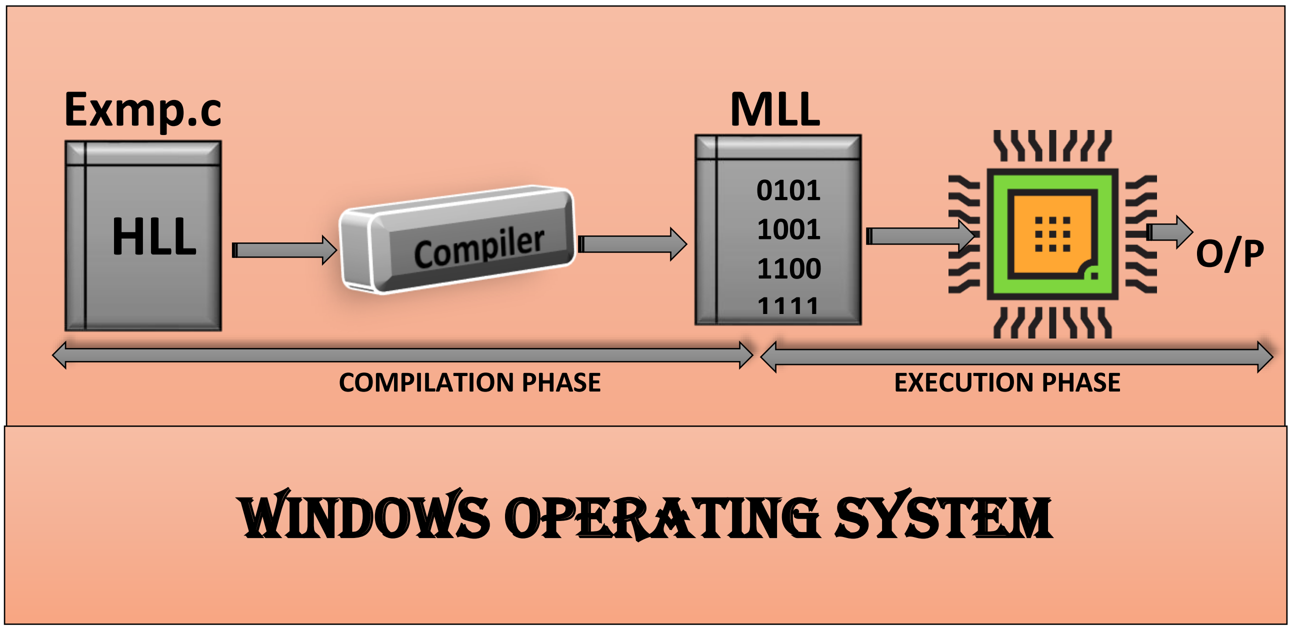
Platform is a combination **of hardware and software.**
Hardware mainly refers to microprocessor and software refers to Operating system.
For example, platform in your computer could be the combination of i5 processor and windows OS or it could be i9 processor and mac OS as shown below. However, from Software engineer's perspective, platform only refers to **OS.**

Platform = Hardware + Software



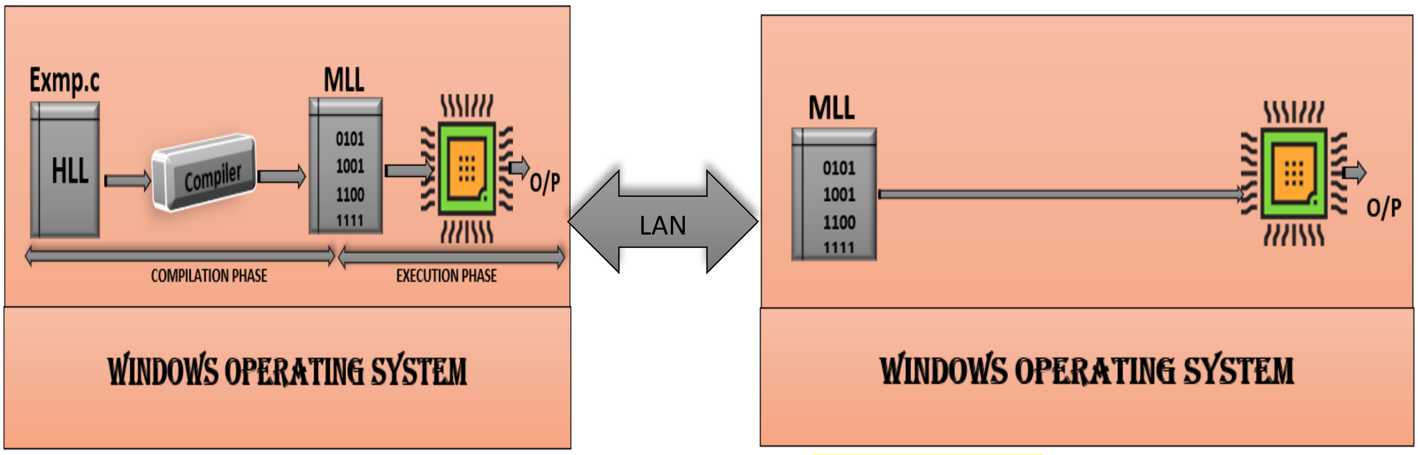
After getting to know what is platform let us now understand platform dependency by considering different cases of ***C language*** *which is very much platform dependent.*

CASE I)

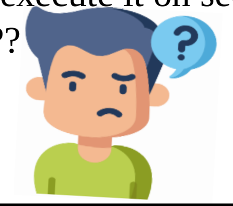


In the above case, we considered a file of c programming language as the extension is exmp.c. Since C programming language is platform dependent programming language, its platform of execution and platform of compilation must be same. In the above case the platform of compilation and platform of execution is same which is windows operating system and hence we got the output.

CASE II)



In the above case, we again considered a file of c programming language as the extension is exmp.c. In this case, we considered two computers having same operating system connected via LAN connection. If we now take a copy of MLL file from first computer and try to execute it on second computer, according to you will the file execute????

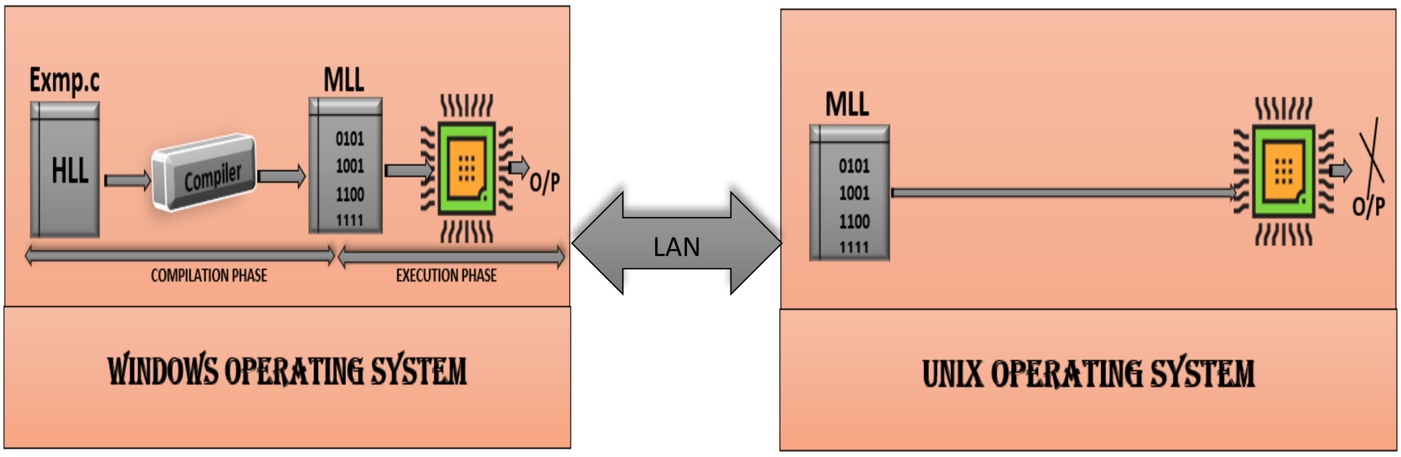


If incase you are still confused, just check two phases.

COMPILATION PHASE	EXECUTION PHASE
Windows	**Windows**

If you can refer the above table, compilation happened on windows and execution happened on windows.
Since platform of compilation is same as that of platform of execution, the file gets executed which simply means you will get the output.

CASE III)



This case is same as that of case ii except the platform of second computer that is

COMPILATION PHASE	EXECUTION PHASE
WINDOWS	**UNIX**

If you can refer the above table, compilation happened on windows and execution happened on Unix.
Since platform of compilation is not same as that of platform of execution, the file will not get executed which simply means you will not get the output.

The problem faced in C or C++ programming language is that their compiler directly converts High-level language code to machine level language code and if this code is copied on other type of operating system and tried to execute, it doesn’t work because the code was written using platform dependent programming language.

To reslove this issue,

JAMES GOSLING, the inventor of java introduced a programming language which was platform independent.

LET US NOW LEARN HOW JAVA ACHIEVED PLATFORM INDEPENENDCY.

Let us assume you are writing code using java in your computer which has windows os. Since Machine understands machine level code not your high level code, conversion must happen.

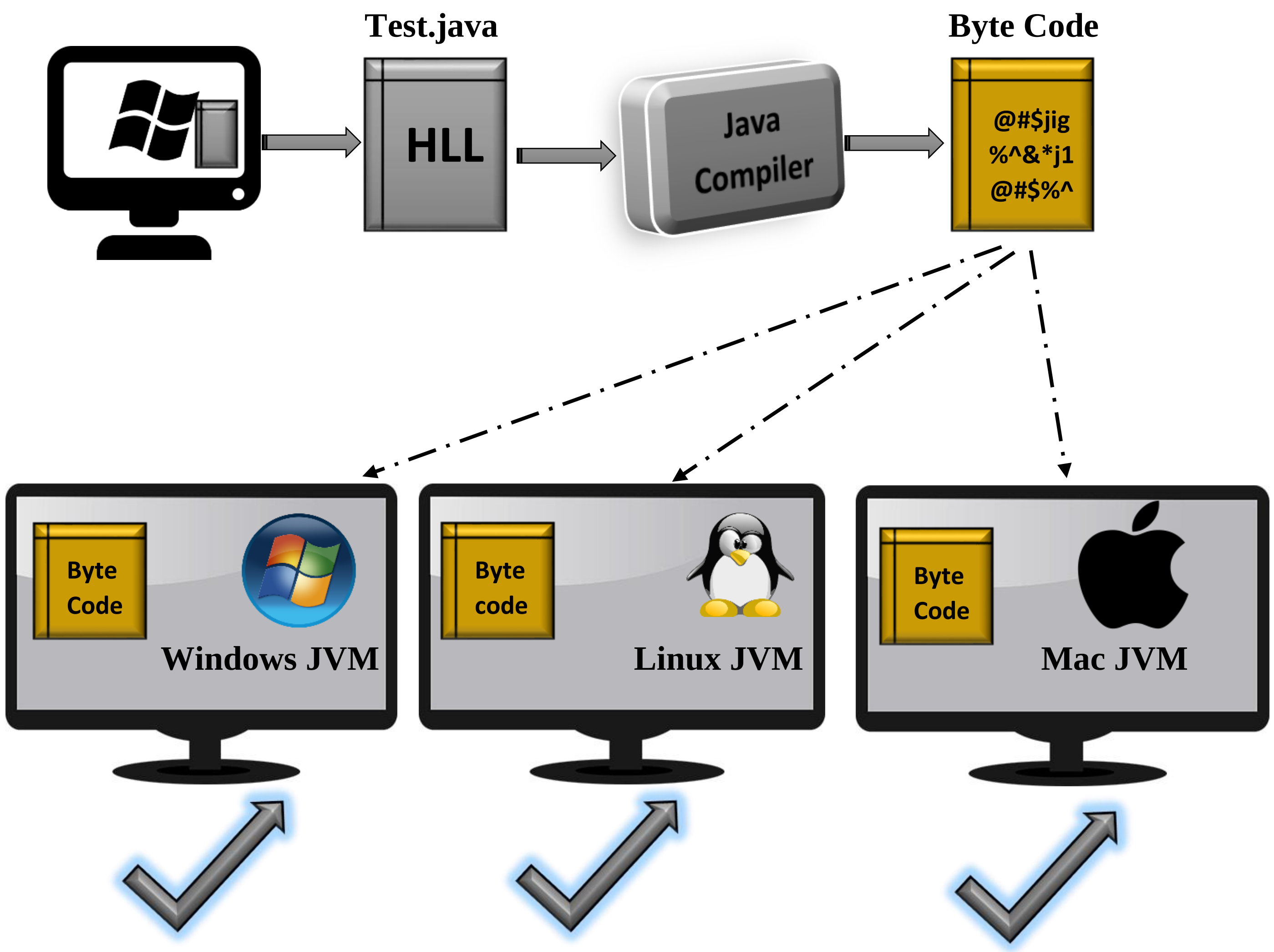
Let us see how exactly conversion happens in java.

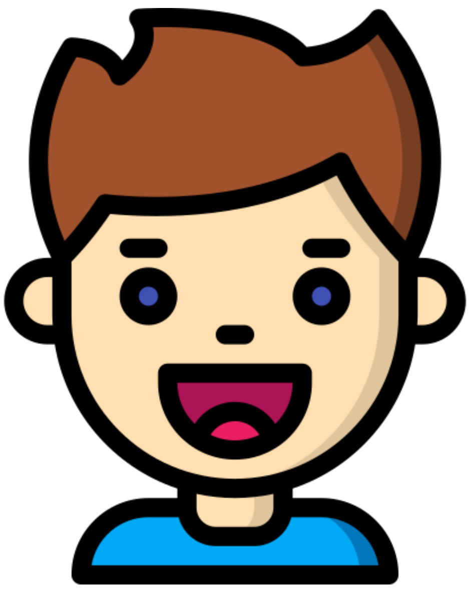
Initially your HLL code is given as input to compiler but java compiler will not give MLL code as ouput like c and c++ compiler rather it takes HLL as input and gives a special type of code as output called as **byte code** which is platform independent. Byte code is neither hll code nor mll code, hence it is also referred to as intermediate code.

If you can recollect machine understands only MLL code but java compiler gave you byte code. To resolve this, James Gosling provided a software called as **JVM(Java vitual Machine)** which was platform dependent that is different OS have different JVM. Since you are writing code on windows OS, you will have to download windwos compatible JVM . JVM will now convert byte code to machine level code which machine can easily understand.

In this way, java achieved platform independency using a special type of code which is **byte code.**

Have a look at the figure shown below which justifies the explaination given above.





He Understood how Java achieved Platform independency.

Did You?

Key Points

1)Java is a platform independent programming language.
2) Java Compiler takes high-level language as input and converts it into byte code.
3) Byte code is neither HLL code not MLL code, it is inbetween and hence it is also called as intermediate code.
4) Byte code is Platform independent where as machine level code is platform dependent.
5) JVM is a software which takes byte code as input and convetrs it into MLL code.
6) JVM is platform dependent.
7) Different OS have different JVM.
8) Java achieved platform independency by making use of byte code as it is a code which is platform independent.
9) Byte code is stored in class file.
10) Platform independency is also refrred as Portability.
11) Java is a portable programming language.
12) One of the disadvantage of java is, it is slow in execution when compared to c and c++. This is due to the extra step involved in the conversion of HLL code to MLL code which is converting HLL to byte code and then byte code to MLL code.

The first electronic computer
ENIAC weighed more than
27 tons and took up
1800 square feet.